A STUDY IN QUANTUM CHEMISTRY THROUGH APPLICATIONS AND PEDAGOGY

by

KEVIN V. MURPHY

(Under the Direction of Henry F. Schaefer III)

ABSTRACT
Quantum chemical theory enables a computational investigation of the electronic structure and properties of small molecules with an accuracy that frequently matches, or even exceeds, experiment. A theoretical investigation of the World War I chemical weapon phosgene utilizing such methods is presented. The results include a description of phosgene’s molecular geometry and fundamental frequencies, acquired with the most rigorous coupled-cluster methods currently available. Another small, high-symmetry system of inquiry is thiosulfine, the sulfur analog of the Criegee intermediate, an ephemeral molecule that plays an essential role in many relevant processes, including those of the atmosphere. Due to its transience, the Criegee intermediate long eluded experimental interrogation. Efforts to investigate its mechanism of reactivity have included studies of the chalcogen analog thiosulfine, which has yet to be isolated in the gas phase. A map of thiosulfine’s potential energy surface, as it undergoes cyclization to the the valence isomer dithiirane, is presented, aided by natural bond orbital methods. Finally, a detailed, pedagogical treatment of perhaps the single most important set of entities in quantum chemistry, the molecular integrals, is included. The materials included in this chapter are designed to help students and researchers better grasp this elusive subject via detailed and readable descriptions and exercises.

INDEX WORDS: phosgene, Criegee intermediate, sulfur, electronic structure theory, molecular integrals, chemical education, quantum chemistry
A STUDY IN QUANTUM CHEMISTRY THROUGH APPLICATIONS AND PEDAGOGY

by

KEVIN V. MURPHY

B.S., Eastern Oregon University, 2010

A Dissertation Submitted to the Graduate Faculty
of the University of Georgia in Partial Fulfillment
of the Requirements for the Degree

DOCTOR OF PHILOSOPHY

ATHENS, GEORGIA
2018
A STUDY IN QUANTUM CHEMISTRY THROUGH APPLICATIONS AND PEDAGOGY

by

KEVIN V. MURPHY

Major Professors: Henry F. Schaefer III
Committee: Geoffrey Smith
          Gary Douberly

Electronic Version Approved:
Suzanne Barbour
Dean of the Graduate School
University of Georgia
May 2018
ACKNOWLEDGEMENTS

Upon my arrival at the Center for Computational Quantum Chemistry (CCQC) Dr. Jay Agarwal was tasked with mentoring me in research. Despite my lack of significant experience, his willingness to let me explore different research avenues, and his expert guidance of my forays into high-accuracy theoretical studies, allowed me to be productive and learn a great deal.

Dr. Justin Turney took over the role of research mentor a little over half-way through my program. When I was advised to consider writing a J. Chem. Ed. article to support my professional interests, he was generous and flexible in taking on this much different kind of project. His expertise and patience were invaluable in seeing the project through to completion.

Professor Schaefer is unparalleled in his work ethic as well as his graciousness. I will be forever grateful to him for permitting me, an unlikely candidate for theoretical chemistry, to join his army of ambitious, brilliant young scientists. He supported me even when my aspirations took unexpected turns. He is an inspiration, and I am grateful to have had him as a doctoral advisor.

Kathryn Juras and Sybil Zimmerman provide the essential infrastructure to the CCQC, as well as do much to add to its kind, positive atmosphere. They have helped me on countless occasions, and I give them both my sincerest thanks.

My family, especially my mom and dad, have been unflagging in their support as my interests evolved. I am very lucky to have grown up in such a warm, eclectic, and nurturing environment. Thank you, Mom and Dad, and to the whole Murphy clan.

Finally, and most importantly, my dear wife Amara has been a constant source of support: practical, psychological, emotional. She has helped see me through many periods of doubt, a fellow traveler on this dynamic, challenging journey, a companion of incredible patience and understanding. I cannot begin to express the depth of my gratitude and appreciation for her.
# TABLE OF CONTENTS

## ACKNOWLEDGEMENTS

### 1 INTRODUCTION & LITERATURE REVIEW

1.1 Molecular Orbital Theory and the Hartree–Fock Method ........................................ 1  
1.2 Post-Hartree–Fock Methods to Treat Electron Correlation ........................................ 4  
1.3 Potential Energy Surfaces and Geometry Optimization .............................................. 8  
1.4 Fundamental Frequencies .............................................................................................. 14  
1.5 Natural Bonding Orbitals ................................................................................................. 17

## 2 PHOSGENE AT THE COMPLETE BASIS SET LIMIT OF CCSDT(Q): MOLECULAR STRUCTURE AND ROVIBRATIONAL ANALYSIS\(^1\) ................................................................. 19

2.1 Abstract .......................................................................................................................... 20  
2.2 Introduction ..................................................................................................................... 20  
2.3 Theoretical Methods ....................................................................................................... 22  
2.4 Results and Discussion ................................................................................................... 25  
2.5 Conclusion ...................................................................................................................... 30

## 3 THIOFORMALDEHYDE S-SULFIDE, SULFUR ANALOG OF THE CRIEGEE INTERMEDIATE: STRUCTURES, ENERGETICS AND ROVIBRATIONAL ANALYSIS\(^2\) ............................................................................ 32

3.1 Abstract ............................................................................................................................ 33  
3.2 Introduction ..................................................................................................................... 33  
3.3 Theoretical Methods ....................................................................................................... 35  
3.4 Results ............................................................................................................................. 37  
3.5 Conclusion ...................................................................................................................... 47

---

\(^1\) This section discusses phosgene, a halogenating agent, focusing on its molecular structure and rovibrational analysis.  

\(^2\) This section explores thioformaldehyde, focusing on its sulfur analog as a Criegl intermediate, including structural, energetic, and rovibrational analyses.
CHAPTER 1

INTRODUCTION & LITERATURE REVIEW

In the following introductory chapter, a broad overview of the quantum chemical methods employed in this dissertation is provided. The chapter begins with a discussion of Hartree–Fock theory, which is effectively the computational form of molecular orbital (MO) theory and the core method for solving the Schrödinger equation for a molecular system on a computer. This is followed by a discussion of efforts to redress the deficiencies of Hartree–Fock results with the aptly named Post–Hartree–Fock methods. These methods serve to append corrections to the electronic energy expectation value. The production of high-accuracy geometries is a cornerstone of the present research, and the basis of geometry optimization is described. The equations for predicting molecular geometry are closely associated with the prediction of infrared frequency bands, and a description of fundamental frequency computations, also pervading the current research, is included. Finally, much chemical insight has been gleaned from these electronic structure calculations by adopting the Natural Bond Orbital framework, and some details of this methodology are included.

1.1 Molecular Orbital Theory and the Hartree–Fock Method

The description of an electron as a spherical charge with a definite trajectory is not always appropriate. Quantum mechanics proffers a mathematical description of the electron vis-à-vis the wave function, $\Psi$, which is related, but not equivalent, to a description of the electron in terms of its probability density.\textsuperscript{1–3}

For all but the simplest systems the exact mathematical form a wave function for the electron of a molecular system takes is impossible to derive. Molecular wave functions $\Psi$ are approximated as combinations of molecular orbitals $\psi_i$, which themselves are linear combinations of atomic orbitals $\phi_\mu$. The mathematical form of these atomic orbitals $\phi_\mu$ are inspired by the exactly known hydrogen orbitals.
The many-electron wave function must be antisymmetric, meaning that it must change sign upon the interchange of any two electrons’ four (three spatial and one spin) coordinates. This stems from experimental observations indicating that no two electrons can simultaneously occupy the same state, or equivalently, be described by the same quantum numbers. This is achieved by expanding the molecular orbitals in a determinant.\(^4\)

The molecular orbitals, in the atomic orbital basis, are variationally optimized during the Hartree–Fock–Roothan–Hall (more simply, Hartree–Fock) procedure.\(^3\) Because the procedure is variational in nature, it is subject to the variational principle, which holds that any such optimized system will have an energy greater than or equal to the exact energy.\(^2\) Thus, the expectation value for the total electronic energy, which stems directly from the variationally optimized orbitals, will never be below that of the exact energy.

\[
E_{\text{HF}} \geq E_0
\]  

(1.1)

The foremost effort in quantum chemistry methods development is to find means by which the energy may be brought ever closer to that which is exact, \(E_0\). An essential approximation of Hartree–Fock (HF) theory belies the tremendously complex interdependence of electron behavior. For a system of \(N\) electrons, the effect on the \(N\)th electron of the remaining \(N-1\) electrons is replaced by an average field which depends only on the coordinates of the electron whose individual wave function, or orbital \(\psi\), is being computed, which makes up part of the many-electron wave function \(\Psi\).

The Fock operator, \(\hat{F}_1\), is central to the Hartree–Fock method, and is analogous to a one-electron Hamiltonian.\(^5\)

\[
\hat{F}_1 = \hat{h}_1 + \sum_u \{ \hat{J}_u(1) - \hat{K}_u(1) \}
\]  

(1.2)

Its eigenfunctions are spin-orbitals. It contains three terms: a kinetic energy term, a potential energy interaction between the electron and all nuclei, and an average field term which represents the electron’s interaction with all of the other electrons. This mean field is known as the HF potential. The average field is calculated using the spin-orbitals of all the electrons.\(^6\) Because the
Fock operator depends on its eigenfunctions, solving for spin-orbitals is a non-linear eigenfunction and must be solved iteratively. In this context, this procedure is known as a self-consistent field (SCF) procedure. Once the average fields of two successive iterations agree to a specific tolerance, the procedure is complete.

The HF potential is made up of a sum over the Coulomb and exchange operators, which are integrals over all space that represent the Coulomb repulsion and non-classical exchange terms arising from the antisymmetric properties of the Slater determinant, respectively.

In order to evaluate the expectation values of the Fock operator in the atomic orbital basis, which is typically expressed as

$$ F_{\mu\nu} = T_{\mu\nu} + V_{\mu\nu} + \sum_{\lambda\sigma} D_{\lambda\sigma} \left( G_{\mu\nu\sigma\lambda} - \frac{1}{2} G_{\mu\lambda\sigma\nu} \right), \quad (1.3) $$

solutions to the so-called molecular integrals must be found.\(^7\)

In matrix notation, the Fock equation to be iteratively solved is

$$ \mathbf{F}\mathbf{C} = \mathbf{S}\mathbf{C}\mathbf{\epsilon}. \quad (1.4) $$

The molecular orbitals \( \mathbf{C} \) are linear combinations of atomic orbitals; \( \mathbf{S} \) is the overlap matrix, necessary because the atomic orbital basis functions are non-orthogonal; \( \mathbf{\epsilon} \) is the vector of orbital energies. Hence, there are four so-called “energy” molecular integrals: the overlap integral, and three expectation values of the Fock operator.

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

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

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

$$ G_{ABCD} = \int \phi_A \phi_B \left( \frac{1}{r_{ij}} \right) \phi_C \phi_D \, d\tau \quad (1.5) $$

The atomic orbitals \( \phi_\mu \) are represented by linear combinations of Gaussian functions.\(^3\) Chapter 4 details the derivation and implementation of these equations in a pedagogical fashion.
1.2 Post-Hartree–Fock Methods to Treat Electron Correlation

Hartree–Fock (HF) theory correctly addresses the correlation between electrons of like spin, which are never found at the same point in space. One defect of HF theory is the absence of correlation between electrons of opposite spin. Clearly electrons of opposite spin are correlated, never found at the same point in space, and so HF theory is unsatisfactory here. There exist several methods for treating such electron correlation, which makes up part of what is generally expressed as

\[ E_{\text{corr}} = E_{\text{HF}} - E_0. \]  

(1.6)

There are two types of correlation: static and dynamic.

Electrons in the HF model do not instantaneously interact with each other, as they do in reality, but rather each and every electron interacts with the average, or mean, field created by all other electrons. Classically speaking, each electron moves in such a way as to avoid those regions of close proximity to all other electrons. The failure of the HF model to correctly reproduce this behavior is the first source of \( E_{\text{corr}} \). Such correlation is called dynamic correlation since it is directly related to electron dynamics.\(^8\)

The wave function in the HF model is a single Slater determinant, which for some systems is a rather poor representation of a many-electron system’s state. In certain cases an electronic state can be well described only by a linear combination of more than one (nearly-)degenerate Slater determinants. This is another reason why \( E_{\text{HF}} \) may differ from \( E_0 \). The corresponding type of correlation is called static, or non-dynamic, since it is not related to electron dynamics.\(^3\)

Interestingly, both static and dynamic correlation effects can be taken into account by mixing in Slater determinants \( \Phi_i \) to the Hartree–Fock “reference,” \( \Phi_0 \).

\[ \Psi = c_0 \Phi_0 + \sum_i c_i \Phi_i \]  

(1.7)

Here, if \( c_0 \) is assumed to be close to 1 and a large number of excited determinants \( \Phi_i \) are added, each of which is assumed to give only a small contribution, then the method primarily treats dynamic correlation. If, on the other hand, it is assumed that there are just a few excited determinants \( \Phi_i \) with weights close to that of the reference determinant \( \Phi_0 \), then the method primarily treats static
correlation. We discuss two methods for treating dynamic correlation (configuration interaction and coupled-cluster theory), because these are used in the dissertation. For completeness, we also briefly address a static correlation method (multi-configuration SCF).

Dynamic Correlation

The essential premise of modern configuration interaction (CI) entails the employment of a reference wave function, typically the HF closed shell wave function – symbolized with Φ₀ – and systematically replacing the occupied spin-orbitals with virtual ones.⁵ Letting r, s, . . . represent occupied orbitals, and a, b, . . . represent virtual ones, we would seek to write

\[ \Psi = \Phi_0 + \sum_r \sum a c_r^a \Phi_r + \sum_r \sum s \sum a \sum b c_{r,s}^{a,b} \Phi_{r,s} + \cdots \]  

(1.8)

The expansion coefficients can be found by solving the variational problem in the usual way.⁹ This involves finding the eigenvalues and eigenvectors of the Hamiltonian matrix; usually one is only interested in a few of the electron states, and methods are available for finding just a small number of eigenvalues and eigenvectors of very large matrices.

In a full CI calculation we would include every possible excited configuration formed by promoting electrons from the occupied spin-orbitals to the virtual ones; for a virtual closed-shell singlet state molecule with m electron pairs and n basis functions there are

\[ \frac{n!(n+1)!}{m!(m+1)!(n-m)!(n-m+1)!} \]  

(1.9)

possible Slater determinants. This number can be very large, and so we must truncate the CI expansion.

In the CI singles (CIS) model only singly excited states are included in addition to \( \phi_0 \) (i.e., \( \sum_a c_r^a \Phi_r^a \)). When the reference wave function is the closed-shell HF wave function \( \Phi_0 \), then the single excitations don’t mix with the ground state and CIS tends to be used as a low-level method for studying excited states.

The next logical step is to include the double excitations. When this is done together with the single excitations, then this is described as the CISD (CI singles and doubles) model. If the double
excitations alone are included, then it is the CID (CI doubles) method.

Generally, all such truncations lead to a problem called size-consistency.\textsuperscript{10} If we have an ensemble of \( n \) particles and their energy is \( n \) times the energy of a single particle, then it is said that the theory “scales correctly” (or that the method is size-consistent). While full CI calculations scale correctly, truncated CI expansions do not.

\textit{Coupled-cluster theory} was developed in part to redress the size-consistency problem of configuration interaction.\textsuperscript{9} The fundamental equation of coupled-cluster theory relates a HF wave function \( \Phi_0 \) to the best possible wave function \( \Psi \) by

\[
\Psi = \exp(\hat{T})\Phi_0 .
\] (1.10)

The exponential factor is defined by a Taylor-series expansion

\[
\exp(\hat{T}) = \sum_{k=0}^{\infty} \frac{\hat{T}^k}{k!}
\] (1.11)

and the \textit{cluster operator} \( \hat{T} \) is defined as

\[
\hat{T} = \hat{T}_1 + \hat{T}_2 + \cdots + \hat{T}_n,
\] (1.12)

where \( n \) is the number of electrons in the molecule. The operators have the effect of replacing occupied spin-orbitals in \( \Phi_0 \) with virtual spin-orbitals. \( \hat{T}_1 \) performs all singly excited substitutions; \( \hat{T}_2 \) performs all doubly excited substitutions, and so on until all \( n \) electrons have been promoted from filled to virtual spin-orbitals. The effect of the exponential operator on the HF wave function is to express it as a linear combination that contains \( \Phi_0 \) and all possible excitations of electrons from occupied to virtual spin-orbitals. We write

\[
\hat{T}_1\Phi_0 = \sum_{r} \sum_{a} c_r^a \Phi_r^a ,
\] (1.13)

\[
\hat{T}_2\Phi_0 = \sum_{r} \sum_{s} \sum_{a} \sum_{b} c_r^a c_s^b \Phi_r^a \Phi_s^b ,
\] (1.14)
and so on, and the aim of the method is to determine the numerical coefficients. It is useful to approximate the $\hat{T}$ operator by including only some of the terms in the cluster operator. The most important contribution is generally considered to be $\hat{T}_2$. This gives the coupled-cluster doubles (CCD) method. Since we take

$$\exp(\hat{T}) \approx \exp(\hat{T}_2) = 1 + \hat{T}_2 + \frac{1}{2!}\hat{T}_2^2 + \cdots \quad (1.15)$$

the resulting wave function contains the HF wave function together with double, quadruple, hex-tuple, etc., substitutions. The CCD method is size consistent.$^9$

The next step is to include the $\hat{T}_1$ operator and so take

$$\exp(\hat{T}) \approx \exp(\hat{T}_1 + \hat{T}_2) = 1 + (\hat{T}_1 + \hat{T}_2) + \frac{1}{2!}(\hat{T}_1 + \hat{T}_2)^2 + \cdots \quad (1.16)$$

which gives CCSD. Inclusion of $\hat{T}_3$ gives the CCSDT in an obvious notation.

**Static Correlation**

One problem with traditional CI calculations is that excitations are to the virtual orbitals; the occupied orbitals are usually determined variationally by the HF procedure whilst the virtual orbitals give poor representations of excited states. One solution to the problem is to write the ground state wave function as a linear combination of Slater determinants and then use the variational procedure to optimize both the linear expansion coefficients and the HF coefficients.$^{11}$ That is, we write

$$\Psi = \Phi_0 + \sum c_j \Phi_j \quad (1.17)$$

where $\Phi_0$ is the HF wave function and $\Phi_j$ an excited state, in the usual sense of CI calculations. This is known as multi-configuration SCF (MCSCF). In certain cases the ratio of some of the $c_j$ will be fixed by symmetry.
1.3 Potential Energy Surfaces and Geometry Optimization

A potential energy surface (PES) is an essential theme of quantum chemistry, which has both conceptual and mathematical utility. In effect, it is a plot or mathematical representation of the energy of a collection of nuclei and electrons against (or as a function of) the geometric coordinates of the nuclei – that is, it is a plot of molecular energy versus molecular geometry.\(^6\)

Its veracity hinges on the Born–Oppenheimer approximation, which posits that the wave function description of a molecule can be decomposed by its nuclear and electron parts.\(^1\) This serves to simplify the mathematics, as well as providing mathematical and physical justification to the idea that a molecule has shape. (In the absence of this decomposition is a quantum soup of nucleons and electrons.) Indeed, it is usually the electronic Hamiltonian that is solved, to which the trivial calculation of inter-nuclear repulsion term is added. Then, the energy (and the other properties) of a molecule is a function of the electron coordinates, but depends only parametrically on the nuclear coordinates. A calculated PES is normally the set of points representing the geometries, and the corresponding energies, of a collection of atomic nuclei.

The characterization of a stationary point on a PES is a geometry optimization. The stationary point of interest might be a minimum, a transition state, or, occasionally, a higher-order saddle point. Locating a minimum is often called an energy minimization, and locating a transition state is often referred to specifically as a transition state optimization. Geometry optimizations are done by starting with an input structure that is believed to resemble the desired stationary point and submitting this plausible structure to a computer algorithm that systematically changes the geometry until it has found a stationary point. The curvature of the PES at the stationary point, \(i.e.\) the second derivatives of energy with respect to the geometric parameters, may then be determined to characterize the structure as a minimum or as some kind of saddle point. The most widely-used algorithms for geometry optimization use the first and second derivatives of the energy with respect to the geometric parameters.

Let \(q\) represent a structural parameter, such as a bond length (the lone geometric degree of freedom in a diatomic molecule). If we know \((dE/dq)_i\), the slope or gradient of the PES at the point of the initial structure, \((d^2E/dq^2)\), the curvature of the PES and \(q_i\), the initial geometry, we can calculate \(q_o\), the optimized geometry. The second derivative of potential energy with respect
to geometric displacement is the force constant for motion along that geometric coordinate; this is an important concept in connection with calculating vibrational spectra.

Most modern geometry optimization methods do depend on calculating the first and second derivatives of the energy at the point on the PES corresponding to the input structure. Since the PES is not strictly quadratic, the second derivatives vary from point to point and are updated as the optimization proceeds.

Optimizations are commonly done using Cartesian coordinates \( x, y, z \). Consider the optimization of a triatomic molecule in a Cartesian coordinate system. Each of the three atoms has an \( x, y \) and \( z \) coordinate, giving nine geometric parameters, \( q_1, q_2, \ldots, q_9 \); the PES would be a nine-dimensional hypersurface on a 10D graph. We need the first and second derivatives of \( E \) with respect to each of the nine \( q \)'s, and these derivatives are manipulated as matrices. The first-derivative matrix, the gradient matrix, for the input structure can be written as a column matrix

\[
\mathbf{g}_i = \begin{pmatrix}
(\partial E/\partial q_1)_i \\
(\partial E/\partial q_2)_i \\
\vdots \\
(\partial E/\partial q_9)_i
\end{pmatrix}
\] (1.18)

and the second-derivative matrix, the force constant matrix, is

\[
\mathbf{W} = \begin{pmatrix}
\partial^2 E/\partial q_1 \partial q_2 & \partial^2 E/\partial q_1 \partial q_2 & \cdots & \partial^2 E/\partial q_1 \partial q_9 \\
\partial^2 E/\partial q_2 \partial q_1 & \partial^2 E/\partial q_2 \partial q_2 & \cdots & \partial^2 E/\partial q_2 \partial q_9 \\
\vdots & \vdots & \ddots & \vdots \\
\partial^2 E/\partial q_9 \partial q_1 & \partial^2 E/\partial q_9 \partial q_2 & \cdots & \partial^2 E/\partial q_9 \partial q_9
\end{pmatrix}
\] (1.19)

The force constant matrix \( \mathbf{W} \) is called the Hessian. The Hessian is particularly important, not only for geometry optimization, but also for the characterization of stationary points as minima, transition states or hilltops, and for the calculation of IR spectra. The geometry coordinate matrices
for the initial and optimized structures are

\[ q_i = \begin{pmatrix} q_{i1} \\ q_{i2} \\ \vdots \\ q_{i9} \end{pmatrix} \] (1.20)

and

\[ q_o = \begin{pmatrix} q_{o1} \\ q_{o2} \\ \vdots \\ q_{o9} \end{pmatrix} \] (1.21)

The matrix equation relating \( q_o \) to \( q_i \), for the general case, can be shown to be

\[ q_o = q_i - H^{-1} g_i. \] (1.22)

For \( N \) atoms we have \( 3N \) Cartesian coordinates; \( q_o, q_i \) and \( g_i \) are \( 3N \times 1 \) column matrices and \( W \) is a \( 3N \times 3N \) square matrix. Eq. 1.22 shows that for an efficient geometry optimization we need an initial structure (for \( q_i \)), initial gradients (for \( g_i \)) and second derivatives (for \( W \)). With an initial guess for the geometry, gradients can be calculated analytically from the derivatives of the molecular orbitals and the derivatives of certain integrals. Since the PES is not really exactly quadratic, the first step does not take us all the way to the optimized geometry (\( q_o \)). Rather, we arrive at \( q_1 \), the first calculated geometry; using this geometry a new gradient matrix and a new Hessian are calculated (the gradients are calculated analytically and the second derivatives are updated using the changes in the gradients). Using \( q_1 \) and the new gradient and Hessian matrices a new approximate geometry matrix \( q_2 \) is calculated. The process is continued until the geometry and/or the gradients have ceased to change appreciably.

Finding a transition state is a more challenging computational problem than finding a minimum, as the characteristics of the PES at the former are more complicated than at a minimum: at the transition state the surface is a maximum in one direction and a minimum in all others, rather than
simply a minimum in all directions. Nevertheless, modifications of the minimum-search algorithm enable transitions states to be located, albeit often with less ease than minima.

Once a stationary point has been found by geometry optimization, it is usually desirable to check whether it is a minimum, a transition state, or a hilltop. This is done by calculating the vibrational frequencies. Such a calculation involves finding the normal-mode frequencies; these are the simplest vibrations of the molecule, which, in combination, can be considered to result in the actual, complex vibrations that a real molecule undergoes.

A nonlinear molecule with \( N \) atoms has \( 3N - 6 \) normal modes: the motion of each atom can be described by three vectors, along the \( x \), \( y \), and \( z \) axes of a Cartesian coordinate system. After removing the three vectors describing the translational motion of the molecule as a whole (the translation of its center of mass) and the three vectors describing the rotation of the molecule (around the three principal axes needed to describe rotation for a three-dimensional object of general geometry), we are left with \( 3N - 6 \) independent vibrational motions. Arranging these in appropriate combinations gives \( 3N - 6 \) normal modes. A linear molecule has \( 3N - 5 \) normal modes.

The normal-mode frequency (only one for a diatomic, of course) is given by:

\[
\nu = \frac{1}{2\pi c} \sqrt{\frac{k}{\mu}}
\]

where \( \nu \) = vibrational “frequency,” in \( \text{cm}^{-1} \), \( c \) = velocity of light, \( k \) = force constant for the vibration, \( \mu \) = reduced mass.

The force constant \( k \) of a vibrational mode is a measure of the “stiffness” of the molecule toward the vibrational mode – the harder it is to stretch or bend the molecule in the manner of the mode, the bigger is that force constant. The fact that the frequency of a vibrational mode is related to the force constant for the mode suggests that it might be possible to calculate the normal-mode frequencies of a molecule from its force constant matrix (its Hessian). This is indeed possible: matrix diagonalization of the Hessian gives the directional characteristics, and the force constants themselves, for the vibrations.
The $9 \times 9$ Hessian for a triatomic molecule (three Cartesian coordinates for each atom) is decomposed by diagonalization into a $P$ matrix whose columns are “direction vectors” for the vibrations whose force constants are given by the $k$ matrix. Actually, columns 1, 2 and 3 of $P$ and the corresponding $k_1$, $k_2$ and $k_3$ of $k$ refer to translational motion of the molecule (motion of the whole molecule from one place to another in space); these three force constants are nearly zero. Columns 4, 5 and 6 of $P$ and the corresponding $k_4$, $k_5$ and $k_6$ of $k$ refer to rotational motion about the three principal axes of rotation, and are also nearly zero. Columns 7, 8 and 9 of $P$ and the corresponding $k_7$, $k_8$ and $k_9$ of $k$ are the direction vectors and force constants, respectively, for the normal-mode vibrations: $k_7$, $k_8$ and $k_9$ refer to vibrational modes 1, 2 and 3, while the seventh, eighth, and ninth columns of $P$ are composed of the $x$, $y$ and $z$ components of vectors for motion of the three atoms in mode 1 (column 7), mode 2 (column 8), and mode 3 (column 9). “Mass-weighting” the force constants, i.e. taking into account the effect of the masses of the atoms, gives the vibrational frequencies. The $P$ matrix is the eigenvector matrix and the $k$ matrix is the eigenvalue matrix from diagonalization of the Hessian $W$. Thus the directions of the normal-mode frequencies are the eigenvectors, and their magnitudes are the mass-weighted eigenvalues, of the Hessian.

Vibrational frequencies are calculated to obtain IR spectra, to characterize stationary points, and to obtain zero point energies. The calculation of meaningful frequencies is valid only at a
stationary point and only using the same method that was used to optimize to that stationary point.

A minimum on the PES has all the normal-mode force constants (all the eigenvalues of the Hessian) positive: for each vibrational mode there is a restoring force. For a transition state, however, one of the vibrations, that along the reaction coordinate, is different. Since a frequency calculation involves taking the square root of a force constant, and the square root of a negative number is an imaginary number, a transition state has one imaginary frequency, corresponding to the reaction coordinate. In general an \( n \)-th-order saddle point (an \( n \)-th-order hilltop) has \( n \) negative normal-mode force constants and so \( n \) imaginary frequencies, corresponding to motion from one stationary point of some kind to another.

Besides indicating the IR spectrum and providing a check on the nature of stationary points, the calculation of vibrational frequencies also provides the zero-point vibrational energy (ZPVE). The ZPVE is the energy a molecule has even at absolute zero, as a consequence of the fact that even at this temperature it still vibrates. The ZPVE of a species is usually not small compared to activation energies or reaction energies, but ZPVEs tend to cancel out when these energies are calculated (by subtraction), since for a given reaction the ZPVE of the reactant, transition state and product tend to be roughly the same. However, for accurate work the ZPVE is added to the “total” (electronic and nuclear repulsion) energies of species and the ZPVE-corrected energies then compared.

The Hessian that results from a geometry optimization is built up in steps from one geometry to the next, approximating second derivatives from the changes in gradients. This Hessian is not accurate enough for the calculation of frequencies and ZPVEs. The calculation of an accurate Hessian for a stationary point can be done analytically or numerically. Accurate numerical evaluation approximates the second derivative, but instead of \( \Delta (\partial V/\partial q) \) and \( \Delta q \) being taken from optimization iteration steps, they are obtained by changing the position of each atom of the optimized structure slightly (\( \Delta q = \text{about } 0.01 \text{ Å} \)) and calculating analytically the change in the gradient at each geometry; subtraction gives \( \Delta (\partial V/\partial q) \). This can be done by the method of finite differences.
Symmetry

Symmetry considerations are essential in the theory of molecular electronic (UV) spectroscopy and sometimes in analyzing in detail molecular wave functions, but the reasons can also be more pragmatic. A calculation run on a molecule whose input structure has the exact symmetry that the molecule should have will tend to be faster and will yield a “better” geometry than one run on an approximate structure, however close this may be to the exact one. Input molecular structures for a calculation are usually created with an interactive graphical program and a computer mouse: atoms are assembled into molecules much as with a model kit, or the molecule might be drawn on the computer screen. If the molecule has symmetry (if it is not is not $C_1$) this can be imposed by optimizing the geometry with molecular mechanics. Now consider water: we would of course normally input the H$_2$O molecule with its exact equilibrium $C_{2v}$ symmetry, but we could also alter the input structure slightly making the symmetry $C_s$ (three atoms must lie in a plane). The $C_{2v}$ structure has two degrees of freedom: a bond length (the two bonds are the same length) and a bond angle. The $C_s$ structure has three degrees of freedom: two bond lengths and a bond angle. The optimization algorithm has more variables to cope with in the case of the lower-symmetry structure.

1.4 Fundamental Frequencies

For a polyatomic molecule, the expression for potential energy can be expanded in a Taylor series.\(^\text{13}\)

\[
V = V_0 + \sum_i \left( \frac{\partial V}{\partial x_i} \right)_0 + \frac{1}{2} \sum_{i,j} \left( \frac{\partial^2 V}{\partial x_i \partial x_j} \right)_0 x_i x_j + \cdots \tag{1.25}
\]

$V_0$ does not affect vibrations and at a stationary point the gradient is zero. Thus, if we truncate the expansion at second order, then the potential energy $V$ is

\[
V = \frac{1}{2} \sum_{i,j} f_{ij} x_i x_j, \quad f_{ij} = \left( \frac{\partial^2 V}{\partial x_i \partial x_j} \right)_0 \tag{1.26}
\]
To simplify the problem mass-weighted coordinates are employed.

\[ q_i = \sqrt{m_i} \cdot x_i \]  

(1.27)

such that the potential becomes

\[ V = \frac{1}{2} \sum_{i,j} k_{ij} q_i q_j \]  

(1.28)

where

\[ k_{ij} = \left( \frac{\partial^2 V}{\partial q_i \partial q_j} \right)_0 = \frac{f_{ij}}{\sqrt{m_i m_j}} \]  

(1.29)

Normal coordinates \( Q_i \) are linear combinations of the mass-weighted coordinates \( q_i \) such that the total energy is diagonal – that is to say, there are no cross-terms.

\[ V = \frac{1}{2} \sum_r \lambda_r Q_r^2 \]  

(1.30)

where \( \lambda_r = \omega_r^2 \), and \( \omega_r \) is the classical frequency of vibrations. To calculate the harmonic frequencies the Hessian (second derivative of the potential energy with respect to the coordinates) is necessary.

In vibrational perturbation theory, the zeroth order Hamiltonian is the harmonic oscillator (normal mode) Hamiltonian. Anharmonicity is appended to this as a perturbation.

\[ \hat{H}_{\text{VPT2}} = \hat{H}_{\text{HO}} + \hat{H}_{\text{anh}} \]  

(1.31)

The vibrational energies are found from second-order perturbation theory.\(^{14}\) \( \hat{H}_{\text{anh}} \) includes force constants that are cubic and quartic such that the potential can be expressed as

\[ V = \frac{1}{2} \sum_r \lambda_r Q_r^2 + \frac{1}{6} \sum_{rst} \phi_{rst} Q_r Q_s Q_t + \frac{1}{24} \sum_{rstu} \phi_{rstu} Q_r Q_s Q_t Q_u + \sum_\alpha \beta_\alpha j_\alpha^2 \]  

(1.32)
where

\[ j_\alpha = \sum_{r<s} \zeta^\alpha_{rs} (q_r q_s - q_s q_r) . \]  

(1.33)

In the above equations \( \alpha \) identifies a rotational axis, \( B_\alpha \) the corresponding equilibrium rotational constant, and \( \zeta^\alpha_{ij} \) is a Coriolis coupling constant. Finally,

\[ \phi_{rst} = \frac{\partial^3 V}{\partial Q_r \partial Q_s \partial Q_t} , \]  

(1.34)

and similarly for the quartic constants.

The derivatives \( \phi_{rst} \) and \( \phi_{rstu} \) are found by numerical differentiation of the analytical Hessian at geometries slightly displaced from equilibrium. Typically, terms of at most three indices are included (e.g., only \( \phi_{1233} \), no \( \phi_{1234} \)). At most, one only requires Hessian matrices at \( 6N - 11 \) different points. This is referred to as a \textit{semi-quartic force field}, which is the approximation present in the present research. In those cases where all unique indices are considered, a full-quartic force field is calculated.

VPT2 is easily automated and gives results of reasonable accuracy\(^{15}\) However, it can’t handle all types of vibrations (progressions) or highly excited states. The potential energy surface is truncated at reasonably low order. It is fairly expensive, requiring a large number of Hessian calculations. It is only applicable to modes where the harmonic term dominates the shape of the potential. Additionally, one must be careful when dealing with vibrational (Fermi) resonances\(^{16}\). The possible presence of singularities due to Fermi resonances is a common problem for VPT2 analysis\(^{17}\). One possible Fermi resonance (Type I) occurs when one harmonic frequency is twice that of another frequency:

\[ \omega_i = 2\omega_j \]  

(1.35)

Another possible Fermi resonance (Type II) occurs when two distinct frequencies sum to the value of a third:

\[ \omega_i = \omega_j + \omega_k \]  

(1.36)
The reason this is important is because some terms in the VPT2 energy correction take on the form

\[ \frac{\phi^2_{ijk}}{16(\omega_i - \omega_j - \omega_k)}, \]

which would give rise to a singularity in the case of a Type 2 resonance. However, no Fermi resonances were encountered in the research presented in this dissertation, and so research efforts aimed at ameliorating this problem are not discussed.

1.5 Natural Bonding Orbitals

Although canonical Hartree–Fock orbitals are, by definition of the variational theorem, the best possible for a single-configuration wave function or electron density, there is an even “better” set of orbitals, called natural orbitals, to describe the correlated electron density.\(^{18}\) An input atomic orbital (AO) basis set is transformed via natural atomic orbitals (NAOs) and natural hybrid orbitals (NHOs) into natural bond orbitals (NBOs).

\[
\text{AO} \rightarrow \text{NAO} \rightarrow \text{NHO} \rightarrow \text{NBO} \rightarrow \text{NLMO} \rightarrow \text{CMO}
\]

This localization algorithm permits the form of the bond orbitals to be fully optimized with respect to a maximum-occupancy criterion based solely on the first-order reduced density matrix.\(^{19}\) Remarkably, the NBOs obtained in this fashion correspond to the widely used Lewis picture, in which two-center bonds and lone pairs are localized. That is, the transformation of the wave function in the AO basis via this localization algorithm produces electron densities\(^{20}\) corresponding to the one-center (“lone pairs”) and two-center (“bonds”) elements of the chemist’s Lewis structure picture. Naturally, this NBO transformation and its corresponding analysis can illuminate chemically intuitive aspects of bonding that underlie chemical phenomena. These intuitive descriptions of orbital interactions arising from the wave function permit discussions of bond orders, basicity, resonance, steric analysis, stereoselectivity, reactivity, intramolecular and intermolecular energy barriers, and other phenomena, in terms of “intuitive” electronic structure theory.\(^{21–23}\)

The NBO method was developed by Weinhold and co-authors and has rapidly become a popular method for disinterring chemical intuition from a wave function that is often difficult to describe.
in its native, canonical MO (CMO) form.\textsuperscript{18–20,24} CMOs are delocalized over the whole molecule. As such, they usually bear little resemblance to localized, 1c/2e (lone pair) or 2c/2e bonds (e.g., $\sigma$, $\pi$). By contrast, the intuitive NBO basis that supports such desirable, intuitive results are due to the computationally inexpensive mathematical transformations from $n$ atomic orbitals (AOs) in the atomic orbital basis into an equal number of NAOs, and the $n$ MOs into an equal number of NBOs.

The NBO localization protocol divides NBOs into core, bonding, anti-bonding, and “Rydberg” orbitals.\textsuperscript{24} The core and bonding orbitals describe the strictly localized Lewis structure of a molecule. The Rydberg orbitals are those “left over” as a consequence of using large numbers of basis functions associated with big basis sets. While these Rydberg orbitals may be important for fine-tuning the potential energy surface, the large majority (99.9+%, typically) of the electron density is found in the orbitals characterized by a minimal basis set, and which are associated with the chemist’s picture of electronic structure.
CHAPTER 2

PHOSGENE AT THE COMPLETE BASIS SET LIMIT OF CCSDT(Q):
MOLECULAR STRUCTURE AND ROVIBRATIONAL ANALYSIS

2.1 Abstract

The ground-state (\( \tilde{X}^1A'' \)) equilibrium geometry of phosgene is examined with coupled-cluster theory, using derivatives extrapolated to the complete basis set (CBS) limit of CCSDT(Q). The C—O and C—Cl bond lengths are predicted to be 1.1768 Å and 1.7374 Å, respectively; the Cl—C—Cl bond angle is 124.03° and the O—C—Cl bond angle is 111.93°. Anharmonic frequencies are determined with VPT2, using CCSD(T)/cc-pCVQZ cubic and quartic force-fields and a CCSDT(Q)/CBS quadratic force field: \( \nu_1 = 1832.9; \nu_2 = 570.5; \nu_3 = 301.2; \nu_4 = 576.3; \nu_5 = 849.4; \) and \( \nu_6 = 438.9 \text{ cm}^{-1} \).

2.2 Introduction

World War I saw the first large scale use of chemical weapons, including bromo ethylacetate (tear gas), chlorine, and mustard gas. These agents were largely replaced by phosgene (Cl\(_2\)CO) in 1915 because of its superior deployment properties and pronounced toxicity.\(^52\) In modern day, the use of phosgene in international conflicts is prohibited by the Geneva Protocol of 1925, but it is still utilized as an industrial reagent in the production of isocyanates, the precursors to polyurethane.\(^53\) Phosgene is produced on the commercial scale via the highly-exothermic reaction of carbon monoxide (CO) with chlorine (Cl\(_2\)). In the terrestrial atmosphere, however, it is formed from the OH-initiated oxidation of chlorinated hydrocarbons, such as CHCl\(_3\).\(^55\)

Fundamental investigations into the chemical structure and reactivity of phosgene followed from its wartime utility. The infrared (IR) spectrum of Cl\(_2\)CO was initially reported by Marton in 1925.\(^57\) This study was followed by Dadieu and Kohlrausch’s\(^58\) determination of the Raman spectrum in 1931, in which five bands ranging from 300 to 1810 cm\(^{-1}\) were detected in condensed-phase material. The planar, \( C_{2v} \)-symmetric configuration of Cl\(_2\)CO was confirmed via electron diffraction by Pauling and colleagues four years later.\(^59\) Around the same time, Kohlrausch and Pongratz provided assignments to the six normal modes of phosgene, wherein all vibrations were noted as being infrared (IR) and Raman active.\(^60\)

The first comprehensive gas-phase IR spectrum of Cl\(_2\)CO was reported by Bailey and Hale in 1938, but only five of the six predicted vibrational modes were resolved.\(^61\) Much later, in 1958, Catalano and Pitzer revealed “the existence of two almost accidentally degenerate” vibrational
modes of different symmetry centered at \( \text{ca. 575 cm}^{-1} \), which was corroborated by Overend and Evans.\(^{63}\) Kuchitsu and colleagues\(^{64}\) determined the equilibrium molecular structure of ground-state phosgene by microwave spectroscopy in 1984;\(^{65}\) a comprehensive rovibrational investigation followed in 1985,\(^{66}\) based on prior work by Schnöckel,\(^{67}\) along with Carpenter and Rimmer.\(^{68}\) Recent work has improved the quality of certain values.\(^{69}\)

The earliest efforts to characterize phosgene with theory appeared about 50 years after Marton’s infrared report: in 1977, the complete-neglect of differential overlap (CNDO) method was used to predict the Raman spectrum.\(^{70}\) Two of the reported values, 567 cm\(^{-1}\) (CCl\(_2\) symmetric stretching) and 285 cm\(^{-1}\) (CCl\(_2\) bending), showed surprising agreement with the experimental Raman spectrum (573 and 302 cm\(^{-1}\), respectively). In 1979, Oberhammer and Boggs employed the Hartree-Fock method in a fully \textit{ab initio} approach.\(^{71}\) This method, coupled with a specially-derived basis set comparable in quality to 4-31G, was used to generate optimized geometries, dipole moments, and atomic charges.

The next theoretical investigation appeared in 2000, when Dixon \textit{et al.} reported the geometries and vibrational frequencies of phosgene using the MP2/cc-pVTZ level of theory.\(^{72}\) Five years later, Choi and Baeck employed equation-of-motion coupled-cluster theory with single and double excitations (EOM-CCSD) to predict the geometric parameters and vibrational frequencies for several low-lying electronic states.\(^{73}\) Later, Bokarev and colleagues\(^{74}\) utilized coupled-cluster theory with perturbative triple excitations and a large basis set [CCSD(T)/cc-pVQZ] to achieve a predicted C—O bond length of 1.179 Å and a C—Cl bond length of 1.743 Å, which deviate by only 0.002 and 0.007 Å, respectively, from experimentally-derived parameters. A recent report by Richter and colleagues reported the vibrational transition for the carbonyl stretch, but their QCISD/cc-pVTZ prediction differs significantly from the experimental measurement.\(^{75}\) Also in 2016, Carrington and co-workers reported the lowest 250 energy levels of phosgene from a six-dimensional potential energy surface constructed from 25,000 CCSD(T)/cc-pVTZ single-point computations that were fit using a neural network method.\(^{76}\)

The current study builds upon the extant literature by providing equilibrium geometries and fundamental vibrational frequencies computed at significantly higher levels of electronic structure theory. Specifically, geometric parameters and vibrational frequencies obtained at the complete basis set (CBS) limit of coupled-cluster theory with full triple and perturbative quadruple excitations.
[CCSDT(Q)] are reported. Anharmonic corrections to the vibrational transitions are predicted using CCSD(T) and the cc-pCVQZ, core-valence basis set. Additionally, vibration-rotation interaction constants \((\alpha^B_r)\) and rotational constants \((B_e)\) are presented. Interpretation of the molecular structure, including atomic charges and chemical bonding, is facilitated with Natural Bond Orbital (NBO) analysis.

2.3 Theoretical Methods

The equilibrium geometry for phosgene was obtained at the complete basis set (CBS) limit of coupled-cluster theory with full triple and perturbative quadruple excitations: CCSDT(Q). Optimized geometric parameters were considered converged when the RMS force fell below \(5 \times 10^{-7}\) H bohr\(^{-1}\). For each step in the optimization procedure, a gradient was constructed by the finite difference of energies using 13 displacements generated by a five-point formula, as implemented in Psi4.

To furnish a CCSDT(Q)/CBS gradient, single-point energies at each of the displaced coordinates were determined at the same, CCSDT(Q)/CBS level of theory. Specifically, a series of CCSD(T), CCSDT, and CCSDT(Q) single-point computations (red-shaded cells in Figure 2.1) were performed at each displacement, using Dunning basis sets up to cc-pV6Z. The results from these computations (marked with “x”) were extrapolated to the CBS limit (grey-shaded cells), using a three-parameter exponential function for the Hartree-Fock energies, following the focal-point approach.

\[
E_X^{HF} = E_X^{CBS} + ae^{-bX}
\]

and a two-parameter power function for the correlation energies,

\[
E_X^{corr} = E_X^{corr} + bX^{-3}
\]

The blue-shaded cells in Figure 2.1 represent values that were determined by additivity. For example, the CCSDT/cc-pV5Z value was not explicitly computed, but determined by: CCSDT/cc-pV5Z = CCSD(T)/cc-pV5Z + [CCSDT/cc-pVQZ - CCSD(T)/cc-pVQZ]. Finally, an approximate “core-correction” was appended to the CCSDT(Q)/CBS value.
obtained by valence focal-point analysis. This correction was calculated from the difference between "all-electron" and "frozen-core" CCSD(T)/cc-pCVQZ computations, with the 1s-like orbitals of chlorine excluded from post Hartree-Fock computations in both cases. Relativistic effects were not included.

<table>
<thead>
<tr>
<th></th>
<th>HF</th>
<th>MP2</th>
<th>CCSD</th>
<th>CCSD(T)</th>
<th>CCSDT</th>
<th>CCSDT(Q)</th>
</tr>
</thead>
<tbody>
<tr>
<td>cc-pVDZ</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>cc-pVTZ</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>cc-pVQZ</td>
<td>x</td>
<td>x</td>
<td></td>
<td>x</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>cc-pV5Z</td>
<td>x</td>
<td>x</td>
<td></td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>cc-pV6Z</td>
<td>x</td>
<td>x</td>
<td></td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CBS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 2.1: Representative focal-point table. Red-shaded cells indicate explicitly computed values. The “x” symbols denote values used to determine the complete-basis set (CBS) results (grey-shaded cells). Blue-shaded cells indicate values determined by additivity.

Harmonic vibrational frequencies were also determined at the CCSDT(Q)/CBS level of theory via the finite difference of energy points (51 total). To account for the anharmonicity of the potential energy surface, anharmonic corrections to the harmonic frequency values were predicted using the CCSD(T) method and the cc-pCVQZ, core-valence basis set. Stringent convergence criteria were specified throughout: geometric parameters were considered optimized when the RMS force fell below $1 \times 10^{-9}$ H bohr$^{-1}$; components of the SCF density matrix were converged to $1 \times 10^{-10}$; and, CC amplitudes were converged to $1 \times 10^{-10}$. Requisite cubic and semi-diagonal quartic components of the force field were calculated by fitting analytic second derivatives at displaced nuclear coordinates along the normal modes. Second-order vibrational perturbation theory, as implemented in PyVPT2, was utilized to obtain the anharmonicity constants ($x_{r,r}$ and $x_{r,s}$): \(^{87,88}\)

\[
\begin{align*}
x_{r,r} &= \frac{1}{16} \phi_{r,r,r,r} - \frac{1}{16} \sum_x \phi_{r,r,s}^2 \frac{8 \omega_r^2 - 3 \omega_s^2}{\omega_s (4 \omega_r^2 - \omega_s^2)} \\
x_{r,s} &= \frac{1}{4} \phi_{r,r,s,s} - \frac{1}{4} \sum_x \phi_{r,r,t} \frac{\omega_t (\omega_r^2 - \omega_s^2)}{\Delta_{r,s,t}} + \Gamma_{r,s} \\
\Delta_{r,s,t} &= (\omega_r + \omega_s + \omega_t)(\omega_r + \omega_s - \omega_t)(\omega_r - \omega_s + \omega_t)(\omega_r - \omega_s - \omega_t) \\
\Gamma_{r,s} &= \left[ A_r (\zeta_r^{(a)}儒^2 + B_r (\zeta_r^{(b)}儒^2 + C_r (\zeta_r^{(c)}儒^2) \right] \left( \frac{\omega_r}{\omega_s} + \frac{\omega_s}{\omega_r} \right)
\end{align*}
\]

where the quadratic force constants ($\omega_r$) and equilibrium rotational constants ($B_r$) were obtained at the CCSDT(Q)/CBS level of theory, as described above, while all other values, including the cubic ($\phi_{r,s,t}$) and quartic ($\phi_{r,s,t}$) force constants, along with the Coriolis zeta constants ($\zeta_r^{(B)}$), were obtained at the CCSD(T)/cc-pCVQZ level of theory. Anharmonic intensities were determined using
the equations proposed by Vázquez and Stanton,\textsuperscript{16} with refinement by Barone:\textsuperscript{17}

\[
\langle \mu^\alpha \rangle_{\alpha r} = \frac{1}{\sqrt{2}} \mu^\alpha_{r} + \frac{1}{4\sqrt{2}} \sum_s \mu^\alpha_{r,s,s} - \frac{1}{8\sqrt{2}} \sum_{s,t} \left\{ \phi_{r,s,t} \mu^\alpha_{s,t} \left( \frac{1}{\omega_r + \omega_s + \omega_t} - \frac{1}{\omega_r - \omega_s - \omega_t} \right) + \frac{2}{\omega_t} \phi_{r,s,t \mu^\alpha_{r,t}} \right\}
\]

\[
+ \frac{1}{16\sqrt{2}} \sum_{s,t,u} \left\{ \phi_{r,t,u} \phi_{s,t,u} \mu^\alpha_{s,t} \left[ \frac{4 \omega_s \omega_r \omega_t (1 - \delta_{r,s}) (1 - \delta_{t,r}) (1 - \delta_{s,u})}{\omega_r^2 - \omega_t^2} + \frac{4 \omega_s (1 - \delta_{r,s}) (1 - \delta_{t,r}) (1 - \delta_{s,u})}{\omega_t (\omega_r^2 - \omega_t^2)} \right] \right\}
\]

\[
+ \phi_{r,s,t} \phi_{u,t} \mu^\alpha_{s,t} \left[ \frac{\delta_{r,s}}{\omega_r + \omega_s + \omega_t} \left( 1 + \frac{2}{9} \delta_{r,t} \delta_{r,u} \right) - \frac{4 \omega_s (1 - \delta_{r,s}) (1 - \delta_{t,r}) (1 - \delta_{s,u})}{\omega_t (\omega_r^2 - \omega_t^2)} \right] - \frac{1}{8\sqrt{2}} \phi_{r,s,t} \mu^\alpha_{r,s} \left( \frac{1}{\omega_r + \omega_s - \omega_t} \right)
\]

\[
+ \frac{1}{2\sqrt{2}} \sum_{s,t} \mu^\alpha_{r,s} \left( \sum_{x,y,z} B^s_{r} \zeta_{r,s} \zeta_{r,t} \left[ \sqrt{\omega_r \omega_t} \left( \frac{1}{\omega_r + \omega_t} + \frac{1 - \delta_{r,t}}{\omega_r - \omega_t} \right) \right] \right)
\]

\[
- \frac{\omega_s}{\sqrt{\omega_r \omega_t}} \left( \frac{1}{\omega_r + \omega_t} - \frac{1 - \delta_{r,t}}{\omega_r - \omega_t} \right)
\]

where $\mu_r$, $\mu_{r,s}$, and $\mu_{r,s,t}$ are dipole derivatives computed with CCSD(T)/cc-pCVQZ. In the preceding equation, the first part (1) is the harmonic component of the intensity, while the remaining parts account for electrical (2), mixed (3), and mechanical anharmonicity; the mechanical contribution is further divided into cubic (4), quartic (5), and Coriolis (6) components.\textsuperscript{16} Note that $r, s, t, u$ run over the normal modes, and $\alpha$ is the Cartesian component.

The dominant CI coefficient from a complete active space self-consistent field (CASSCF) computation, for an active space of 14 electrons distributed in 14 orbitals, indicates that 91% of the ground-state wavefunction of phosgene can be described by a single-reference determinant. This result validates the single-reference methods employed in the present research.

MOLPRO 2010.1 was used to perform the CCSD(T) single-point and CASSCF computations,\textsuperscript{89} while the CFour 2.0 suite of programs was utilized for analytic CCSD(T) second-derivatives and higher order coupled-cluster computations (via the NCC module of Matthews \textit{et al.}).\textsuperscript{90,91} The INTDER program was employed for total energy distribution (TED) analysis.\textsuperscript{92}
The electronic structure of phosgene was further probed with Natural Bond Orbital (NBO) techniques, which includes the transformation of nonorthogonal atomic orbitals to the complete and orthonormal sets of natural atomic orbitals, hybrid orbitals, and bond orbitals. These localized basis sets describe electron densities and other properties by the smallest number of filled orbitals in the most rapidly convergent fashion.\textsuperscript{20} NBO computations were performed with the NBO 6.0 package,\textsuperscript{93} as interfaced with Q-Chem 4.4.\textsuperscript{94} For these NBO computations, the B3LYP functional and cc-pVTZ basis set were utilized. Note that NBOs are relatively insensitive to increasingly rigorous treatment of electron correlation, and that B3LYP provides satisfactory values for a quantitative assessment of phosgene.

### 2.4 Results and Discussion

#### Molecular Structure

Optimized geometric parameters from the present study are listed in Table 2.1, with comparisons to prior theoretical reports and experimentally-derived equilibrium parameters. Excellent agreement between the computed CCSDT(Q)/CBS values and the experimental parameters is observed, with deviations in predicted bond lengths ($r_{C-O}$ and $r_{C-Cl}$) within 0.0009 Å; bond angles deviate by 0.02°. Table 2.1 clearly reveals that higher levels of theory – that is, more rigorous treatments of electron correlation – along with larger one-particle basis sets yield more accurate results, as expected.

<table>
<thead>
<tr>
<th>method</th>
<th>$r_{C-O}$ (Å)</th>
<th>$r_{C-Cl}$ (Å)</th>
<th>$\angle_{Cl-C-Cl}$ (°)</th>
<th>$\angle_{O-C-Cl}$ (°)</th>
<th>reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>HF/4-31G</td>
<td>1.180</td>
<td>1.754</td>
<td>112.0</td>
<td>124.0</td>
<td>[71]</td>
</tr>
<tr>
<td>MP2/cc-pVTZ</td>
<td>1.195</td>
<td>1.741</td>
<td>111.6</td>
<td>124.2</td>
<td>[72]</td>
</tr>
<tr>
<td>EOM-CCSD/6-31+G</td>
<td>1.190</td>
<td>1.744</td>
<td>113.0</td>
<td>123.5</td>
<td>[73]</td>
</tr>
<tr>
<td>CCSD(T)/cc-pVQZ</td>
<td>1.179</td>
<td>1.743</td>
<td>111.9</td>
<td>124.1</td>
<td>[74]</td>
</tr>
<tr>
<td>CCSDT(Q)/CBS</td>
<td>1.1768</td>
<td>1.7374</td>
<td>111.93</td>
<td>124.03</td>
<td>present study</td>
</tr>
<tr>
<td>experiment ($r_e$)</td>
<td>1.1766(22)</td>
<td>1.7365(12)</td>
<td>111.91(24)</td>
<td>124.05(24)</td>
<td>[66]</td>
</tr>
</tbody>
</table>

Notably, the carbonyl bond length appears to be particularly sensitive to dynamic electron correlation, while the C—Cl bond is most affected by the inclusion of core electrons in post Hartree-Fock computations. MP2 theory predicts a C—O bond length of 1.195 Å,\textsuperscript{72} which is 0.02 Å too long.
The C—O bond length is reduced to 1.190 Å with CCSD, 1.179 Å with CCSD(T), and finally to 1.1768 Å with CCSDT(Q), which is only 0.0002 Å longer than the experimental value. Conversely, the C—Cl bond is only overestimated by 0.005 Å with MP2 theory, remarkably close to experiment. The deviation from the experimental value is little changed at the CCSD and CCSD(T) levels of theory (0.008 and 0.007 Å, respectively), where valence basis sets are employed. Core electrons are included in the present work using an approximate “core-correction” to the valence focal-point results, yielding a deviation of only 0.0009 Å from experiment. Indeed, geometric parameters containing second-row atoms, and chlorine in particular, have been previously found to exhibit high sensitivity to core electrons.

Rovibrational Analysis

The predicted fundamental frequencies of Cl₂CO are tabulated in Table 2.2, with comparisons to prior gas-phase experimental reports. Anharmonic corrections (δν) to the CCSDT(Q)/CBS harmonic vibrational frequencies were determined with VPT2 (δν = 2xᵣ,r + 1/2 ∑ᵣ≠s xᵣ,s), using the anharmonicity constants listed in Table 2.4 (also see Theoretical Methods). To aid in the interpretation of the six fundamental vibrational frequencies, a total energy distribution (TED) analysis was performed; those results are given in Table 2.5.

Table 2.2: Fundamental vibrational frequencies (cm⁻¹) for phosgene (Cl₂CO, C₂v). Anharmonic frequencies (ν) are given by ν = ω + δν, where ω is a harmonic frequency and δν is the VPT2 anharmonic correction (also see Theoretical Methods). Anharmonic intensities are shown in km mol⁻¹, and are normalized to 100. a = stretching, s = symmetric, a = anti-symmetric, def. = deformation, o-o-p. = out-of-plane. b Schnöckel (ref. 67, 1975); c Tchana et al., 3⁵Cl₂CO (ref. 69, 2015); d Overend and Evans (ref. 63, 1959); e Nielsen et al. (ref. 96, 1952). \(^{7}G₀ = -6.0\) cm⁻¹.

<table>
<thead>
<tr>
<th>Mode</th>
<th>Sym.</th>
<th>Type</th>
<th>Present study</th>
<th>IR (g)⁵</th>
<th>IR (g)⁶</th>
<th>IR (g)⁷</th>
<th>IR (g)⁸</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>ω  δν  ν</td>
<td>ν</td>
<td>Int.</td>
<td>ν</td>
<td>Int.</td>
</tr>
<tr>
<td>ν₁</td>
<td>a₁</td>
<td>CO</td>
<td>1841.6 −8.7</td>
<td>1832.9</td>
<td>(67)</td>
<td>1830 (vs)</td>
<td>1828.20</td>
</tr>
<tr>
<td>ν₂</td>
<td>a₁</td>
<td>CCl₂</td>
<td>578.2 −7.7</td>
<td>570.5</td>
<td>(4)</td>
<td>574 (m)</td>
<td>−</td>
</tr>
<tr>
<td>ν₃</td>
<td>a₁</td>
<td>CCl₂</td>
<td>303.6 −2.4</td>
<td>301.2</td>
<td>(0)</td>
<td>303 (w)</td>
<td>−</td>
</tr>
<tr>
<td>ν₄</td>
<td>b₁</td>
<td>o-o-p.</td>
<td>582.9 −6.6</td>
<td>576.3</td>
<td>(1)</td>
<td>586 (m)</td>
<td>−</td>
</tr>
<tr>
<td>ν₅</td>
<td>b₂</td>
<td>CCl₂</td>
<td>867.1 −17.7</td>
<td>849.4</td>
<td>(100)</td>
<td>849 (s)</td>
<td>851.01</td>
</tr>
<tr>
<td>ν₆</td>
<td>b₂</td>
<td>CO</td>
<td>443.6 −4.7</td>
<td>438.9</td>
<td>(0)</td>
<td>445 (m)</td>
<td>−</td>
</tr>
<tr>
<td>ZPVE</td>
<td></td>
<td></td>
<td>2308.5</td>
<td>2298.0</td>
<td>¹</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The 1975 study by Schnöckel provides the most recent set of infrared transitions for phosgene with natural isotope abundance; however, Tchana et al. recently suggested improved values for
Table 2.3: Components of the anharmonic intensities (10^{-3} Debye, see Theoretical Methods).

<table>
<thead>
<tr>
<th></th>
<th>(\nu_1)</th>
<th>(\nu_2)</th>
<th>(\nu_3)</th>
<th>(\nu_4)</th>
<th>(\nu_5)</th>
<th>(\nu_6)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Harmonic</td>
<td>254.4</td>
<td>107.5</td>
<td>-12.0</td>
<td>59.1</td>
<td>453.5</td>
<td>6.6</td>
</tr>
<tr>
<td>Electrical</td>
<td>-1.7</td>
<td>-0.8</td>
<td>0.1</td>
<td>-0.7</td>
<td>-1.0</td>
<td>0.1</td>
</tr>
<tr>
<td>Mixed</td>
<td>2.3</td>
<td>-0.5</td>
<td>-0.4</td>
<td>-0.2</td>
<td>-1.2</td>
<td>-0.5</td>
</tr>
<tr>
<td>Mechanical, Cubic</td>
<td>-23.5</td>
<td>1.4</td>
<td>-1.7</td>
<td>-0.1</td>
<td>-1.4</td>
<td>10.4</td>
</tr>
<tr>
<td>Mechanical, Quartic</td>
<td>0.2</td>
<td>1.0</td>
<td>-1.3</td>
<td>-0.4</td>
<td>1.2</td>
<td>7.9</td>
</tr>
<tr>
<td>Mechanical, Coriolis</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
</tbody>
</table>

the brightest modes, \(\nu_1\) and \(\nu_5\), of \(^{35}\)Cl\(_2\)CO.\(^{69}\) Compared to the values reported by Schnöckel, our predicted transitions differ by 4 cm\(^{-1}\), on average. The largest deviations are observed for \(\nu_4\) (9.7 cm\(^{-1}\)) and \(\nu_6\) (6.1 cm\(^{-1}\)), which correspond to out-of-plane deformation and C–O deformation, respectively. Note that the transition origin for \(\nu_6\) was reported to be lower in earlier reports by Nielsen \textit{et al.} and Overend and Evans.\(^{63,96}\) These authors suggest a value of 440 cm\(^{-1}\), which is excellent agreement with our computed value of 439 cm\(^{-1}\) and the value reported by Carrington and co-workers (441 cm\(^{-1}\)) using a six-dimensional PES.\(^{76}\) From the standpoint of experiment, the difficulty in resolving \(\nu_6\) may follow from its low intensity – the harmonic component of the intensity is predicted to be the smallest among the fundamental modes (see Table 2.3).

Like \(\nu_6\), Overend and Evans also reported a lower value for \(\nu_4\) (580 cm\(^{-1}\))\(^{63}\) that is closer to our computed transition of 576 cm\(^{-1}\), and even closer to the value reported by Carrington and co-workers (581 cm\(^{-1}\)).\(^{76}\) These comparisons suggest that further experiments are needed to definitively identify the fundamental transitions for \(\nu_4\) and \(\nu_6\). Gratifyingly, the brightest modes (\(\nu_1\) and \(\nu_5\)) deviate by 1.7 cm\(^{-1}\) on average from Schnöckel, giving credence to the “hybrid” approach utilized here, where high-quality cubic and quartic force-fields [CCSD(T)/cc-pCVQZ] are combined with a very high-quality quadratic force-field at the CBS limit [CCSDT(Q)/CBS].

Table 2.4: Vibrational anharmonicity constants (\(x_{r,s}, \text{cm}^{-1}\)).

<table>
<thead>
<tr>
<th>(x_{11})</th>
<th>(x_{21})</th>
<th>(x_{31})</th>
<th>(x_{41})</th>
<th>(x_{42})</th>
<th>(x_{43})</th>
<th>(x_{51})</th>
<th>(x_{52})</th>
<th>(x_{53})</th>
<th>(x_{54})</th>
<th>(x_{61})</th>
<th>(x_{62})</th>
<th>(x_{63})</th>
<th>(x_{64})</th>
<th>(x_{65})</th>
<th>(x_{66})</th>
</tr>
</thead>
<tbody>
<tr>
<td>-12.85</td>
<td>0.42</td>
<td>0.04</td>
<td>-12.85</td>
<td>-1.51</td>
<td>-1.76</td>
<td>-0.06</td>
<td>-6.67</td>
<td>-3.17</td>
<td>-3.73</td>
<td>-6.67</td>
<td>-0.61</td>
<td>40.5</td>
<td>0.23</td>
<td>-0.14</td>
<td></td>
</tr>
<tr>
<td>4.02</td>
<td>-1.51</td>
<td>-1.76</td>
<td>4.02</td>
<td>-0.06</td>
<td>-6.67</td>
<td>-3.17</td>
<td>-3.73</td>
<td>-6.67</td>
<td>-0.61</td>
<td>40.5</td>
<td>0.23</td>
<td>-0.14</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

TED analysis (Table 2.5) indicates that \(\nu_1\) (CO stretch) may be described by a single coordi-
nate \((S_{1,2})\). Mode \(\nu_2\), the symmetric CCl\(_2\) stretching band, is predominantly described by C–Cl stretching \((S_{1,3}, S_{1,4})\), as expected, with a minor (10\%) contribution from Cl–C–O bending \((B_{3,1,2}, B_{4,1,2})\). Like \(\nu_1\), mode \(\nu_4\) can be completely described by one coordinate \((T_{4,1,2,3})\), but \(\nu_3\), \(\nu_5\), and \(\nu_6\) are more complex and include contributions from stretching \((S_{1,4, S_{1,3}})\) and bending \((B_{4,1,2}, B_{3,1,2})\).

Table 2.5: Total Energy Distribution (TED) analysis showing relative contributions from the indicated coordinates. \(^a\)S = stretch, B = bend, T = torsion; also see caption of Table 2.2.

<table>
<thead>
<tr>
<th>Mode</th>
<th>Description</th>
<th>TED (% Coordinate)(^a)</th>
<th>Atom Labels</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\nu_1)</td>
<td>CO str.</td>
<td>(S_{1,2}(98))</td>
<td>(\text{O}_2)</td>
</tr>
<tr>
<td>(\nu_2)</td>
<td>CCl(_2) s-str.</td>
<td>(S_{1,3}(45) + S_{1,4}(45) + B_{3,1,2}(5) + B_{4,1,2}(5))</td>
<td>(\text{Cl}_3)</td>
</tr>
<tr>
<td>(\nu_3)</td>
<td>CCl(_2) def.</td>
<td>(B_{3,1,2}(45) + B_{4,1,2}(45) - S_{1,3}(5) - S_{1,4}(5))</td>
<td>(\text{Cl}_4)</td>
</tr>
<tr>
<td>(\nu_4)</td>
<td>o-o-p. def.</td>
<td>(T_{4,1,2,3}(100))</td>
<td>(\text{C}_1)</td>
</tr>
<tr>
<td>(\nu_5)</td>
<td>CCl(_2) a-str.</td>
<td>(B_{1,1,2}(31) - B_{3,1,2}(31) + S_{1,4}(19) - S_{1,3}(19))</td>
<td></td>
</tr>
<tr>
<td>(\nu_6)</td>
<td>CO def.</td>
<td>(S_{1,4}(31) - S_{1,3}(31) + B_{3,1,2}(19) - B_{4,1,2}(19))</td>
<td></td>
</tr>
</tbody>
</table>

The predicted equilibrium rotational constants for phosgene \((A_e, B_e, C_e)\) are shown in Table 3.4 along with zero-point corrected values \((A_0, B_0, C_0)\). These terms are related by the expression:

\[
B_0 = B_e - \frac{1}{2} \sum_r \alpha_r^B
\]  

(2.7)

where \(\alpha_r^B\) are vibration-rotation interaction constants (Table 3.5) describing the coupling of rotations about the principal axis corresponding to \(B\) with the normal mode \(r\). Prior work has demonstrated the high-accuracy of \(\alpha_r^B\) values predicted at the CBS limit of coupled-cluster theory.\(^{98}\) The vibrationally-averaged rotational constants \(A_0, B_0,\) and \(C_0\) are lower than their equilibrium counterparts by 0.3, 0.4, and 0.5\%, respectively. While these predicted values are for phosgene with natural isotope abundance, they show good agreement with experimental rotational constants for \(^{35}\text{Cl}_2\text{CO}\): \(A = 0.2641426020(450), B = 0.1159121950(170),\) and \(C = 0.8046257960(140)\) cm\(^{-1}\).\(^{69}\)

Table 2.6: Equilibrium rotational constants \((B_e)\) for natural abundance Cl\(_2\)CO determined at the CCSDT(Q)/CBS level of theory. Vibrationally-averaged rotational constants \((B_0)\) are shown using corrections obtained from VPT2 analysis using CCSD(T)/cc-pCVQZ. All values are in cm\(^{-1}\).

|  | \(B_e\) | \(B_0\) | \(|B_e - B_0|\) |
|---|---|---|---|
| \(A\) | 0.26421 | 0.26334 | 0.00087 |
| \(B\) | 0.11470 | 0.11424 | 0.00046 |
| \(C\) | 0.07998 | 0.07958 | 0.00040 |
Table 2.7: Vibration-rotation interaction constants ($\alpha^B_r$, in $10^{-4}$ cm$^{-1}$) for Cl$_2$CO computed at the CCSD(T)/cc-pCVQZ level of theory.

<table>
<thead>
<tr>
<th>Mode ($r$)</th>
<th>$A$</th>
<th>$B$</th>
<th>$C$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>8.136</td>
<td>1.063</td>
<td>0.706</td>
</tr>
<tr>
<td>2</td>
<td>31.961</td>
<td>2.121</td>
<td>3.952</td>
</tr>
<tr>
<td>3</td>
<td>-7.104</td>
<td>2.106</td>
<td>1.01</td>
</tr>
<tr>
<td>4</td>
<td>-31.668</td>
<td>-0.415</td>
<td>1.742</td>
</tr>
<tr>
<td>5</td>
<td>11.946</td>
<td>3.506</td>
<td>2.948</td>
</tr>
<tr>
<td>6</td>
<td>4.209</td>
<td>-0.352</td>
<td>-1.144</td>
</tr>
</tbody>
</table>

**Bonding Analysis**

From Natural Bond Orbital (NBO) computations, the central carbon atom of phosgene is predicted to form bonds with oxygen via $sp^{1.63}$ hybridized orbitals, akin to $sp^2$ but with slightly higher $s$ character. The C—O $\sigma$ bond forms between these orbitals and $sp^{1.25}$ hybridized orbitals on oxygen; a $\pi$ bond forms between the unhybridized carbon and oxygen $2p$ orbitals that are perpendicular to the molecular plane:

$$C=O : \sigma_{C-O} + \pi_{C-O} = [0.579(sp^{1.63})C + 0.816(sp^{1.25})O] + [0.586(p)C + 0.811(p)O]$$

The oxygen atom features two hybridized orbitals that contain non-bonded electron pairs: one NBO in the molecular plane that is $s$ rich ($n^\sigma_O$), and a pure $p$ NBO that is unhybridized and out of the molecular plane ($n^\pi_O$). The “angularly rounded” $s$-rich orbital ($n^\sigma_O$) contributes to a larger O—C—Cl bond angle than the idealized 120° angle for trigonal-planar compounds; the $n^\pi_O$ orbital has little effect on this parameter. Overall, the C—O bond is predicted to possess a natural bond order of 2.00 (double bond) that is 63% covalent and 37% ionic.

Natural resonance theory (NRT) suggests that the major resonance contributor (70%) pertains to a structure featuring the expected C—O double bond, while minor resonance contributors include a triple bond ($2 \times 7\% = 14\%$) and a single bond ($2 \times 7\% = 14\%$), see Figure 2.2.

The C—Cl $\sigma$ bonds are predicted to form between $sp^{2.22}$-hybridized carbon orbitals and $sp^{5.90}$-hybridized chlorine orbitals:

$$C-Cl : \sigma_{C-Cl} = 0.669(sp^{2.22})C + 0.735(sp^{5.90})Cl$$
The seemingly peculiar hybridization of the chlorine atom derives from its rich $p$ character (86%; F. Weinhold, University of Wisconsin-Madison, personal communication, 2017). As a result, the angular directionality of the $\sigma$ bond is limited and reduces the Cl–C–Cl bond angle below the idealized value of 120°. The higher $p$ character on carbon (2.22 vs. 2) is consistent with Bent’s rule, which stipulates that more electronegative ligands favor higher $p$-character hybrids at the central atom. The C–Cl bond has a natural bond order of 0.99, and is 83% covalent (17% ionic). Natural population analysis predicts that carbon has 3.47 valence electrons, while oxygen and chlorine have 6.42 and 6.99, respectively. These values indicate that the carbon atom is slightly electron deficient, while the more electronegative oxygen atom is electron-rich; chlorine has the expected seven valence electrons.

The molecular orbital composition is consistent with the aforementioned bonding picture. Figure 2.3 indicates that for phosgene the highest occupied molecular orbital (HOMO) and HOMO–1 are characterized almost exclusively by the non-bonding electrons residing in the $2p_y$ and $3p_y$ atomic orbitals of the oxygen and chlorine atoms. The lowest unoccupied molecular orbital (LUMO) features a large anti-bonding orbital centered over the carbon atom, with contributions from oxygen. Phosgene is considered an especially electrophilic acid chloride, and the electrophilicity of the carbon atom, in particular, serves as a sink for nucleophilic attack. The LUMO+1 predominantly features the anti-bonding C–Cl molecular orbital.

### 2.5 Conclusion

In this work, phosgene is examined with high-level *ab initio* methods. Geometry optimization using first-derivatives extrapolated to the complete basis set (CBS) limit of CSDT(Q) yields
Figure 2.3: Highest occupied molecular orbitals (HOMO−1, HOMO) and lowest unoccupied molecular orbitals (LUMO, LUMO+1) of phosgene, computed using the B3LYP/cc-pVTZ method.

an equilibrium geometry (C—O = 1.1768 Å, C—Cl = 1.7373 Å, and ∠O-C-Cl = 124.03°) that is in excellent agreement with the experimentally-derived values of Kuchitsu and coworkers, with an average deviation in bond length of 0.0005 Å from experiment. Fundamental frequencies are predicted using a “hybrid” approach that combines a CCSDT(Q)/CBS quadratic force field with CCSD(T)/cc-pCVQZ cubic and quartic force-fields to reduce the overall computational cost and leverage the rapid convergence of higher-order force constants with regard to level of theory and size of basis set. The predicted transitions display excellent agreement with the experimental values of Schnöckel, demonstrating the accuracy of this approach; however, further experimental examination of ν₄ and ν₆ is suggested. NBO computations show that the central carbon atom is electron deficient, with a natural valence of less than 3.5, which contributes to its activity as an electrophile. The dominant resonance structure (70%) features the expected CO double bond and CCl single bonds; the two minor resonance contributors (14% each) include a CO single bond and CO triple bond, respectively.
CHAPTER 3

THIOFORMALDEHYDE S-SULFIDE, SULFUR ANALOG OF THE CRIEGEE INTERMEDIATE: STRUCTURES, ENERGETICS AND ROVIBRATIONAL ANALYSIS

3.1 Abstract

The ephemeral Criegee intermediate, first postulated over 70 years ago, has only recently been isolated in the gas phase. The sulfur analog of this canonical zwitterion, thioformaldehyde S-sulfide, has eluded similar analysis; however, argon matrix isolation has been achieved (Angew. Chem. 2001, 40, 393-396). Here, thioformaldehyde S-sulfide and its valence isomer dithiirane are examined with high-level coupled-cluster methods, including the minimum energy pathway for interconversion. Relative enthalpies, calculated from extrapolated energies at the complete basis set limit of the full CCSDTQ method, are reported. Isomerization from thioformaldehyde S-sulfide to the lower-lying dithiirane (-7.2 kcal mol$^{-1}$) is predicted to include a 27.0 kcal mol$^{-1}$ barrier. Harmonic and anharmonic vibrational frequencies are also predicted using second-order vibrational perturbation theory (VPT2). These results should aid in future gas-phase identification.

3.2 Introduction

Rudolf Criegee’s proposed mechanism of olefin ozonolysis includes the formation of a zwitterionic, carbonyl O-oxide (CH$_2$OO), often termed the “Criegee intermediate” (e.g., 1b in Figure 1).$^{25}$ For over half a century this transient species evaded gas-phase isolation.$^{26}$ Then, in 2013, Huang, Witek, and Lee reported its infrared signature by generating CH$_2$OO from the reaction of O$_2$ and CH$_2$I – the latter being the photolysis product of CH$_2$I$_2$.$^{28}$ Early attempts to form this intermediate via the cycloreversion of 1,2,4-trioxolane (1e in Figure 1), which is the ozonolysis product of ethylene (via 1a), were unsuccessful.$^{29}$ However, cycloreversion proved a successful strategy for studying the sulfur analog, thioformaldehyde S-sulfide (CH$_2$SS; 2b in Figure 2).$^{30}$ Indeed, Maier and co-workers$^{31}$ showed via matrix isolation that the thermal fragmentation of 1,2,4-trithiolane (2a), a stable substance that is common in nature,$^{32-34}$ yields CH$_2$SS (2b) and the companion product

![Figure 3.1: Mechanism of ozonolysis as proposed by Criegee$^{27}$, including formation of the carbonyl O-oxide intermediate, 1b.](image-url)
dithiirane (H₂CS₂; 2c in Figure 2) in a ratio of 1:2.5. Complete conversion to H₂CS₂ (2c) was achieved photochemically, through irradiation of the product mixture with visible light (λ > 570 nm). A photoequilibrium (CH₂SS ⇌ H₂CS₂) was achieved when ca. 500 nm light was employed.

The fate of transient thiosulfines (CR₂SS) and their valence isomers, the dithiiranes (R₂CS₂), is unclear, but it has been speculated that each isomer yields distinct products when combined with thioformaldehyde (2d) or via self-reaction. As such, there is interest in trapping these intermediates for synthetic purposes.³⁵ For reference, the lighter congener of dithiiran, dioxiran (1c), can undergo homolytic O–O cleavage to yield dioxyethylene, which subsequently decomposes to furnish simpler downstream molecules such as CO₂.³⁶ Conversely, the Criegee intermediate (1b) combines with formaldehyde (1d) to generate the canonical product, 1,2,4-trioxolane (1e).³⁷ Likely, the sulfur analogs exhibit equally rich but unknown chemistry.

In 2001, argon matrix isolation was utilized by Maier and coworkers³¹ to characterize CH₂SS (2b) and H₂CS₂ (2c) for the first time; seven fundamental vibrational transitions were identified for both structures using Fourier-transform infrared (IR) spectroscopy. This prior work is pursued in the present research through the characterization of CH₂SS (2b) and H₂CS₂ (2c) with high-level electronic structure theory. The similarity between CH₂SS (2b) and CH₂OO (1b) provides the opportunity to study intricate ozonolysis chemistry with a heavy congener that may exhibit a longer lifetime, facilitating experimental interrogation.³⁸ Notwithstanding this application, the chemistry of CH₂SS (2b) is associated with several other processes — for example, 1,2,4-trithiiolane (2a) has relevance to the sulfur cycle³⁹,⁴₀,¹⁰¹ and astrochemistry,⁴¹,⁴₂,¹⁰² underscoring interest in its cycloreversion. Dithiiranes (CR₂SS) and their valence isomers, the thiosulfines (R₂CS₂), have also been proposed as intermediates in reaction mechanisms leading to sulfur-rich compounds.⁴³–⁴⁶
To date, however, substituted forms of thiosulfine (thiocarbonyl S-sulfides) have received the most attention.\textsuperscript{47–51}

The current study reports results from high-level theory to augment the limited literature concerned with the unsubstituted forms, as well as to assist the future identification of thioformaldehyde S-sulfide (2b) and dithiirane (2c) in the gas phase. Specifically, relative electronic energies between stationary points on the isomerization pathway are reported at the complete basis set (CBS) limit of coupled-cluster theory with full quadruple excitations (CCSDTQ). Fundamental frequencies are predicted for the first time using coupled-cluster theory [CCSD(T)] and second-order vibrational-perturbation theory (VPT2). Vibration-rotation interaction constants ($\alpha B_r$) and rotational constants ($B_e$ and $B_0$) are also reported.\textsuperscript{87} Analysis of atomic charges, resonance structures, and chemical bonding is facilitated by Natural Bond Orbital (NBO) methods,\textsuperscript{93} and comparisons to prior research are made where possible.\textsuperscript{31,35}

### 3.3 Theoretical Methods

Optimized geometries for minima on the potential energy surface for the CH\textsubscript{2}SS $\rightleftharpoons$ H\textsubscript{2}CS\textsubscript{2} isomerization were obtained using coupled-cluster theory with single, double, and perturbative triple excitations [CCSD(T)].\textsuperscript{77,78,103,104} The carbon and hydrogen atoms were described with Dunning’s augmented, correlation-consistent, valence, triple-ζ (aug-cc-pVTZ) basis set.\textsuperscript{80,105} Sulfur atoms were described (with added tight $d$ functions) using the aug-cc-pV(T+$d$)Z basis set,\textsuperscript{106,107} because the aug-cc-pVTZ basis sets do not fully capture core polarization and valence-orbital correlation effects in second row atoms.\textsuperscript{108}

For all computations, a frozen core (FC) approximation was applied, such that inner shell electrons of the carbon and sulfur atoms were not correlated in the post-Hartree-Fock computations. Harmonic vibrational frequencies were obtained at the same level of theory as the geometry optimizations. A restricted Hartree-Fock (RHF) reference was employed throughout, and optimized structures were verified as local minima or transition states by vibrational analyses. The relative enthalpies at 0 K ($\Delta H_{0K}$) were determined for all species with respect to thioformaldehyde S-sulfide.
(2b) by computing relative electronic energies ($\Delta E_e$) using the focal point approach (FPA)\textsuperscript{84,109} and appending zero-point vibrational (ZPVE) energy corrections.\textsuperscript{110} Specifically, absolute electronic energies were systematically extrapolated to the complete basis set (CBS) limit of coupled-cluster theory with full quadruple excitations (CCSDTQ) using a three-parameter exponential function\textsuperscript{82} and a two-parameter power function\textsuperscript{83} for the Hartree-Fock and correlation energies, respectively. In addition to the augmented, triple-ζ Dunning basis sets already described, CBS extrapolation was achieved with quadruple- and quintuple-ζ basis sets — i.e., aug-cc-pVXZ ($X = Q, 5$) on carbon and hydrogen atoms, and aug-cc-pV$(X+d)Z$ on sulfur atoms.

The $T_1$ diagnostic was employed as a metric to gauge the multireference character of stationary points,

$$\begin{align*}
T_1 &= \sqrt{\frac{|t_{1\alpha}|^2 + |t_{1\beta}|^2}{N_{\text{corr}}}} 
\end{align*}$$

along with the largest $t_{ij}^{ab}$ amplitudes.\textsuperscript{111,112} No value above 0.044 for $T_1$ and 0.108 for $t_{ij}^{ab}$ was observed.\textsuperscript{113} The CASSCF reference coefficients for an active space of 14 electrons distributed in 14 orbitals were also considered. The dominant CI coefficients indicate that 87, 84, and 90% of the reactant, transition state, and product wavefunctions, respectively, can be described by a single-reference determinant, corroborating the validity of the methods employed in the present research.

Geometry optimizations, computed at the CCSD(T) level, were performed in CFour 2.0 using the ECC module;\textsuperscript{114} higher-order coupled-cluster computations were performed using the NCC module.\textsuperscript{115–117} All CCSD(T) computations for the focal point analysis, as well as the CASSCF computations, were performed in MOLPRO 2010.1.\textsuperscript{89} Natural bond orbital (NBO) analyses\textsuperscript{18–20,24} were performed using the NBO 6.0 package,\textsuperscript{93,118} as interfaced with Q-Chem 4.4.\textsuperscript{94} NBO results include natural resonance theory (NRT) analysis,\textsuperscript{21–23} which provide properties from a weighted average of first-order density matrices. The B3LYP functional and cc-pVTZ basis set were used for the NBO computations only.
3.4 Results

In the present research, optimized geometries, relative energies, and anharmonic frequencies for thioformaldehyde S-sulfide (2b, \( C_s \) point-group symmetry), dithiirane (2c, \( C_{2v} \)), and the corresponding cyclization transition state (\( C_1 \); see Figure 3.3) are reported. Optimized structures are compared to the earlier computational study by Fabian and Senning,\(^{35} \) which included MP2 theory and the 6-31+G** Pople basis-set. For qualitative purposes, comparisons are also made to the predicted geometries of carbonyl O-oxide reported by McCarthy, Stanton, and colleagues.\(^{119} \)

*Molecular Structures*

![Figure 3.3: Optimized geometric parameters for thioformaldehyde S-sulfide (\( C_s \)), the cyclization transition state (\( C_1 \)), and dithiirane (\( C_{2v} \)). Reported distances (\( \text{Å} \)) and angles (\( ^\circ \)) are from CCSD(T)/aug-cc-pVTZ computations, which include additional tight d functions on sulfur. Also see the Supporting Information.](image)

The structure of thioformaldehyde S-sulfide (\( \text{CH}_2\text{SS}, \ 2b \)) is predicted, by the results of the present study, to possess a S—S bond length of 1.941 Å (see Figure 3.3). Upon cyclization to dithiirane (\( \text{H}_2\text{CS}_2, \ 2c \)), this bond elongates 8% to 2.100 Å; a similar, 13% change in O—O bond distance is observed when the Criegee intermediate (1.349 Å) cyclizes to dioxirane (1.525 Å).\(^{119} \) The C—S bond length extends 10% from 1.634 Å (\( \text{CH}_2\text{SS}, \ 2b \)) to 1.801 Å (\( \text{H}_2\text{CS}_2, \ 2c \)), accompanied by a lowering of bond order and the formation of another C—S bond. The C—S bond length is more than 0.35 Å longer than the C—O bond in \( \text{CH}_2\text{OO} \) (1b).\(^{119} \)

In \( \text{CH}_2\text{SS} \) (2b), the C-S-S bond angle is 113.9° (see Figure 3.3). In the transition state, the same bond is approximately a right angle, 84.0°. Closing the three-membered ring (i.e., continuing along
the reaction path) yields dithiirane, \( \text{H}_2\text{CS}_2 \) (2c), which possesses a C-S-S angle of 54.3°. Starting from \( \text{CH}_2\text{SS} \), the dihedral angle between the H-C-H plane involving methylene hydrogen atoms relative to the C-S-S plane is 0°. The H-C-H plane rotates to yield a dihedral angle of 90° in the product; in the \( \text{C}_1 \) transition state, the same angle is 54.5°. The latter value is interesting in light of Hammond’s postulate, which states that in an exothermic reaction the geometry of the transition state may be more reactant-like than product-like. Although the S—S bond of the \( \text{C}_1 \) transition state is 69% reactant-like, the C—S bond is only 35% reactant-like — i.e., closer in length to the product. Perhaps most distinctly, the H-C-S-S dihedral angle is only 39% reactant-like. It would appear that Hammond’s postulate is not appropriate for these evanescent intermediates. Interestingly, an \textit{ab initio} molecular dynamics study by Kalinowski and colleagues suggests that isomerization of the Criegee intermediate also appears to challenge Hammond’s postulate, though along different coordinates: the C—O bond of the transition state is 96% product-like, but the O—O bond is 79% reactant-like. The H-C-O-O dihedral angle is not reported, therein.

For both \( \text{CH}_2\text{SS} \) (2b) and \( \text{H}_2\text{CS}_2 \) (2c), the present coupled-cluster theory values for the S—S bond length are shorter than prior predictions with MP2/6-31+G** by 0.035 Å and 0.004 Å respectively.\(^35\) The C—S bond distance of \( \text{CH}_2\text{SS} \) (2b) is predicted by MP2 to be 0.097 Å longer than the current results; in \( \text{H}_2\text{CS}_2 \) (2c) this MP2 overestimate is 0.065 Å. While CCSD(T) is generally the superior theory, particularly for thermochemistry,\(^122\) MP2 provides geometries for closed-shell, first- and second-row systems that are reasonably comparable to CCSD(T).\(^95,123,124\) Likely, the contraction of bonds observed here are attributable to the higher quality basis set.\(^6\) In fact, the improvement gained by adding tight \( d \) functions has been previously noted to be dramatic for triple-\( \zeta \) basis sets in sulfur-containing systems.\(^125\)

\textit{Relative Energies}

To further characterize the \( C_s \), \( C_1 \), and \( C_{2v} \) stationary points pertaining to the \( \text{CH}_2\text{SS} \rightleftharpoons \text{H}_2\text{CS}_2 \) isomerization, the energetic relationships between the three conformers were computed using coupled-cluster theory with up to full quadruple excitations, extrapolated to the complete-basis limit. The focal point extrapolations in Table 3.1 show good convergence, with respect to
enthalpy of -7.2 kcal mol\(^{-1}\) and -25.0 kcal mol\(^{-1}\), than the product of the present system (dithiirane, \(1c\)). Even so, this barrier is 8.0 kcal mol\(^{-1}\) greater than the analogous barrier to cyclization of the Criegee intermediate (\(1b\); 19.0 kcal mol\(^{-1}\)) computed by Bowman, Guo, and colleagues.\(^{126}\) Additionally, the isomerization product of the Criegee intermediate (dioxirane, \(1c\)) lies considerably lower, at -25.0 kcal mol\(^{-1}\), than the product of the present system (dithiirane, \(2c\)), which has a relative enthalpy of -7.2 kcal mol\(^{-1}\).

### Table 3.1: Focal point extrapolations of the relative energies (in kcal mol\(^{-1}\)) to the CBS limit of CCSDTQ. CBS extrapolation was achieved with double-, triple-, quadruple- and quintuple-\(\zeta\) basis sets — i.e., aug-cc-pV\(XZ\) (\(X = D, T, Q, 5\)) on carbon and hydrogen atoms, and aug-cc-pV\((X+d)Z\) on sulfur atoms; these are notated in the table as DZ, TZ, QZ, and 5Z, respectively. Harmonic zero-point vibrational energy corrections (ZPVE) were computed at CCSD(T)/aug-cc-pVTZ, with added tight \(d\) functions on sulfur.

<table>
<thead>
<tr>
<th>(\text{CH}_2\text{SS} \rightarrow \text{TS})</th>
<th>HF</th>
<th>+(\delta)MP2</th>
<th>+(\delta)CCSD</th>
<th>+(\delta)CCSD(T)</th>
<th>+(\delta)CCSDT</th>
<th>+(\delta)CCSDT(Q)</th>
<th>+(\delta)CCSDTQ</th>
<th>Net</th>
</tr>
</thead>
<tbody>
<tr>
<td>DZ</td>
<td>+29.81</td>
<td>-0.19</td>
<td>-2.45</td>
<td>-1.82</td>
<td>+0.38</td>
<td>-0.50</td>
<td>+0.31</td>
<td>+25.54</td>
</tr>
<tr>
<td>TZ</td>
<td>+30.69</td>
<td>-0.75</td>
<td>-2.07</td>
<td>-1.82</td>
<td>+0.31</td>
<td>-0.44</td>
<td>+0.31</td>
<td>+26.23</td>
</tr>
<tr>
<td>5Z</td>
<td>+31.06</td>
<td>-0.50</td>
<td>-1.69</td>
<td>-1.76</td>
<td>[+0.31]</td>
<td>-0.44</td>
<td>+0.31</td>
<td>+27.30</td>
</tr>
<tr>
<td>CBS limit</td>
<td>[+31.12]</td>
<td>[-0.57]</td>
<td>[-1.56]</td>
<td>[-1.69]</td>
<td>[+0.31]</td>
<td>-0.44</td>
<td>+0.31</td>
<td>+27.48</td>
</tr>
</tbody>
</table>

\[\Delta H_0 = \Delta E_e + \Delta ZPVE[\text{CCSD(T)/aug-cc-pVTZ}] = +27.48 + (-0.51) = +26.97 \text{ kcal mol}^{-1}\]

<table>
<thead>
<tr>
<th>(\text{CH}_2\text{SS} \rightarrow \text{H}_2\text{CS}_2)</th>
<th>HF</th>
<th>+(\delta)MP2</th>
<th>+(\delta)CCSD</th>
<th>+(\delta)CCSD(T)</th>
<th>+(\delta)CCSDT</th>
<th>+(\delta)CCSDT(Q)</th>
<th>+(\delta)CCSDTQ</th>
<th>Net</th>
</tr>
</thead>
<tbody>
<tr>
<td>DZ</td>
<td>-15.06</td>
<td>+4.27</td>
<td>-0.75</td>
<td>+2.51</td>
<td>+0.31</td>
<td>+0.50</td>
<td>-0.13</td>
<td>-8.35</td>
</tr>
<tr>
<td>TZ</td>
<td>-14.75</td>
<td>+3.01</td>
<td>-0.94</td>
<td>+2.57</td>
<td>+0.19</td>
<td>+0.69</td>
<td>-0.13</td>
<td>-9.35</td>
</tr>
<tr>
<td>QZ</td>
<td>-14.50</td>
<td>+3.45</td>
<td>-0.88</td>
<td>+2.57</td>
<td>[+0.19]</td>
<td>+0.69</td>
<td>-0.13</td>
<td>-8.60</td>
</tr>
<tr>
<td>5Z</td>
<td>-14.37</td>
<td>+3.45</td>
<td>-0.82</td>
<td>+2.57</td>
<td>[+0.19]</td>
<td>+0.69</td>
<td>-0.13</td>
<td>-8.41</td>
</tr>
<tr>
<td>CBS limit</td>
<td>[-14.32]</td>
<td>[+3.45]</td>
<td>[-0.75]</td>
<td>[+2.57]</td>
<td>[+0.19]</td>
<td>+0.69</td>
<td>-0.13</td>
<td>-8.29</td>
</tr>
</tbody>
</table>

\[\Delta H_0 = \Delta E_e + \Delta ZPVE[\text{CCSD(T)/aug-cc-pVTZ}] = -8.29 + 1.13 = -7.16 \text{ kcal mol}^{-1}\]

### Vibrational Frequencies

The fundamental vibrational frequencies of thioformaldehyde S-sulfide (\(\text{CH}_2\text{SS}; 2b\)) and dithiirane (\(\text{H}_2\text{CS}_2; 2c\)), reported in Table 3.2, were computed to aid in the detection and analysis of these molecules in the gas phase. Fundamental values were obtained by appending anharmonic corrections (\(\delta\nu\)) determined at CCSD(T)/aug-cc-pVTZ, with added tight \(d\) functions on sulfur, to the harmonic frequencies (\(\omega\)) determined at the same level of theory. Second-order vibrational perturbation theory (VPT2) was employed to yield the anharmonic corrections using analytic second
derivatives at displaced geometries along the normal coordinates.\textsuperscript{127} The theoretical frequencies correspond well with the argon matrix experiment results of Maier and co-workers.\textsuperscript{31} The average absolute deviation is 0.77\%, which is consistent with the matrix shifts typically associated with argon (0.65\%).\textsuperscript{128} To further understand the vibrational modes of CH\textsubscript{2}SS (2\textsuperscript{b}), H\textsubscript{2}CS\textsubscript{2} (2\textsuperscript{c}), and the transition state linking them (see Figure 3.3), a total energy distribution (TED) analysis was computed as advocated by Pulay and Törökk\textsuperscript{129} and implemented in the INTDER program by Allen.\textsuperscript{92} The results for CH\textsubscript{2}SS (2\textsuperscript{b}) and H\textsubscript{2}CS\textsubscript{2} (2\textsuperscript{c}) appear in Table 3.2; those for the transition state appear in Table 3.3.

The most intense vibrational frequency of thioformaldehyde S-sulfide (CH\textsubscript{2}SS, 2\textsuperscript{b}) is ν\textsubscript{8}, and consists of the methylene “wag” in and out of the C\textsubscript{s} plane. Mode ν\textsubscript{6}, the S—S stretch, is predicted to have the second largest IR absorption. The large intensity derives from the change in dipole moment induced not only by the dominant S—S vibration, but also the bending motion of the carbon and two sulfur atoms (B\textsubscript{312}; see Figure 3.3), a motion that draws the sulfur atoms towards one another and anticipates cyclization. From argon matrix IR spectroscopy, Maier and co-workers\textsuperscript{31} also showed this to be an intense band. The C—S stretch, ν\textsubscript{4}, is of medium strength.

Figure 3.4: Relative enthalpy profile (see Table 3.1) of the isomerization of thioformaldehyde S-sulfide to dithirane, and the natural charges (units $e = 1.6 \times 10^{-19}$ coulombs) of each conformer.
Table 3.2: Fundamental vibrational frequencies (cm$^{-1}$) for thioformaldehyde S-sulfide (CH$_2$SS, $C_s$) and dithiirane (H$_2$CS$_2$, $C_{2v}$). Frequencies are given by $\nu = \omega + \delta\nu$, where $\omega$ is a harmonic frequency and $\delta\nu$ is the anharmonic correction. All values were determined with CCSD(T)/aug-cc-pVTZ (with added tight $d$ functions on sulfur). Harmonic intensities are shown in km mol$^{-1}$. Total Energy Distributions (TED) show percentage contributions from the indicated coordinates. Third column: str. = stretching, sciss. = scissoring, rock. = rocking, wag. = wagging, twist. = twisting, def. = deformation. Last column: S = stretch, B = bend; T = torsion; subscripts refer to atoms as shown in Figure 3.3.  

<table>
<thead>
<tr>
<th>Mode</th>
<th>Sym</th>
<th>Description</th>
<th>ω</th>
<th>δν</th>
<th>ν</th>
<th>Int</th>
<th>ν</th>
<th>Int</th>
<th>TED</th>
</tr>
</thead>
<tbody>
<tr>
<td>CH$_2$SS ($C_s$)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\nu_1$</td>
<td>$a'$</td>
<td>CH$_2$ str.</td>
<td>3262.0</td>
<td>-146.8</td>
<td>3115.1</td>
<td>(1.9)</td>
<td>3126.3</td>
<td>(vw)</td>
<td>$S_{14}(50) + S_{15}(50)$</td>
</tr>
<tr>
<td>$\nu_2$</td>
<td>$a'$</td>
<td>CH$_2$ str.</td>
<td>3135.0</td>
<td>-129.1</td>
<td>3005.9</td>
<td>(2.1)</td>
<td>3010.4</td>
<td>(w)</td>
<td>$S_{14}(50) + S_{15}(50)$</td>
</tr>
<tr>
<td>$\nu_3$</td>
<td>$a'$</td>
<td>CH$_2$ sciss.</td>
<td>1408.3</td>
<td>-53.3</td>
<td>1354.9</td>
<td>(8.4)</td>
<td>1366.4</td>
<td>(m)</td>
<td>$S_{13}(4) + B_{412}(43) + B_{312}(53)$</td>
</tr>
<tr>
<td>$\nu_4$</td>
<td>$a'$</td>
<td>CS</td>
<td>984.6</td>
<td>-23.1</td>
<td>961.5</td>
<td>(24.4)</td>
<td>970.9</td>
<td>(m)</td>
<td>$S_{13}(95) - S_{12}(5) + B_{412}(11)$</td>
</tr>
<tr>
<td>$\nu_5$</td>
<td>$a'$</td>
<td>CH$_2$ rock.</td>
<td>925.0</td>
<td>5.9</td>
<td>930.9</td>
<td>(11.5)</td>
<td>910.2</td>
<td>(m)</td>
<td>$B_{312}(23) + B_{412}(36) + B_{512}(40)$</td>
</tr>
<tr>
<td>$\nu_6$</td>
<td>$a'$</td>
<td>SS str.</td>
<td>621.1</td>
<td>-6.6</td>
<td>614.5</td>
<td>(36.9)</td>
<td>622.8</td>
<td>(s)</td>
<td>$S_{12}(62) + B_{412}(37)$</td>
</tr>
<tr>
<td>$\nu_7$</td>
<td>$a'$</td>
<td>def.</td>
<td>302.1</td>
<td>-1.8</td>
<td>300.3</td>
<td>(0.8)</td>
<td>–</td>
<td>–</td>
<td>$S_{12}(45) + B_{312}(40) + B_{412}(9) + B_{512}(6)$</td>
</tr>
<tr>
<td>$\nu_8$</td>
<td>$a''$</td>
<td>CH$_2$ wag.</td>
<td>754.0</td>
<td>0.0</td>
<td>754.0</td>
<td>(51.3)</td>
<td>756.8</td>
<td>(s)</td>
<td>$T_{4123}(64) + T_{3123}(36)$</td>
</tr>
<tr>
<td>$\nu_9$</td>
<td>$a''$</td>
<td>CH$_2$ twist.</td>
<td>557.3</td>
<td>-17.6</td>
<td>539.7</td>
<td>(2.8)</td>
<td>–</td>
<td>–</td>
<td>$T_{4123}(36) + T_{3123}(64)$</td>
</tr>
</tbody>
</table>

| H$_2$CS$_2$ ($C_{2v}$) | | | | | | | | | |
| $\nu_1$ | $a_1$ | CH$_2$ str. | 3127.5 | -127.3 | 3000.2 | (7.2) | 2986.7 | (m) | $S_{14}(50) + S_{15}(50)$ |
| $\nu_2$ | $a_1$ | CH$_2$ sciss. | 1467.1 | -41.7 | 1425.4 | (1.8) | 1412.0 | (w) | $B_{312}(29) + B_{412}(29) + T_{4123}(21) + T_{3121}(21)$ |
| $\nu_3$ | $a_1$ | ring str. | 860.1 | -18.4 | 841.7 | (0.7) | – | – | $S_{12}(52) + S_{13}(52)$ |
| $\nu_4$ | $a_1$ | SS str. | 501.2 | -5.0 | 496.3 | (1.1) | – | – | $B_{312}(104)$ |
| $\nu_5$ | $a_2$ | CH$_2$ twist. | 954.6 | -20.7 | 933.7 | (- -) | – | – | $B_{312}(21) + B_{412}(21) + T_{4123}(29) + T_{3121}(29)$ |
| $\nu_6$ | $b_1$ | CH$_2$ str. | 3226.2 | -150.0 | 3076.2 | (0.0) | – | – | $S_{13}(45) + S_{12}(5) + S_{14}(50)$ |
| $\nu_7$ | $b_1$ | CH$_2$ rock. | 941.7 | -4.1 | 933.7 | (2.0) | 928.3 | (m) | $B_{312}(29) + B_{412}(29) + T_{4123}(21) + T_{3121}(21)$ |
| $\nu_8$ | $b_2$ | CH$_2$ wag. | 1067.8 | -24.0 | 1043.8 | (1.2) | 1053.0 | (m) | $B_{312}(21) + B_{412}(21) + T_{4123}(30) + T_{3121}(30)$ |
| $\nu_9$ | $b_2$ | ring str. | 596.4 | -14.3 | 582.1 | (8.8) | 581.0 | (s) | $S_{12}(52) + S_{13}(52)$ |

The CH$_2$ rocking, $\nu_5$, is comprised in roughly equal measure of bending of the peripheral atoms around the central carbon ($B_{312}, B_{412}, B_{512}$), giving rise to a dynamic contraction and relaxation of the polar ends of the molecule, and thereby inducing a moderately strong IR absorption of intensity 11.5 km mol$^{-1}$. This appears as a band of medium intensity in the experiments of Maier et al.\textsuperscript{31}

The vibrational frequencies corresponding to CH$_2$ stretching and scissoring ($\nu_1, \nu_2, \nu_3$) bear considerable similarity to the oxygen analog – cf., the NEVPT2/aug-cc-pVDZ computed band of a C–H stretch is 3149 cm$^{-1}$ in the Criegee intermediate (1b),\textsuperscript{28} and 3115 cm$^{-1}$ in the present system (CH$_2$SS, 2b). Salient frequency differences involve modes of the chalcogen atoms. Notably, the O–O stretch of CH$_2$O (1b) is experimentally shown\textsuperscript{28} to be 908 cm$^{-1}$, while the predicted gas-phase S–S stretch in CH$_2$SS (2b) is 615 cm$^{-1}$. This disparity is attributable to the differing masses of the chalcogen atoms.\textsuperscript{130}

The imaginary mode, $\nu_9$, of the transition state (see Figure 3.3) includes S–S lengthening, the
Table 3.3: Harmonic frequencies $\omega$ (cm$^{-1}$) and intensities (km mol$^{-1}$) for the $C_1$ transition state linking thioformaldehyde $S$-sulfide (CH$_2$SS) and dithiirane (H$_2$CS$_2$). Also see Table 3.2 caption.

<table>
<thead>
<tr>
<th>Mode</th>
<th>Description $^a$</th>
<th>$\omega$</th>
<th>Int</th>
<th>TED: Coordinate (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\nu_1$</td>
<td>CH$_2$ str.</td>
<td>3253.8</td>
<td>(1.2)</td>
<td>$S_{14}(68) + S_{15}(32)$</td>
</tr>
<tr>
<td>$\nu_2$</td>
<td>CH$_2$ str.</td>
<td>3121.6</td>
<td>(0.9)</td>
<td>$S_{14}(32) + S_{15}(68)$</td>
</tr>
<tr>
<td>$\nu_3$</td>
<td>CH$_2$ sciss.</td>
<td>1442.5</td>
<td>(2.6)</td>
<td>$B_{412}(26) + T_{4123}(25) + T_{5123}(44)$</td>
</tr>
<tr>
<td>$\nu_4$</td>
<td>CS str.</td>
<td>971.5</td>
<td>(1.1)</td>
<td>$B_{412}(26) + T_{4123}(25) + T_{5123}(44)$</td>
</tr>
<tr>
<td>$\nu_5$</td>
<td>CH$_2$ rock.</td>
<td>913.8</td>
<td>(1.9)</td>
<td>$S_{13}(74) + B_{512}(22)$</td>
</tr>
<tr>
<td>$\nu_6$</td>
<td>SS str.</td>
<td>773.1</td>
<td>(34.3)</td>
<td>$S_{13}(13) + B_{312}(23) + B_{412}(32) + B_{512}(27)$</td>
</tr>
<tr>
<td>$\nu_7$</td>
<td>def.</td>
<td>595.1</td>
<td>(24.6)</td>
<td>$S_{12}(20) + B_{312}(14) + B_{412}(22) + B_{512}(27) + T_{4123}(17)$</td>
</tr>
<tr>
<td>$\nu_8$</td>
<td>CH$_2$ wag.</td>
<td>523.3</td>
<td>(5.4)</td>
<td>$S_{12}(44) + B_{312}(55)$</td>
</tr>
<tr>
<td>$\nu_9$</td>
<td>CH$_2$ twist.</td>
<td>586.1</td>
<td>(52.3)</td>
<td>$S_{12}(34) + B_{312}(35) + B_{512}(13) + T_{4123}(8) + T_{5123}(4)$</td>
</tr>
</tbody>
</table>

continued bending of carbon and the two sulfurs ($B_{312}, B_{512}$), and twisting of the hydrogens relative to carbon and sulfur ($T_{4123}, T_{5123}$). The C—S stretching mode of the transition state, $\nu_4$, also features a large contribution from this twisting motion. Modes $\nu_6$ and $\nu_7$ describe a deformation of the entire molecule as it isomerizes from planar $C_s$ symmetry toward a structure bearing the methylene oriented at a right angle to the planar three-membered ring.

In dithiirane (H$_2$CS$_2$, 2c) the CH$_2$ twist is IR inactive. Furthermore, the IR bands of H$_2$CS$_2$ (2c) are generally much less intense than those of CH$_2$SS (2b), due to the reduced dipole moment, which is matched by experiment. The most intense modes involve a “breathing” motion ($\nu_9$) formed by the stretching of both symmetrical C—S bonds to induce ring stretching, as well as the methylene C—H symmetric stretches ($\nu_1$). The S—S stretching mode, $\nu_4$, is coincident with the bending of the SCS angle ($B_{312}$), and less intense. The largest deviation from experiment is the CH$_2$ stretching mode in the $C_{2v}$ structure H$_2$CS$_2$ (2c), $\nu_1$, wherein the current study’s predictions of gas-phase frequency differ from the published argon matrix IR spectroscopy value by 13.5 cm$^{-1}$. This discrepancy is for a high-frequency mode, and the corresponding errors are amplified. Part of this discrepancy could also be a consequence of matrix effects.

Rotational and Vibration-Rotation Interaction Constants

Equilibrium rotational constants for the species under study ($A_e, B_e, C_e$) are reported in Table 3.4. Zero-point corrected values ($A_0, B_0, C_0$) are also provided, and account for the vibrational
dependence of $B_0$ along the principal axis $b$ by the expression\textsuperscript{97}

\[ B_0 = B_e - \frac{1}{2} \sum_r \alpha_r^B \] (3.2)

where $\alpha_r^B$ are vibration-rotation interaction constants. These constants, $\alpha_r^B$, describe the coupling of rotations about the principal axis corresponding to $B$ with the normal mode $r$; the computed values appear in Table 3.5. (Analogous expressions apply to the vibrational dependence of $A_0$ and $C_0$, which involve rotation about the axes $a$ and $c$, respectively.) The high accuracy of $\alpha_r^B$ constants obtained from \textit{ab initio} theory has been demonstrated previously.\textsuperscript{87,98,131,132}

The vibrationally averaged rotational constants $A_0$, $B_0$, and $C_0$ are lower than their equilibrium counterparts by 0.3, 0.4, and 0.5\%, respectively. Both CH$_2$SS (2b) and H$_2$CS$_2$ (2c) are asymmetric tops; however, by Ray’s asymmetry parameter $\kappa$:\textsuperscript{133}

\[ \kappa = \frac{2B - A - C}{A - C} \]

\[ \kappa \rightarrow \begin{cases} -1.0 \implies \text{near prolate} \\ +1.0 \implies \text{near oblate} \end{cases} \] (3.3)

CH$_2$SS (2b) is near prolate ($\kappa = -0.95$) and H$_2$CS$_2$ (2c) is approximately prolate ($\kappa = -0.74$). The rotational constants of CH$_2$SS (2b) are 62\% smaller than those of the Criegee intermediate CH$_2$OO (1b) computed by Bowman, Guo, and colleagues ($A_0 = 2.5934$, $B_0 = 0.4158$, $C_0 = 0.3576$).\textsuperscript{126} This is to be expected, as smaller rotational constants are observed with heavier atoms and longer bond lengths.\textsuperscript{130}

\textit{Bonding Analysis}

Cremer, Schmidt, Gauss, and Radhakarishnan describe the cyclization of the Criegee intermediate as being initiated by donation of an electron from the carbon atom to the neighboring oxygen.\textsuperscript{134} The lowest unoccupied molecular orbital (LUMO) of CH$_2$SS (2b) features a large $3p$ orbital over the proximal sulfur, which is the electron sink for the highest occupied molecular orbital (HOMO) of the methylene carbon. The HOMO of CH$_2$SS (2b) is formed from a large $3p$ orbital on the non-bonding sulfur atom, as well as overlapping $p$-orbitals on the bonded sulfur and carbon atoms, for which Natural Bond Orbital (NBO) analysis (B3LYP/cc-pVTZ) describe as possessing $sp^2$ and
Table 3.4: Vibrational contributions to the equilibrium rotational constants and dipole moments were obtained from VPT2 analysis [CCSD(T)/aug-cc-pVTZ, +d functions on sulfur]. Rotational constants are in cm$^{-1}$ and dipole moments are in Debye. $\mu$ is the equilibrium dipole moment, whereas $\langle \mu \rangle$ is the vibrationally averaged dipole moment.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Thioformaldehyde S-sulfide (CH$_2$SS)</th>
<th>Dithiirane (H$_2$CS$_2$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A_e$</td>
<td>0.9663</td>
<td>0.5675</td>
</tr>
<tr>
<td>$B_e$</td>
<td>0.1604</td>
<td>0.2335</td>
</tr>
<tr>
<td>$C_e$</td>
<td>0.1376</td>
<td>0.1712</td>
</tr>
<tr>
<td>$A_0$</td>
<td>0.9638</td>
<td>0.5626</td>
</tr>
<tr>
<td>$B_0$</td>
<td>0.1598</td>
<td>0.2329</td>
</tr>
<tr>
<td>$C_0$</td>
<td>0.1369</td>
<td>0.1702</td>
</tr>
<tr>
<td>$</td>
<td>A_e - A_0</td>
<td>$</td>
</tr>
<tr>
<td>$</td>
<td>B_e - B_0</td>
<td>$</td>
</tr>
<tr>
<td>$</td>
<td>C_e - C_0</td>
<td>$</td>
</tr>
<tr>
<td>$\mu$</td>
<td>2.89</td>
<td>1.56</td>
</tr>
<tr>
<td>$\langle \mu \rangle$</td>
<td>2.83</td>
<td>1.53</td>
</tr>
</tbody>
</table>

Table 3.5: Vibration-rotation interaction constants ($\alpha^B_r$, in 10$^{-3}$ cm$^{-1}$) for thioformaldehyde S-sulfide (CH$_2$SS) and dithiirane (H$_2$CS$_2$).

<table>
<thead>
<tr>
<th>Mode ($r$)</th>
<th>CH$_2$SS</th>
<th>H$_2$CS$_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$A$</td>
<td>$B$</td>
</tr>
<tr>
<td>1</td>
<td>1.720</td>
<td>0.072</td>
</tr>
<tr>
<td>2</td>
<td>2.087</td>
<td>0.048</td>
</tr>
<tr>
<td>3</td>
<td>-2.180</td>
<td>-0.044</td>
</tr>
<tr>
<td>4</td>
<td>5.352</td>
<td>0.296</td>
</tr>
<tr>
<td>5</td>
<td>-4.430</td>
<td>-0.108</td>
</tr>
<tr>
<td>6</td>
<td>-1.463</td>
<td>0.637</td>
</tr>
<tr>
<td>7</td>
<td>-8.811</td>
<td>0.011</td>
</tr>
<tr>
<td>8</td>
<td>4.016</td>
<td>0.195</td>
</tr>
<tr>
<td>9</td>
<td>8.770</td>
<td>0.100</td>
</tr>
</tbody>
</table>

$sp^3$ character, respectively. The overlapping $p$-orbitals observed in the HOMO are consistent with the double-bond character of the dominant resonance structure in CH$_2$SS (2b; also see Figure 3.5). The bond order of the C—S bond ($b_{CS}$) in the transition state is reduced from to 1.7 to 1.3, and the $p$ character is increased on the bonded sulfur atom. This rearrangement (Figure 3.3) entails a concerted electrocyclic mechanism, whereby ring closure occurs with the formation of a new sigma bond, concomitant with the reduction of the double bond to a single bond. In H$_2$CS$_2$ (2c), the molecular orbital of each sulfur donates an electron into the methylene LUMO to furnish the symmetric structure. The bond order of the C—S bond in H$_2$CS$_2$ (2c) is 1.0, and the sulfur atoms are
described by \( p \) character. The HOMO-LUMO gap of \( \text{H}_2\text{CS}_2 \) (11.5 eV) is larger than that of \( \text{CH}_2\text{SS} \) (9.1 eV), indicative of lesser thermal reactivity.\(^{135}\)

For \( \text{CH}_2\text{SS} \ (2b) \), natural population analysis yields natural charges for the carbon atom of -0.58 e (where \( e = 1.6 \times 10^{-19} \) coulomb), along with the bonded sulfur atom (0.51 e), and the non-bonded sulfur atom (-0.36 e), see Figure 3.4. The leading resonance structure features a double bond between the carbon and proximal sulfur atoms, a single bond between the two sulfur atoms, and three lone pairs (one contributing to a negative charge) on the terminal sulfur. These features of \( \text{CH}_2\text{SS} \ (2b) \) match the leading zwitterionic form of carbonyl \( O \)-oxide, \( \text{CH}_2\text{OO} \ (1b) \).\(^{136}\) The natural bond order analysis reveals, for \( \text{CH}_2\text{SS} \), an 84% covalent (16% ionic) bond between the carbon and sulfur atoms \( (b_{\text{CS}} = 1.7) \), and 68% covalent bond between the two sulfur atoms \( (b_{\text{SS}} = 1.2) \). While this structure describes 58% of the resonance character, an important contributor (22%) is one in which the double bond is located between the two sulfur atoms. A minor contributor is one in which a single bond forms between the terminal sulfur atom and the carbon atom, rendering the primary C—S interaction a single bond; this structure is akin to the transition state. Interestingly, in \( \text{CH}_2\text{OO} \ (1b) \), this same transition state-like structure, in addition to the dominant C—O double bond resonance form, comprises its sole resonance contributors, notably absent an O—O double bond structure. This is reasonable given that \( \text{CH}_2\text{OO} \)’s smaller interatomic distances foster the formation of a new bond. The dipole moment of \( \text{CH}_2\text{SS} \ (2b) \) is 2.89 Debye, considerably less than the 4.0 Debye dipole moment of \( \text{CH}_2\text{OO} \ (1b) \),\(^{137}\) consistent with the smaller electronegativity of sulfur.

The transition state (Figures 3.4 and 3.5) displays reduced polarized charges: -0.45 e on the carbon atom, and 0.28 and -0.25 e on the proximal and terminal sulfur atoms, respectively. The leading resonance structure from natural bond order analysis (54%) features single bonds between the carbon atom and both sulfur atoms, akin to the product. The bond between the proximal sulfur atom and carbon atom is 82% covalent \( (b_{\text{CS}} = 1.3) \), while the newly formed bond between the terminal sulfur atom and the carbon atom is just 25% covalent \( (b_{\text{CS}} = 0.6) \). The covalent character of the S—S bond is increased from its predecessor to 80% \( (b_{\text{SS}} = 1.1) \). In the second leading resonance structure (26%), which is similar to the reactant, the double bond exists between the carbon and proximal sulfur atom; the bond between the carbon atom and terminal sulfur atom
Figure 3.5: Resonance contributions of the equilibrium stationary points: Thioformaldehyde $S$-sulfide (2b); cyclization transition state (TS); dithiirane (2c); carbonyl $O$-oxide, the Criegee intermediate (1b); dioxirane (1c).
is absent. These relative contributions support the interpretation that Hammond’s postulate may not apply to this system.

In dithiirane (2c) both C—S bonds are 88% covalent ($b_{CS} = 1.0$), whereas the S—S bond is entirely covalent ($b_{SS} = 1.0$). This nearly exclusive (94%) “resonance” structure features single bonds between all atoms. A similar picture describes the dominant form of CH$_2$OO (1b). The carbon atom of CH$_2$SS has a natural charge, -0.68 e, while the other atoms are nearly neutral. The dipole moment is reduced to 1.56 Debye.

### 3.5 Conclusion

A high-level *ab initio* study of the sulfur analog of the Criegee intermediate (thioformaldehyde S-sulfide), its valence isomer (dithiirane), and the transition state linking these molecules is described. The results of a focal point extrapolation and vibrational analysis indicate that dithiirane is 7.2 kcal mol$^{-1}$ lower in energy than thioformaldehyde S-sulfide, and that there is a 27.0 kcal mol$^{-1}$ barrier to cyclization. The resonance contributors of the $C_s$-symmetric thioformaldehyde S-sulfide structure are predominantly characterized by the presence of a double-bond bond between the carbon and proximal sulfur atoms, and a third lone pair on the terminal sulfur atom, lending it a negative charge. The $C_{2v}$-symmetric structure dithiirane is formed by a ring-closing, electrocyclic mechanism, and features lengthened S—S and C—S bonds relative to the antecedent structure. Dithiirane’s S—S bond is almost exclusively covalent and of single-bond character, while its two C—S bonds are 88% covalent. The geometric parameters of the transition state generally bear more resemblance to the thioformaldehyde S-sulfide reactant than the dithiirane product. The computed fundamental frequencies are in very good agreement with the argon matrix experiments of Maier et al.,$^{31}$ and our predictions of the currently unknown gas-phase infrared bands should be reliable.
CHAPTER 4

A STUDENT-FRIENDLY GUIDE TO MOLECULAR INTEGRALS

\footnote{K.V. Murphy, J.M. Turney, and H.F. Shaefer. Submitted to \textit{J. Chem. Ed.}, April 2018.}
4.1 Abstract

Before 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 included as supplementary information documents, 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 discuss such topics as the atomic orbital basis. 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.

4.2 Background to Molecular Integrals and Hartree–Fock Theory

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 him or her 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 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*: they are not the fastest means of generating these numbers. There is a rich literature dedicated to this purpose, 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.

**Article Structure**

The main body of the present article sets the stage for molecular integrals by describing their role in the HF equations. Because of the detail necessary to explicate this topic, the bulk of the article actually appears in separate “handout” documents of the supplementary information (SI), as outlined in Fig. 4.1. It is recommended the documents be read in order.

| SI 1. Handout 1: *Derivation* of the molecular integrals, with exercises |
| SI 2. Handout 2: *Coding* in Python: The essentials (with exercises) |
| SI 3. Handout 3: *Implementation* of the molecular integrals: Programming project |
| SI 4. Handout 4: *Using* the molecular integrals: HF programming project |
| SI 5. Python code files: |
| 5a. Basis, with STO-3G basis set through krypton (basis.py) |
| 5b. One-electron integrals (oei.py) |
| 5c. Two-electron integrals (eri.py) |
| 5d. HF (scf.py) |
| 5e. Driver of code (main.py) |

Figure 4.1: Supplementary information contents, structured as separate handouts. It is advised the documents be read in the order presented, after reading the main article. The experienced programmer may skip SI (Handout) 2.
The first of these Handout documents forms the core of this publication when coupled with the parent article. It features the full derivation of the molecular integral equations, with many exercises for active participation by the reader. These equations and their derivation are based on the original 1966 paper by Taketa, Huzinaga, and O-ohata\textsuperscript{139}, and use notation developed by Cook\textsuperscript{140}. The second Handout document is a brief 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. The coding introduction is followed by the full implementation of the molecular integrals in Handout 3. Having generated the molecular integrals, Handout 4 describes their deployment in the implementation of the HF method. Completion of the materials up through this fourth document marks the development of a complete, independent quantum chemistry program. The final document, SI 5, compiles 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 only as a last resort and only as needed. Otherwise the learning value is forfeit.

The main article concludes with a brief discussion of the authors’ experience using this programming project 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\textsuperscript{1,141,142} and \textit{J. Chem. Ed.} articles,\textsuperscript{143–155} if review is needed.

\textit{Molecular Integrals and the HF Method}

At the core of any molecular modeling computation based upon quantum mechanical methods (as opposed to classical mechanical methods such as those generally termed “molecular mechanics”) are the \textit{molecular integrals}.

In computational molecular modeling using quantum chemical methods, the chemist seeks to determine the energy of a molecular system associated with some geometry, but without employing any empirical parameters. Quantum chemistry is an \textit{ab initio} method, whereby energies of atomic
and molecular systems are computed directly from first principles, rather than depending on any experimental measurements. In a sense this requires asking the computer to solve Schrödinger’s differential equation,

\[ \hat{H} \Psi = E \Psi. \]  

(4.1)

The wave function \( \Psi \) is difficult to define in tangible terms. Suffice it to say that it contains information about the system we are interested in, and that it has a mathematical form. This system could be a free electron, an atom, a molecule, or something hypothetical like a particle in an energy well. Physically meaningful information is extracted from the wave function when an operator is applied to it. The most important operator of quantum mechanics is the Hamiltonian \( \hat{H} \),

\[ \hat{H} = \hat{T} + \hat{V}, \]  

(4.2)

where \( \hat{T} \) is kinetic energy and \( \hat{V} \) is potential energy. When the wave function \( \Psi \) is subjected to the Hamiltonian, as in the Schrödinger equation (Eq. 4.1), the values of energy \( E \) are returned as multiplicative constants of an unaltered \( \Psi \).

Whether one can do the complicated mathematics necessary to find the solutions \( E \) and \( \Psi \) is another matter. The undergraduate physical chemist learns integro-differential techniques for deriving the solutions – calculating the energies and wave functions – of a few systems, including the hypothetical particle in a box, the harmonic oscillator, and perhaps the hydrogen atom. In all three cases, solving Schrödinger’s equation is the eigenvalue problem of finding an eigenfunction (wave function) and eigenvalue (energy). These systems are three of only a handful for which mathematically exact, or analytical, solutions exist. Such analytically-soluble systems are precious few because of the many-body problem, whereby the exact solution of a system composed of more than two bodies is mathematically intractable. The particle in a box, whether in one dimension or three, is just one body. The hydrogen atom matches the many-body limit with one proton and one electron.

The many-body problem is particularly irksome to the chemist, who is interested in solving
Schrödinger’s equation for systems composed of many bodies (nuclei and electrons) – not only the many bodies of atoms more complex than hydrogen, but of course those comprising molecules. Fortunately, certain approximations make the application of Schrödinger’s equation to chemically relevant many-body systems possible. The first of these is the Born–Oppenheimer approximation,\(^{156}\) which proffers that the nuclear motion of an atom or molecule can be decoupled from electronic motion, because nuclei are thousands of times heavier than electrons and move much slower. Hence, quantum chemistry comes down to solving the electronic Schrödinger equation,

\[
\hat{H}_e \Psi_e = E_e \Psi_e ,
\]  

which uses the molecule’s geometry as fixed parameters, but leaves out the nuclear operator terms of the Hamiltonian. The operator of the electronic Hamiltonian,

\[
\hat{H}_e = \sum_i \left( \frac{-1}{2} \nabla_i^2 - \sum_C \frac{Z_C}{r_{iC}} \right) + \sum_{i<j} \frac{1}{r_{ij}} ,
\]

informs the structure of the molecular integrals. There are one-electron components, \(h(i)\), as well as two-electron components, \(v(i,j)\). The one-electron components (themselves operators) contain the kinetic energy, which notably features the Laplace operator,

\[
\nabla_i^2 = \frac{\partial^2}{\partial x_i^2} + \frac{\partial^2}{\partial y_i^2} + \frac{\partial^2}{\partial z_i^2} .
\]

Eq. 4.4 also has a term for nuclear attraction, where \(r_{iC}\) implies the distance between electron \(i\) and nucleus \(C\). \(Z_C\) is the atomic number of nucleus \(C\) and is just a numerical constant. The two-electron component is the electron repulsion operator, and implies the distance between electrons \(i\) and \(j\). Note that atomic units are employed throughout this paper, which substantially simplifies equations; see Table 4.1.

Another important approximation in quantum chemistry is that of molecular orbital (MO) theory. This states that the wave function, \(\Psi\), can be decomposed into a combination of molecular
Table 4.1: The present paper employs atomic units (a.u.), the standard unit of quantum chemistry. Conversion to SI units is possible as shown.

<table>
<thead>
<tr>
<th>Physical quantity</th>
<th>Atomic units</th>
<th>SI units</th>
</tr>
</thead>
<tbody>
<tr>
<td>Length</td>
<td>$1 \ a_0$</td>
<td>$5.2918 \times 10^{-11} \ m$</td>
</tr>
<tr>
<td>Mass</td>
<td>$1 \ m_e$</td>
<td>$9.1095 \times 10^{-31} \ kg$</td>
</tr>
<tr>
<td>Charge</td>
<td>$1 \ e$</td>
<td>$1.6022 \times 10^{-19} \ C$</td>
</tr>
<tr>
<td>Energy</td>
<td>$1 \ E_h$ (Hartree)</td>
<td>$4.3598 \times 10^{-18} \ J$</td>
</tr>
<tr>
<td>Angular momentum</td>
<td>$1 \ h$</td>
<td>$1.0546 \times 10^{-34} \ J \ s$</td>
</tr>
<tr>
<td>Wave function</td>
<td>$1 \ a_0^{-3/2}$</td>
<td>$2.5978 \times 10^{15} \ m^{-3/2}$</td>
</tr>
</tbody>
</table>

orbitals $\psi_i$:

$$\Psi = \psi_1 \psi_2 ... \psi_n.$$ (4.6)

Note that the $e$ subscript has been dropped. (In order to satisfy the principle of antisymmetry, this product of molecular orbitals is recast as a determinant, but that will not have direct consequence on the evaluation of molecular integrals, nor on implementing the HF method.) Each molecular orbital is constructed as a sum of weighted contribution from each atomic orbital:

$$\psi_i = \sum_{\mu=1}^{N} C_{\mu i} \phi_{\mu}.$$ (4.7)

In fact, Eqs. 4.6 and 4.7 are nothing more than the mathematical formulation of the Linear Combination of Atomic Orbitals – Molecular Orbital (LCAO–MO) theory, which may be introduced in freshman chemistry. The coefficient $C_{\mu i}$ is simply a number that indicates how much atomic orbital $\phi_{\mu}$ contributes to molecular orbital $\psi_i$.

But how is $C_{\mu i}$ determined, and what does $\phi_{\mu}$ look like? We address these questions by presenting the HF equations (more accurately, the HF–Roothan–Hall$^{157,158}$ equations). Only the essential HF equations necessary for motivating the molecular integrals are presented; their full derivation is provided in the classic text by Szabo and Ostlund$^{3}$, and discussed in other $J. \ Chem. \ Ed.$ articles$^{159,160}$. The central HF equation is

$$FC = SCe.$$ (4.8)
Each term is a two-dimensional matrix whose size is determined by the number of atomic orbitals present.

The HF equation, Eq. 4.8, is the computational chemist’s approximation to the Schrödinger equation. It even bears a similar form: The Hamiltonian operator $\hat{H}$ has been substituted by $F$, the “Fock operator” matrix; the wave function $\Psi$ has been substituted by $C$, which is the matrix of $C_{\mu i}$ coefficients. Each column of the matrix $C$ represents one molecular orbital; in the language of linear algebra, each column of $C$ is an eigenvector. Analogous to the eigenfunctions in a differential eigenvalue equation, eigenvectors are the molecular orbitals. More specifically, we say that molecular orbitals are eigenvectors “in the atomic orbital basis,” because of Eq. 4.7. The vector $\epsilon$ holds the eigenvalues (analogous to $E$), and so it is a one-dimensional matrix of energies. The first eigenvalue, or element of the vector $\epsilon$, is the energy of the first eigenvector (column) in $C$; the second eigenvalue is the energy of the second eigenvector in $C$; and so on.

Each eigenvector of $C$ is a linear combination of the atomic orbitals. This means that each element of this column vector is a number, the coefficient $C_{\mu i}$, indicating how much of a particular atomic orbital $\phi_\mu$ contributes to the molecular orbital $\psi_i$.

One thing clearly distinguishing the Schrödinger equation (Eq. 4.1) from the HF equation (Eq. 4.8) is the presence of $S$, which is a matrix of overlap integrals. Every element of this matrix quantifies to what extent one atomic orbital of the molecule overlaps with another atomic orbital; the maximal value is 1, because orbitals are normalized. The value of 1, indicating perfect orbital overlap, only occurs when describing the overlap of an orbital with itself. $S$ is the first of the molecular integral matrices we need to generate to solve the HF equation.

The Fock operator matrix $F$ is a sum of three two-dimensional matrices,

$$F = T + V + P,$$  \hspace{1cm} (4.9)

or, element-by-element,

$$F_{\mu\nu} = T_{\mu\nu} + V_{\mu\nu} + P_{\mu\nu}$$

$$= T_{\mu\nu} + V_{\mu\nu} + \sum_{\lambda\sigma} D_{\lambda\sigma} \left( G_{\mu\nu\sigma\lambda} - \frac{1}{2} G_{\mu\lambda\sigma\nu} \right).$$  \hspace{1cm} (4.10)
This is where the remaining three molecular integrals appear. $T$ represents the kinetic energy integrals matrix; $V$ is the nuclear attraction integrals matrix. The four-index matrix $G$ (note the four subscript terms) makes up part of $P$, and is the electron repulsion integrals rank-4 tensor. That $G$ is a rank-4 tensor simply means it is a four-dimensional matrix. All of the usual matrix algebra operations that apply to two-dimensional matrices, which can be considered rank-2 tensors, also apply to rank-4 tensors such as $G$; these operations are all that we need. Figs. 4.14 and 4.15 of Handout 1 offer one way of visualizing a rank-4 tensor.

The density matrix $D$, in $P$ of Eq. 4.10, is built from the molecular orbitals:

$$D_{\lambda\sigma} = \sum_{i}^{N/2} C_{\lambda i} C_{\sigma i}.$$  \hspace{1cm} (4.11)

It is formed by multiplying together the lower $N/2$ molecular orbitals of the wave function, where $N$ is the number of electrons, and each orbital holds two electrons. These lower $N/2$ orbitals are the occupied orbitals, whereas the upper $N/2$ orbitals are unoccupied or, in the parlance of quantum chemistry, virtual.

The four molecular integral matrices we are interested in generating, then, are:

1. $S$: overlap integrals matrix
2. $T$: kinetic energy integrals matrix
3. $V$: electron-nuclear attraction integrals matrix
4. $G$: electron-electron repulsion integrals tensor

Each integral matrix is formed only once, and remains unchanged during the HF procedure. By contrast, the molecular orbitals $C$ (and hence the density $D$) change with each iteration, until there is no change between cycles. It is a successive process of energy minimization, each step tweaking the orbital coefficients until a set of molecular orbitals that gives the lowest total electronic energy for the system is produced.\textsuperscript{161}

The determination of a molecular orbital $\psi_i$ additionally yields its energy $\epsilon_i$. If we have every $\psi_i$, then we have the wave function $\Psi$ (Eq. 4.6). The computational equivalent of the Schrödinger equation has been solved.
The total electronic energy, or the HF energy, of the system does not come directly out of Eq. 4.8. But it is an expectation value that is easily evaluated, the equation for which appears in the HF project, Handout 4.

Of course, before we can actually execute the HF iterative procedure, we must populate the matrices with numbers. The question remains as to how to calculate the matrix elements of $S$, $T$, $V$, and $G$. Individual elements are where integrals are actually solved, which when solved yield numbers in the energy units Hartree. This is because each element is an expectation value. The form of the expectation value for a generic operator $\hat{O}$ is

$$\langle O \rangle = \int \phi_A \hat{O} \phi_B \, d\tau.$$  (4.12)

Because we form molecular orbitals $\psi_i$ out of atomic orbitals $\phi_\mu$, the operators act on $\phi_\mu$, as Eq. 4.12 shows. The linear combination of these expectation values over $\phi_\mu$ is equal to having the expectation value of $\psi_i$.

Based on our discussion of the matrices necessary to solve the HF equation (Eq. 4.8), the operators required for calculating the matrix elements are kinetic energy, nuclear attraction energy, and electron repulsion energy. These are found in the electronic Hamiltonian, Eq. 4.4.

Consider the matrix element $T_{AB}$, which corresponds to the first one-electron operator of Eq. 4.4:

$$T_{AB} = \langle T \rangle = \int \phi_A \hat{T} \phi_B \, d\tau = \int \phi_A \left\{ -\frac{1}{2} \nabla^2 \right\} \phi_B \, d\tau.$$  (4.13)

$\hat{T}$ acts on orbital $\phi_B$. The integrand is formed by multiplying the result of this operation by orbital $\phi_A$; integration of the integrand is over all space, $d\tau$. We will also use the notation $(A| - \frac{1}{2} \nabla^2 |B)$ to express this single matrix element, where $A$ represents orbital $\phi_A$, and $B$ represents orbital $\phi_B$.

The matrix elements $V_{AB}$ of the electron-nuclear attraction integral matrix are

$$V_{AB} = \langle V \rangle = (A| - \frac{Z_C}{r_{iC}} |B) = \int \phi_A \hat{V} \phi_B \, d\tau = \int \phi_A \left\{ -\frac{Z_C}{r_{iC}} \right\} \phi_B \, d\tau.$$  (4.14)

The operator features the atomic number of atom $C$ in the numerator; $r_{iC}$ indicates the distance between electron $i$ and nucleus $C$. 

57
For the matrix elements of $G$ – perhaps “tensor elements” would be more apt – we have two electrons $i$ and $j$:

$$G_{ABCD} = (G) = (AB|CD) = \int \phi_A \phi_B \tilde{G} \phi_C \phi_D \, d\tau = \int \phi_A \phi_B \left( \frac{1}{r_{ij}} \right) \phi_C \phi_D \, d\tau . \quad (4.15)$$

It is a peculiar convention that the notation $(AB|CD)$ does not show the operator, but it is implied.

Of course, matrix elements of the overlap integral matrix are needed. For these, there is no operator, and the integrand is simply the product of two orbitals,

$$S_{AB} = (A|B) = \int \phi_A \phi_B \, d\tau . \quad (4.16)$$

It is worth noting that the elements of matrices $S$, $T$, and $V$ are two-center, one-electron integrals (orbitals $A$ and $B$, for electron $i$), whereas elements of the tensor $G$ are four-center, two-electron integrals (orbitals $A, B, C$ and $D$, for electrons $i$ and $j$).

Before these integrals can be evaluated, we must consider the mathematical form the orbitals $\phi_\mu$ take.

**Atomic Orbitals and Gaussian Products**

Molecular orbitals are linear combinations (weighted sums) of atomic orbitals. The expectation value for each operator in the electronic Hamiltonian acts on atomic orbitals. We say that atomic orbitals form our basis, and that each energy integral matrix is in the atomic orbital basis.

In addition to being useful qualitative devices, atomic orbitals (the “objects” that electrons occupy) are mathematical constructions. Their mathematical form, which cannot be exactly known but only approximated, is based on the assumption that the orbitals for any atom can be approximated by the known mathematical form of the orbitals of the hydrogen atom. (Erwin Schrödinger derived the equations of hydrogen’s orbitals. Due to the many-body problem, the mathematics is intractable for determining exact orbital equations for any other atom.)

The American physicist John C. Slater suggested this approximation in 1930\(^{162}\). Like the orbitals Schrödinger derived for the hydrogen atom, Slater-type orbitals (STO) have an angular
and a radial component,

\[ \phi = N \sqrt[angular]{x^l y^m z^n} e^{-\alpha r}. \]  

(4.17)

In Eq. 4.17, \( N \) is a normalization constant, and \( \alpha \) is a number found in tables for basis sets unique to every atom and orbital (\( s, p, d \), etc.). The variables \( x, y, \) and \( z \) are the three-dimensional Cartesian coordinates describing the position of the electron (the nucleus being at the origin), and the powers \( l, m, \) and \( n \) associated with each coordinate are integers describing the directional contributions to the electron’s orbital angular momentum. The sum \( l + m + n \) indicates the type of orbital to which the electron belongs: \( l + m + n = 0 \) is an \( s \) orbital, \( l + m + n = 1 \) is a \( p \) orbital, \( l + m + n = 2 \) is a \( d \) orbital, and so on. If \( l + m + n = 1 + 0 + 0 = 1 \), then the electron belongs to a \( p_x \) orbital; \( l + m + n = 0 + 2 + 0 = 2 \) is a \( d_{y^2} \) orbital; and so on. The term \( r \) is the distance of the electron from the center of the atom.

\[ r = \sqrt{x^2 + y^2 + z^2} \]  

(4.18)

It turns out that Slater-type orbitals form effectively intractable (i.e., not analytically solvable) integrals (Eqs. 4.13 through 4.16) for anything but atoms, some diatomics, and some linear polyatomic molecules. In 1950, Samuel F. Boys proposed that Slater-type orbitals could be approximated by Gaussian functions:

\[ \phi \approx \sum_{p=1}^{K_p} d_p N_p x^l y^m z^n e^{-\alpha pr^2}. \]  

(4.19)

Therein, \( p \) is a generic index substituted with \( a, b, c, \) or \( d \) when referring to orbitals \( \phi_A, \phi_B, \phi_C, \) or \( \phi_D \), respectively. Notably, Eq. 4.19 is distinguished from Eq. 4.17 by the squaring of \( r \). This has, perhaps surprisingly, enormous significance for the taking of derivatives and the calculation of integrals over multiple centers. This is because of the Gaussian Product Theorem: The product of two Gaussian functions on two different centers yields one Gaussian function located on a center between the original two Gaussians. As a simple demonstration, consider two \( s \) orbitals, \( \phi_A \) and \( \phi_B \), each approximated by Eq. 4.19. Each orbital has \( l = m = n = 0 \). For the moment let’s
disregard the constants $d_p$ and $N_p$. Then,

$$\phi_A \times \phi_B = \exp\left(-\alpha_a r_A^2\right) \times \exp\left(-\alpha_b r_B^2\right)$$

$$= \exp\left[-\left(\frac{\alpha_a \alpha_b}{\alpha_a + \alpha_b}\right) |\vec{A}\vec{B}|^2\right] \times \exp\left[-(\alpha_a + \alpha_b) r_P^2\right]$$

(4.20)

Clearly, the product of two Gaussian functions yields a third, single Gaussian function where $\tilde{K}$ is a constant (not a function of the variable $r$), $|\vec{A}\vec{B}|^2$ is the squared distance between $\vec{A}$ and $\vec{B}$, and $r_P$ describes a position located between $r_A$ and $r_B$. Two centers, $A$ and $B$, are reduced to one, $P$; see Figs. 4.2 and 4.3. It is not hard to imagine that an integral over a single center is simpler to evaluate than an integral over two centers. This vital property is especially valuable when analyzing the four-center, two-electron integrals of $G$: The four centers $A, B, C$ and $D$ are reduced to two centers, $P$ and $Q$.

In addition to the substitution of an exponential radial function for a Gaussian radial function, Eq. 4.19 features summation and a new constant, $d_p$. To highlight their significance, consider the hydrogen 1s orbital approximated by the STO, versus that estimated with a Gaussian function. This Gaussian approximation is also known as a Slater-Type Orbital – 1 Gaussian (STO–1G). The nomenclature STO-NG is a little unfortunate, as it would seem to imply that these functions are Slater-type orbitals. But the terms after the hyphen indicate that they are approximations to STOs by way of a linear combination of Gaussian functions; in the case of STO-1G, just one Gaussian function. (Recall that STO–1G is an approximation to the STO, which is an approximation to the actual orbital, the exact form of which we cannot know.)

$$\phi_{A,\text{STO}} = N_a x^l y^m z^n e^{-\alpha_a r_A} = \frac{1}{\sqrt{\pi}} \times x^0 y^0 z^0 \times e^{-1.0 r_A}$$

$$\phi_{A,\text{STO-1G}} = \sum_{a=1}^{K_a} d_a N_a x^l y^m z^n e^{-\alpha_a r_A} = 1.0 \times 0.26766 \times x^0 y^0 z^0 \times e^{-0.28294 r_A^2}$$

In the STO-1G basis of $\phi_A$, a hydrogen 1s orbital, there is only one coefficient $d_a$ and one exponential factor $\alpha_a$ ($K_a = 1$). As Fig. 4.4 reveals, a single Gaussian function does a poor job of emulating a Slater-type orbital. It grossly underestimates the amplitude of the orbital at small
Figure 4.2: A demonstration of the Gaussian Product Theorem with two \( s \)-type orbitals approximated as Gaussian functions. Ignoring \( y \)- and \( z \)-components, the \( x \)-component of orbital \( \phi_A \) (blue) is centered at \( x = -2.0 \), and the \( x \)-component of orbital \( \phi_B \) (green) is centered at \( x = 1.0 \). The width of each individual orbital is determined by the exponential factor \( \alpha \). The thicker orbital \( \phi_A \) has the smaller exponential factor, \( \alpha_a = 0.10 \); the thinner orbital \( \phi_B \) has the larger exponential factor, \( \alpha_b = 0.30 \). The product of \( \phi_A \) and \( \phi_B \) is the red line; in accordance with Eq. 4.20, it is located between the centers of \( \phi_A \) and \( \phi_B \).

If we add a second Gaussian function to our approximation of STO, which would be the Slater-Type Orbital – 2 Gaussian Functions (STO–2G) basis set, the approximation is better. In this case, we have a linear combination of Gaussian functions. Each Gaussian function is called a Gaussian primitive, and is weighted by a coefficient. This weighting factor, \( d_p \), is known as a contraction coefficient. This contraction of Gaussian primitives gives us the contracted Gaussian-type orbital (CGTO), also known as a contracted Gaussian-type function (CGTF). Thus, with a minimal basis set like STO-3G, Eq. 4.19 is known as a contracted Gaussian-type orbital. (In Handout 1, we will discuss why it is better to refer to Eq. 4.19 as a CGTF when using extended basis sets.) Applied
Figure 4.3: In taking the product of orbitals $\phi_A$ and $\phi_B$, each represented by Cartesian Gaussian functions, a third center $\vec{P}$ located between $\vec{A}$ and $\vec{B}$ is formed. $\vec{r}_P$ is the distance between the electron and the third center $\vec{P}$.

to the hydrogen 1s orbital,

$$\phi_{A,\text{STO-2G}} = \sum_{a=1}^{K_a=2} d_a N_a x^l y^m z^n e^{-\alpha_a r_A^2}$$

$$= d_1 N_1 x^l y^m z^n e^{-\alpha_1 r_A^2} + d_2 N_2 x^l y^m z^n e^{-\alpha_2 r_A^2}$$

$$= 0.67891 \times 0.17359 \times x^0 y^0 z^0 \times e^{-0.233136 r_A^2} + 0.43013 \times 0.63193 \times x^0 y^0 z^0 \times e^{-1.399756 r_A^2}.$$ 

Once again, the basis set parameters $\alpha_p$ and $d_p$ are found from a table. Fig. 3 provides a visualization of this contraction of two Gaussian primitives, and shows that it is a better approximation to a true STO.

As we would expect, the contraction of three Gaussian primitives in the STO-3G basis set improves on the STO–2G approximation; a linear combination of six primitives, STO-6G, does a better job still (see Fig. 4.6). Generally, the more Gaussian functions that are included, the better the approximation to Slater-type orbitals. The price that is paid is computational expense. This is
Figure 4.4: The approximation of a hydrogen 1s Slater-type orbital by a single Gaussian function with the STO-1G basis set. One Gaussian function (STO-1G) does a rather poor job of emulating an STO.

because an integral must be evaluated for each primitive, not just each orbital. In a STO-1G basis, only one integral is evaluated per matrix element. In a STO-3G basis, three integrals are evaluated and summed to form the final value of the matrix element. Higher quality comes at a price. Fig. 4.6 reveals that even STO–6G fails to capture Slater-type orbital behavior very near the nucleus – the cusp condition – and underestimates the function at longer distances. Depending on the molecular system, increasing basis set size beyond a certain point yields diminishing returns. In the programming projects, a balance will be struck between expense and accuracy by employing the STO-3G basis set.

These basis sets and their tables are curated on the EMSL (Environmental Molecular Sciences Laboratory) Basis Set Exchange (https://bse.pnl.gov/bse/portal). An example of how to read these tables is shown in Fig. 4.20.

There are many excellent discussions of how to select basis sets in computations, and so these considerations are not addressed in this paper. The present focus is on the essential content of a basis set, such as the fact that they are composed of contraction coefficients $d_p$ and exponent
Figure 4.5: The approximation of a Slater-type orbital (hydrogen 1s) by Gaussian-type functions with the STO-2G basis set, which is a *contraction* (linear combination) of two weighted (with contraction coefficients) Gaussian *primitives* (Gaussian functions “g1” and “g2”). Hence, the hydrogen 1s orbital constructed with the STO-2G basis set is a *contracted Gaussian-type orbital* (CGTO).

factors $\alpha_p$, as well as their direct relation to the molecular integrals. In addition to the earlier discussion, further details on the basic structure of basis sets is provided in Handout 2.

**Moving Forward**

When a computer “calculates the molecular integrals,” it is not as if it is solving new calculus or differential equation problems with every computation. The solutions to the molecular integrals (Eqs. 4.13 – 4.16) have already been derived\textsuperscript{139}, just as the reader can anticipate the derivation in Handout 1. “Calculating the molecular integrals” is a matter of having the computer substitute numbers in to variables in the known solutions, which are implemented in computer code (as the reader will also implement them). These numbers include the basis set parameters ($\alpha_p$ and $d_p$), and the geometric information about the orbitals in the molecule. Because orbitals are typically centered on atoms, orbital coordinates are taken directly from the molecule coordinates .xyz file. Knowing what orbitals to use depends on both the input file and the basis set: if carbon is an atom of the input file, then there must be at least carbon 1s, 2s, and 2p orbitals present in the basis. Some basis
Figure 4.6: Increased numbers of Gaussian primitives in an orbital leads to an improvement in approximating the true Slater-type orbital. However, larger basis sets (those including more primitives) are also more computationally expensive.

sets will, however, include orbitals of higher principal number and angular momentum ($3s$, $3p$, $3d$, ...) whether they are occupied or not; doing so often improves the quality of the molecular orbitals $C$ and accuracy of the HF energy$^8$. Because each of these orbital types ($s, p, d, ...$) has angular momentum associated with it, the corresponding integer values of $l$, $m$, and $n$ are also (Eq. 4.19) “plugged in” to the solutions.

In Handout 1, which follows from the foregoing discussion, the readers participate in the derivation of the molecular integrals. In Handout 3, they will implement the solutions in Python code. The reader with little or no programming experience should work through Handout 2 before Handout 3, because it an introduction to the Python programming language, tailored to the integrals and HF programming projects.

Once the material in Handout 1, Handout 2, Handout 3, and Handout 4 (HF implementation) is complete, the reader will have developed the beginnings of an independent quantum chemistry code. In a future article, we will address the more modern methods of molecular integral evaluation, such as the recursion schemes of the Obara–Saika method$^{168}$ and Head-Gordon–Pople method$^{169}$.
Figure 4.7: An example of the structure of the tabulated STO–3G basis set for the carbon atom. Many table formats are found; this is “TurboMole” selected from the drop-down menu (with “Optimized General Contractions”) of the EMSL Basis Set Exchange, with “carbon” highlighted in the web site’s periodic table. The red text was added in this figure to highlight salient features of the table. \( n \) is the principal quantum number.

the quadrature of Dupuis and King\textsuperscript{170}, and others. The present article establishes the fundamental knowledge and skills upon which a discussion of those methods will depend.

Results and Scaffolding Techniques for Instructors

During the summer of 2017, a small sample of students in the authors’ research group completed the central programming project, described in Handout 3. This was preceded by a lecture addressing most of the content present in the main article and parts of the derivations in Handout 1. 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; he did consult online resources for coding guidance. His background was in chemistry and mathematics, and he had some coding experience. The student who completed the project in forty hours was an undergraduate studying chemistry, and had no experience in programming or formal background in mathematics. He required several hours of close assistance from one of the authors. 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 Handout 5), 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 ideas 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 STOs. 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
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 H₂, HeH⁺, HF (hydrogen fluoride), and H₂O are packaged as a .zip file in SU 5.

2. The overlap integrals are the easiest to implement. It may be appropriate to have some students code just the S matrix. (Handout 2 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. 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 2.) 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 2 or Handout 3), where each equation (function) 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 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.
4.3 Handout 1: Derivation of the Molecular Integrals

In the following document, derivations of solutions to the molecular integrals are presented in guided exercise form. Key equations and solutions are boxed as Essential Equations. In order, the four following energy molecular integrals are presented:

1. **Overlap** \((A|B)\). In this section, the necessary specialized notation that will be used throughout all derivations is addressed. Additional details of how two centers \((A, B)\) are reduced to one \((P)\) are provided. In addition to the radial components of both CGTOs \(\phi_A\) and \(\phi_B\), which are treated by the Gaussian Product Theorem, the angular momentum components are derived through the use of standard binomial coefficients. Standard integral formulae, as are found in tables of integrals, are presented for use, allowing for the derivation of the final, code-able expression of \((A|B)\). The student/reader also derives the normalization constant \(N\). Additional discussion of matrix structure, with respect to the atomic orbital basis, is included, and the discussion applies to \(T\) and \(V\) in addition to \(S\). This section ends with a discussion of the difference between minimal and extended basis sets.

2. **Kinetic energy** \((A|\frac{1}{2} \nabla^2|B)\). This brief section leads the reader to the conclusion that the expression for a kinetic energy matrix element is simply a sum of overlap integrals, each term in the summation led by a constant.

3. **Electron-nuclear attraction** \((A|\frac{Z_C}{r_{IC}}|B)\). The presence of an inverse operator \((\frac{Z_C}{r_{IC}}\) being in the denominator) marks a radical departure from \((A|B)\) and \((A|\frac{1}{2} \nabla^2|B)\). Its solution fundamentally depends on a Fourier transformation, which serves to draw \(\frac{r_{IC}}{Z_C}\) out of the denominator. This section introduces the Boys Function, \(F_\nu\), which can only be solved numerically.

4. **Electron-electron repulsion** \((AB|CD)\). Characterized by another inverse operator \((\frac{1}{r_{ij}})\), these molecular integrals entail a solution similar to that of \((A|\frac{Z_C}{r_{IC}}|B)\). While the previous integrals are one-electron in nature, requiring one third center \(P\), the electron-electron repulsion treats two electrons through the use of two third centers, \(P\) and \(Q\). Hence, there is approximately twice as much manipulation as for the electron-nuclear attraction integrals.
Additionally, these matrix elements make up a rank-4 tensor (or four-dimensional matrix) instead of a two-dimensional matrix; the structure of a rank-4 tensor is discussed.

As the student or reader works through the following material, they are encouraged to use a sheet to cover up exercise solutions, which immediately follow the questions and are marked by a thick black line. Most exercises are relatively straightforward, and shouldn’t take more than 15 minutes each to complete.
Overlap integrals

A single matrix element for the overlap of two atomic orbitals \( \phi_A \) and \( \phi_B \) has the general form

\[
(A|B) = \int \phi_A \phi_B \, d\tau ,
\]

where \( d\tau \) is the differential volume element over three dimensions. We employ Cartesian coordinates,

\[
d\tau \mapsto dV \equiv dx \, dy \, dz .
\]

As described in the main article, a single atomic orbital \( \phi_A \) can be represented by linear combinations of Cartesian Gaussian primitives \( \chi_a \),

\[
\phi_A = \sum_{a=1}^{K_a} d_a N_a \chi_a = \sum_{a=1}^{K_a} d_a N_a x_{lA}^A y_{mA}^A z_{nA}^A e^{-\alpha_ar_A^2} ,
\]

where \( r_A^2 = x_A^2 + y_A^2 + z_A^2 \).

In fact, the exact notation for any electron position \( r \), including \( r_A \), is \( \vec{r} \) (which in this case would be \( \vec{r}_A \)). For notational clarity the vector symbol is sometimes only implied, as was the case in the main article. Because \( \vec{r} \) is a vector quantity, then, \( x_A \equiv \vec{r}_A \cdot \vec{e}_x = \vec{r}_A - \vec{A}_x \), the \( x \)-component of the distance between the electron and center \( A \); the simpler notation \( x_A \) is used for notational clarity. See Fig. 1. Also \( x_{lA}^A \) implies \( x_{lA}^A \), as seen in Eq. 4.22.

Thus, Eq. 4.21 becomes

\[
(A|B) = \sum_{a=1}^{K_a} \sum_{b=1}^{K_b} d_a d_b N_a N_b \int \chi_a(\vec{A}, \alpha_a, l_A, m_A, n_A) \chi_b(\vec{B}, \alpha_b, l_B, m_B, n_B) \, dV .
\]

While this form allows us to see that each primitive is normalized and multiplied by its contraction coefficient, we still have a general integral expression that is not code-able. The notation adopted in Eq. 4.23 indicates the functional independence of each primitive on several parameters:

- \( \vec{A} \) is a positon vector which describes the Cartesian coordinates of the atom on which the orbital is centered in terms of \( x, y, \) and \( z \) components (this gives rise to the term “atom-centered orbitals,” which are indeed the basis of our molecular integrals)
Figure 4.8: The molecular coordinate system. \( \vec{r} \) defines the electron’s coordinates with respect to the origin; \( \vec{A} \) is the coordinates of atom \( A \), or atom-centered orbital \( \phi_A \). \( \vec{r}_A \) is the distance between the electron and the atom’s center.

- \( \alpha_a \) is a constant (a basis set parameter, alongside contraction coefficient \( d_a \)) which appears as a factor in the exponent

- \( l, m, \) and \( n \) are angular momentum components with respect to coordinates \( x, y, \) and \( z \)

Expanding out the integral over the Gaussian primitives \( \chi_a \) and \( \chi_b \), we have

\[
\int \chi_A \chi_B dV = \int \frac{x^l_A y^m_A z^n_A e^{-\alpha_a r^2_A} x^l_B y^m_B z^n_B e^{-\alpha_b r^2_B}}{\chi_a} dV.
\]  

(4.24)

Using \( r^2 = x^2 + y^2 + z^2 \), this becomes

\[
\int_0^\infty \int_0^\infty \int_0^\infty x^l_A y^m_A z^n_A e^{-\alpha_a (x^2_A + y^2_A + z^2_A)} x^l_B y^m_B z^n_B e^{-\alpha_b (x^2_B + y^2_B + z^2_B)} dx dy dz.
\]  

(4.25)

Eq. 4.25 can be rearranged on the basis of the separability of the Cartesian coordinates

\[
\int_0^\infty x^l_A x^l_B e^{-\gamma (x^2_A + x^2_B)} dx \int_0^\infty y^m_A y^m_B e^{-\gamma (y^2_A + y^2_B)} dy \int_0^\infty z^n_A z^n_B e^{-\gamma (z^2_A + z^2_B)} dz,
\]  

(4.26)
where

$$\gamma = \alpha_a + \alpha_b .$$

(4.27)

Each of these integrals are analogous, so we will focus on solving for one, the $x$-component.

$$\int_0^\infty x^{l_A} x^{l_B} e^{-\gamma(x_A^2 + x_B^2)} \, dx .$$

(4.28)

It is relatively straightforward to derive the expression for the overlap of $s$ orbitals – that is, when $l_A$ and $l_B$ equal 0, neither having angular momentum.

**Exercise.**

Derive the analytical solution for the overlap of $s$ orbitals $\phi_A$ and $\phi_B$ (non-normalized). First form the Gaussian product of centers $\vec{A}$ and $\vec{B}$, as discussed in the main article. Then, having formed $\tilde{K}$ and the exponential function, $\exp(-\gamma r_P^2)$, where $\gamma = \alpha_a + \alpha_b$, integrate over the $x$-component of this function by using the following equation from a table of integrals.

**Essential Equation**

$$\int_0^\infty e^{-kr^2} = \frac{1}{2} \sqrt{\frac{\pi}{k}}$$

(4.29)

Note that, upon going from coordinates over $\vec{r}_A$ and $\vec{r}_B$ to $\vec{r}_P$, the limits of integration change from $\int_0^\infty$ to $\int_{-\infty}^{\infty}$. This is because, while the distance between an electron and the center of an atom ($\vec{r}_A$) can never be less than zero, this restriction no longer applies to a center between two nuclei. In such a case, the result of using Eq. 4.29 must be multiplied by two.

The solution for the $y$ and the $z$ components will be entirely analogous to the solution for the $x$ component. The results can be easily combined into a single solution over the three Cartesian coordinates. Include the results for the $y$ and $z$, in addition to $x$, components in your answer for a solution over all space in all dimensions.
Solution.

Letting \( l = m = n = 0, \)

\[
\phi_A = x^0 y^0 z^0 e^{-\alpha a r_A^2} = e^{-\alpha a r_A^2},
\]
\[
\phi_B = x^0 y^0 z^0 e^{-\alpha b r_B^2} = e^{-\alpha b r_B^2}.
\]

Forming the Gaussian product of these two orbitals, centered on \( \vec{P} \):

\[
\phi_A \phi_B = e^{-\alpha a r_A^2} e^{-\alpha b r_B^2}
\]
\[
= \exp \left( -\frac{\alpha_a \alpha_b}{\gamma} |\vec{A}\vec{B}|^2 \right) \exp(-\gamma r_P^2)
\]
\[
= \tilde{K}_P \exp(-\gamma r_P^2),
\]

where

\[\vec{A}\vec{B} = \vec{A} - \vec{B}.\]

\(|\vec{A}\vec{B}|^2\) was introduced in the main article, and is simply the distance formula.

\(|\vec{A}\vec{B}|^2 = (\vec{A}_x - \vec{B}_x)^2 + (\vec{A}_y - \vec{B}_y)^2 + (\vec{A}_z - \vec{B}_z)^2.\] (4.30)

Because \( \tilde{K}_P \) is a constant, it can be pulled outside the integral:

\[
\int_{-\infty}^{\infty} \tilde{K}_P \exp(-\gamma r_P^2) \, dV = \tilde{K}_P \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \exp \left[ -\gamma (x_P^2 + y_P^2 + z_P^2) \right] \, dx \, dy \, dz
\]
\[
= \tilde{K}_P \int_{-\infty}^{\infty} e^{-\gamma x_P^2} \, dx \int_{-\infty}^{\infty} e^{-\gamma y_P^2} \, dy \int_{-\infty}^{\infty} e^{-\gamma z_P^2} \, dz.
\]

First solving for the \( x \)-component, we use the standard solution from a table of integrals, shown in Eq. 4.29. Since we are integrating not over \([0, \infty)\), but rather \((\infty, \infty)\), we must multiply the solution by two. Thus,

\[
\int_{-\infty}^{\infty} e^{-\gamma x^2} = \sqrt{\frac{\pi}{\gamma}}.
\] (4.31)
The result will be identical for the $y$ and the $z$ components. Thus,

$$\int \phi_A \phi_B \, dV = \tilde{K} P \int_{-\infty}^{\infty} e^{-\gamma x^2} \, dx \int_{-\infty}^{\infty} e^{-\gamma y^2} \, dy \int_{-\infty}^{\infty} e^{-\gamma z^2} \, dz = \tilde{K} \left( \frac{\pi}{\gamma} \right)^{3/2} . \quad (4.32)$$

[ end of solution ]

The overlap integral is more complicated when one or both orbitals have orbital angular momentum, because this requires taking the product of polynomials (e.g., $x^l_A x^l_B$) in addition to the Gaussian functions.

We will consider the polynomial part of the integral first. Recall that $x_A \equiv (\vec{r}_x - \vec{A}_x)^l_A$. We have been and will continue to use $x$ as shorthand notation for $\vec{r}_x$. Thus, our product of polynomials can be written as

$$x^l_A x^l_B \equiv (x - \vec{A}_x)^l_A (x - \vec{B}_x)^l_B . \quad (4.33)$$

Our objective is to extract the $x$ (i.e., $\vec{r}_x$) from each polynomial, because this is the coordinate being integrated over. By employing the Gaussian product theorem and forming a third center $\vec{P}$ between centers $\vec{A}$ and $\vec{B}$ (see Fig. 4.9),

$$4.145 \vec{P} = \frac{\alpha_a \vec{A} + \alpha_b \vec{B}}{\alpha_a + \alpha_b} , \quad (4.34)$$

then one of the polynomials in Eq. 4.33 can be expanded as

$$(x - \vec{A}_x)^l_A = \left[ (x - \vec{P}_x) + (\vec{P}_x - \vec{A}_x) \right]^l_A = (x_P - \vec{P} \vec{A}_x)^l_A . \quad (4.35)$$

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,

$$(a + b)^n = \sum_{k=0}^{n} \binom{n}{k} a^{n-k} b^k , \quad (4.36)$$

76
Figure 4.9: \( \vec{r}_P \) is the distance between the electron and the newly formed third center \( \vec{r}_P \), which is formed as the Gaussian product of centers \( \vec{A} \) and \( \vec{B} \).

where \( \binom{n}{k} \) is the binomial coefficient meaning “\( n \) choose \( k \)” or

\[
\binom{n}{k} = \frac{n!}{k!(n-k)!},
\]

we can recast Eq. 4.35.

**Exercise.**

Express Eq. 4.35 as a binomial expansion, with a binomial coefficient “\( l_A \) choose \( i \).”

**Solution.**

\[
(x_P - \vec{P}A_x)^{l_A} = \sum_{i=0}^{l_A} \binom{l_A}{i} (\vec{P}A_x)^{l_A-i} (x_P)^i
\]

[ end of solution ]
By taking the product of the binomial expansions of each polynomial, we have

\[
(x_P - \vec{P}A_x)^{l_A}(x_P - \vec{P}B_x)^{l_B} = \sum_{i=0}^{l_A} \binom{l_A}{i} (\vec{P}A_x)^{l_A-i}(x_P)^i \sum_{j=0}^{l_B} \binom{l_B}{j} (\vec{P}B_x)^{l_B-j}(x_P)^j
\]

\[
= \sum_{i=0}^{l_A} \sum_{j=0}^{l_B} \binom{l_A}{i} \binom{l_B}{j} (\vec{P}A_x)^{l_A-i}(\vec{P}B_x)^{l_B-j}(x_P)^{i+j} \quad (4.39)
\]

\[
= \sum_{i=0}^{l_A} \sum_{j=0}^{l_B} \binom{l_A}{i} \binom{l_B}{j} (\vec{P}A_x)^{l_A-i}(\vec{P}B_x)^{l_B-j}(x_P)^{i+j}
\]

Having collected terms \(x^i\) and \(x^j\), and by introducing a coefficient \(c_k\), we can reduce two sums to

\[
(x_P - \vec{P}A_x)^{l_A}(x_P - \vec{P}B_x)^{l_B} = \sum_{k=0}^{l_A+l_B} c_k(l_A,l_B,\vec{P}A_x,\vec{P}B_x)x_P^k \quad (4.40)
\]

Now we define the coefficient \(c_k\) with some generalized variables.

<table>
<thead>
<tr>
<th>Essential Equation</th>
</tr>
</thead>
</table>
| \[
\begin{align*}
   c_k(l, m, a, b) &= \sum_{i=0}^{l} \sum_{j=0}^{m} \binom{l}{i} \binom{m}{j} a^{l-i} b^{m-j} ; \quad i + j = k
\end{align*}
\] |

In Eq. 4.41 the variables for \(c_k\) are generalized because we will be using this coefficient again later with different arguments. This \(c_k\) retains the indices \(i\) and \(j\) that appear in Eq. 4.39, but it must be appreciated that \(c_k\) only evaluated when the sum of \(i\) and \(j\) is equal to \(k\). The notation featuring an equals sign with an exclamation over it is the directive that the coefficient formula is only evaluated when the sum of \(i\) and \(j\) is equal to \(k\). In the parent function to which \(c_k\) belongs, Eq. 4.40, the single summation constrains the sum of \(i\) and \(j\) in the coefficient to being equal to \(k\). For example, if \(l_A + l_B = 2 + 3 = 5\), and we are in the state of summation when \(k = 1\), then when evaluating \(c_k\), either summation variable \(i\) (which has an upper bound of 2) or summation variable \(j\) (which has an upper bound of 3) must be equal to one, and the other to zero. If this condition is not met, the formula is bypassed until the next combination of summations in \(c_k\) is equal to \(k\).

The radial portion of the orbital product \(\phi_A \phi_B\) is analyzed just as it was in an earlier exercise.
Over $x$, $y$, and $z$ components,

\[
(e^{-\alpha_a r_A^2}) \times (e^{-\alpha_B r_B^2}) = \exp\left(\frac{-\alpha_a \alpha_B |A\vec{B}|^2}{\gamma}\right) \exp\left(-\gamma x^2_P\right)
\]

\[
= \tilde{K}_P \exp\left(-\gamma x^2_P\right).
\]

We now couple the results of this analysis of the radial portion of $\phi_A \phi_B$ with our analysis of the angular portion, seen in final form in Eqs. 4.40 and 4.41 for the $x$-component (with analogous expression for $y$ and $z$). Looking at just the $x$-component of their product – with $\tilde{K}_P$ set aside until the end of the derivation, because it is just a constant requiring no further manipulation – they make up an equation we call $S_x$. It can be seen that the general equation for the overlap of the $x$-component of Gaussian primitives (Eq. 4.28) now has the following form:

\[
S_x = \sum_{k=0}^{l_A+l_B} c_k(l_A, l_B, \vec{P}A_x, \vec{P}B_x) \int_{-\infty}^{\infty} x^k \exp\left(-\gamma x^2_P\right) dx
\]

(4.43)

We have succeeded in recasting an integral that initially had been over the two centers $\vec{A}$ and $\vec{B}$ (see Eq. 4.28) to one over the single intermediate center $\vec{P}$. It remains to evaluate the “integral” portion of Eq. 4.43. Because it is a Gaussian integral over a single center, it is actually a standard integral with a known solution that can easily be found in a table of integrals. Several such closely related integrals are shown below.

<table>
<thead>
<tr>
<th>Essential Equation</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\int_{0}^{\infty} x^{2k}e^{-ax^2}dx = \sqrt{\frac{\pi}{a}} \frac{(2k-1)!!}{2^{k+1}a^k} \quad (k \in \mathbb{Z}, a &gt; 0, !! \text{ is the double factorial})$ (4.44)</td>
</tr>
<tr>
<td>$\int_{-\infty}^{\infty} x^{2k}e^{-ax^2}dx = 2 \int_{0}^{\infty} x^{2k}e^{-ax^2}dx$ (4.45)</td>
</tr>
<tr>
<td>$\int_{-\infty}^{\infty} x^k e^{-ax^2}dx = \sqrt{\frac{\pi}{a}} \frac{(2k-1)!!}{(2a)^k} \quad (k \in \mathbb{Z}, a &gt; 0, !! \text{ is the double factorial})$ (4.46)</td>
</tr>
</tbody>
</table>
A double factorial is defined as follows:

\[(2n-1)!! = \begin{cases} 1 \cdot 3 \cdot 5 \cdots (n-2) \cdot n & \text{if } n \text{ is odd} \\ 2 \cdot 4 \cdot 6 \cdots (n-2) \cdot n & \text{if } n \text{ is even} \end{cases}\]

\[(-1)!! = 0!! = 1\]

Exercise.

Insert the tabulated solution in Eq. 4.46 to solve the integral portion of \(S_x\) (Eq. 4.43).

Solution.

Inserting the tabulated solution gives

\[
\int_{-\infty}^{\infty} x^k_p \exp(-\gamma x^2_p) \, dx = \sqrt{\frac{\pi}{\gamma}} \frac{(2k-1)!!}{(2\gamma)^k}. \tag{4.47}
\]

[ end of solution ]

Thus, we have a code-able equation for \(S_x\).

**Essential Equation**

\[
S_x = \sum_{k=0}^{(l_A+l_B)/2} c_{2k}(l_A, l_B, P_A, P_B) \sqrt{\frac{\pi}{\gamma}} \frac{(2k-1)!!}{(2\gamma)^k}. \tag{4.48}
\]

Exercise.

The upper bound of summation and the argument for the \(c_k\) function changed in Eq. 4.144 from 4.43. This has the effect of passing in to the general \(c_k\) function only even integers of \(k\). Why should only even integers be evaluated? (Hint: In the integral of Eq. 4.43, we are integrating not over \([0, \infty)\), but \((-\infty, \infty)\). Consider this with respect to the parity of the integrand – that is, whether the power function \(x^k\) is even or odd, as defined by whether \(k\) is even or odd.)

Solution.

If the exponent \(k\) in the integral of \(S_x\) (Eq. 4.47) is an odd number, then integration over it returns zero. If \(k\) is an even function, then the integral will be non-zero. Hence, it is only worthwhile evaluating even values of \(k\).
Strictly speaking, we could leave the summation and arguments to $c_k$ function as they were in Eq. 4.43, but this would mean spending precious computational power on terms that we know mathematically will be zero.

[ end of solution ]

With a solution for $S_x$ (Eq. 4.144), analogous expressions for $S_y$ and $S_z$, and by re-including $\vec{K}_P$, we have a code-able equation for the overlap integral $(A|B)$.

**Essential Equation**

\[
(A|B) = \sum_{a=1}^{K_a} \sum_{b=1}^{K_b} d_a d_b N_a N_b \times \exp\left(-\frac{\alpha_a \alpha_b |\vec{AB}|^2}{\gamma} \right) \times S_x \times S_y \times S_z \quad (4.49)
\]

The formula for $|\vec{AB}|^2$ is found in Eq. 4.30. Recalling that the contraction coefficients $(d_a, d_b)$ and exponential factors $(\alpha_a, \alpha_b)$ are tabulated basis set values, then the only remaining factor of Eq. 4.49 to calculate is the normalization constant, $N$. The following formula is used.

**Essential Equation**

\[
N(\alpha, l, m, n) = \left[ \frac{2\alpha}{\pi} \right]^{3/2} \left( \frac{(4\alpha)^{l+m+n}}{(2l-1)!(2m-1)!(2n-1)!} \right)^{1/2} \quad (4.50)
\]
Exercise.

Derive the normalization constant $N$. Begin by considering the overlap integral of primitive $\chi_a$,

$$\chi_a = N x^l y^m z^n e^{-\alpha r^2}$$

with itself, where $N$ is what we must solve for after integration. That is, for

$$\int \chi_a \chi_a \, dx \, dy \, dz = 1,$$

which is expanded as

$$N^2 \int_0^\infty \int_0^\infty \int_0^\infty (x^l y^m z^n e^{-\alpha r^2})(x^l y^m z^n e^{-\alpha r^2}) \, dx \, dy \, dz,$$

solve for $N$. (Hint: Use Eq. 4.44 once per coordinate.)

Solution.

Before we present the solution to this problem, recall that typically several Gaussian primitives are summed (“contracted”) to represent one orbital $\phi_A$, but integrals must be evaluated with respect to individual primitives. Hence, the results of each of these integral evaluations are then summed.

Now,

$$\int \chi_a \chi_a \, dV = 1$$

Thus,

$$\int \chi_a \chi_a \, dV = \int_0^\infty (N x^l y^m z^n e^{-\alpha r^2})(N x^l y^m z^n e^{-\alpha r^2}) \, dV$$

$$= N^2 \int_0^\infty x^{2l} y^{2m} z^{2n} e^{-2\alpha r^2} \, dV.$$ (4.52)

Recalling that

$$r^2 = x^2 + y^2 + z^2,$$ (4.53)
then

\[ N^2 \int_0^\infty x^{2l}y^{2m}z^{2n}e^{-2\alpha x^2} \, dV = N^2 \int_0^\infty \int_0^\infty \int_0^\infty x^{2l}y^{2m}z^{2n}e^{-2\alpha(x^2+y^2+z^2)} \, dx \, dy \, dz \]

\[ = N^2 \int_0^\infty x^{2l}e^{-2\alpha x^2} \, dx \int_0^\infty y^{2m}e^{-2\alpha y^2} \, dy \int_0^\infty z^{2n}e^{-2\alpha z^2} \, dz. \]  

(4.54)

(Here the integration limits are 0 to infinity, because the coordinates are with respect to atomic centers, not third, Gaussian product centers.) By employing Eq. 4.44

\[ \int_0^\infty x^{2l}e^{-2\alpha x^2} \, dx = \frac{(2l-1)!!\sqrt{\pi}}{(4\alpha)^l\sqrt{2\alpha}}. \]  

(4.55)

By analogy for the remaining terms, we have

\[ \int_0^\infty x^{2l}e^{-2\alpha x^2} \, dx \int_0^\infty y^{2m}e^{-2\alpha y^2} \, dy \int_0^\infty z^{2n}e^{-2\alpha z^2} \, dz = N^2 \times \frac{(2l-1)!!\sqrt{\pi}}{(4\alpha)^l\sqrt{2\alpha}} \times \frac{(2m-1)!!\sqrt{\pi}}{(4\alpha)^m\sqrt{2\alpha}} \times \frac{(2n-1)!!\sqrt{\pi}}{(4\alpha)^n\sqrt{2\alpha}} \]

\[ = N^2 \times \left[ \left( \frac{\pi}{2\alpha} \right)^{3/2} \frac{(2l-1)!!(2m-1)!!(2n-1)!!}{(4\alpha)^{l+m+n}} \right] = 1. \]

(4.56)

Rearranging the last line of Eq. 4.56 to solve for \( N \) simply requires multiplying both sides by the inverse of the expression in square brackets, and then taking the square root of both sides. This gives us our normalization constant \( N \).

\[ \frac{1}{N} = \left[ \left( \frac{2\alpha}{\pi} \right)^{3/2} \frac{(4\alpha)^{l+m+n}}{(2l-1)!!(2m-1)!!(2n-1)!!} \right]^{1/2} \]

(4.57)

[ end of solution ]

Eq. 4.49 corresponds to one matrix element – one “box” of \( 7 \times 7 \) overlap matrix for \( \text{H}_2\text{O} \) shown in Fig. 4.10, where \( \phi_A = |A\rangle = |O1s\rangle \), or \( |O2s\rangle \), or \( |O2p_x\rangle \), or ... and \( \phi_B = |O1s\rangle \), or \( |O2s\rangle \), or \( |O2p_x\rangle \), etc. Among the \( 7 \times 7 = 49 \) elements, we have the overlap of oxygen’s \( 2s \) orbital with hydrogen’s \( 1s \) orbital. We also have the other hydrogen’s \( 1s \) orbital overlapping with oxygen’s \( 2p_x \) orbital, and also the oxygen’s \( 2p_y \) orbital overlapping with its \( 1s \) orbital – and so on. Each of these possible overlaps constitute a single element of the overlap integral matrix, which we term \( S \); individual overlaps in \( (A|B) \) are shown in Fig. 4.11. It is worth pointing out that an orbital’s overlap with itself will, when normalized, be exactly equal to 1. In addition to factors of symmetry
(s vs. p, etc.) and differences in orbital energy, the overlap integrals of atomic orbitals that are separated by greater distance will have smaller values. This is demonstrated in Fig. 4.12.

<table>
<thead>
<tr>
<th></th>
<th>O1s</th>
<th>O2s</th>
<th>O2p_x</th>
<th>O2p_y</th>
<th>O2p_z</th>
<th>H_1s</th>
<th>H_2s</th>
</tr>
</thead>
<tbody>
<tr>
<td>O1s</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>O2s</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>O2p_x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>O2p_y</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>O2p_z</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>H_1s</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>H_2s</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 4.10: The 7 × 7 overlap integral matrix S for water in the STO-3G orbital basis.

<table>
<thead>
<tr>
<th></th>
<th>O1s</th>
<th>O2s</th>
<th>O2p_x</th>
<th>O2p_y</th>
<th>O2p_z</th>
<th>H_1s</th>
<th>H_2s</th>
</tr>
</thead>
<tbody>
<tr>
<td>O1s</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>O2s</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>O2p_x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>O2p_y</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>O2p_z</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>H_1s</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>H_2s</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 4.11: The overlap integral matrix S showing explicitly every integral (A|B) that must be evaluated.

This concludes our discussion of how one matrix element of the overlap integral matrix S is evaluated.

An Aside on Minimal vs. Extended Basis Sets

This is a good time to mention the idea of a minimal basis set versus other kinds of basis sets. The minimal basis set corresponds most closely to the electron configurations we learn about in freshman chemistry. For example, while the electron configuration of a carbon atom is

\[ 1s^2 2s^2 2p^2, \]  

our minimal basis configuration would consist of the set of all 1s, 2s and 2p spatial orbitals making up carbon’s shells:

\[ 1s 2s 2p_x 2p_y 2p_z. \]
<table>
<thead>
<tr>
<th></th>
<th>(\text{O}_1\text{s})</th>
<th>(\text{O}_2\text{s})</th>
<th>(\text{O}_2\text{p}_x)</th>
<th>(\text{O}_2\text{p}_y)</th>
<th>(\text{O}_2\text{p}_z)</th>
<th>(\text{H}_a\text{1s})</th>
<th>(\text{H}_b\text{1s})</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\text{O}_1\text{s})</td>
<td>1.00000</td>
<td>0.23670</td>
<td>0.00000</td>
<td>0.00000</td>
<td>0.00000</td>
<td>0.00678</td>
<td>0.00678</td>
</tr>
<tr>
<td>(\text{O}_2\text{s})</td>
<td>0.23670</td>
<td>1.00000</td>
<td>0.00000</td>
<td>0.00000</td>
<td>0.00000</td>
<td>0.09883</td>
<td>0.09883</td>
</tr>
<tr>
<td>(\text{O}_2\text{p}_x)</td>
<td>0.00000</td>
<td>0.00000</td>
<td>1.00000</td>
<td>0.00000</td>
<td>1.00000</td>
<td>0.00000</td>
<td>0.00000</td>
</tr>
<tr>
<td>(\text{O}_2\text{p}_y)</td>
<td>0.00000</td>
<td>0.00000</td>
<td>0.00000</td>
<td>1.00000</td>
<td>0.00000</td>
<td>0.08159</td>
<td>-0.08159</td>
</tr>
<tr>
<td>(\text{O}_2\text{p}_z)</td>
<td>0.00000</td>
<td>0.00000</td>
<td>0.00000</td>
<td>0.00000</td>
<td>1.00000</td>
<td>-0.06843</td>
<td>-0.06843</td>
</tr>
<tr>
<td>(\text{H}_a\text{1s})</td>
<td>0.00678</td>
<td>0.09883</td>
<td>0.00000</td>
<td>0.08159</td>
<td>-0.06843</td>
<td>1.00000</td>
<td>0.02420</td>
</tr>
<tr>
<td>(\text{H}_b\text{1s})</td>
<td>0.00678</td>
<td>0.09883</td>
<td>0.00000</td>
<td>-0.08159</td>
<td>-0.06843</td>
<td>0.02420</td>
<td>1.00000</td>
</tr>
</tbody>
</table>

**Figure 4.12:** The outcome of evaluating every element of the overlap integrals matrix \(S\) in the minimal STO-3G basis, using Eq. 4.49 for each element, and the .xyz coordinates (units Bohr) shown below it.

In other words we provide functions for *every* orbital of a shell that is occupied in the electron configuration. This means that the atomic orbital configuration of boron looks the same as it does for carbon \((1s \ 2s \ 2p_x \ 2p_y \ 2p_z)\), in spite of having a different electron configuration. Even lithium has this orbital configuration, in spite of not having any \(2p\) orbitals occupied in the ground state. Early investigators found that including terms for the complete shell, instead of just the occupied orbitals, gave more accurate results. (Note that the basis set parameters \(d_p\) and \(\alpha_p\) do differ between atoms, even though all second-row atoms have the same minimal basis *configuration* shown in Eq. 4.59.)
Exercise.

Write the electron configurations and minimal basis sets of:

1. Lithium
2. Nitrogen
3. Oxygen
4. Chlorine
5. Vanadium
6. Krypton

Solution.

1. 1s\(^{2}\) 2s; 1s 2s 2p\(_{x}\) 2p\(_{y}\) 2p\(_{z}\)

2. 1s\(^{2}\) 2s\(^{2}\) 2p\(^{3}\); 1s 2s 2p\(_{x}\) 2p\(_{y}\) 2p\(_{z}\)

3. 1s\(^{2}\) 2s\(^{2}\) 2p\(^{4}\); 1s 2s 2p\(_{x}\) 2p\(_{y}\) 2p\(_{z}\)

4. 1s\(^{2}\) 2s\(^{2}\) 2p\(^{6}\) 3s\(^{2}\) 3p\(^{5}\); 1s 2s 2p\(_{x}\) 2p\(_{y}\) 2p\(_{z}\) 3s 3p\(_{x}\) 3p\(_{y}\) 3p\(_{z}\)

5. 1s\(^{2}\) 2s\(^{2}\) 2p\(^{6}\) 3s\(^{2}\) 3p\(^{6}\) 4s\(^{2}\) 3d\(^{3}\); 1s 2s 2p\(_{x}\) 2p\(_{y}\) 2p\(_{z}\) 3s 3p\(_{x}\) 3p\(_{y}\) 3p\(_{z}\) 4s 3d\(_{xz}\) 3d\(_{yz}\) 3d\(_{xy}\) 3d\(_{x^{2}−y^{2}}\)

6. 1s\(^{2}\) 2s\(^{2}\) 2p\(^{6}\) 3s\(^{2}\) 3p\(^{6}\) 4s\(^{2}\) 3d\(^{10}\) 4p\(^{6}\); 1s 2s 2p\(_{x}\) 2p\(_{y}\) 2p\(_{z}\) 3s 3p\(_{x}\) 3p\(_{y}\) 3p\(_{z}\) 4s 3d\(_{xz}\) 3d\(_{yz}\) 3d\(_{xy}\) 3d\(_{x^{2}−y^{2}}\) 4p\(_{x}\) 4p\(_{y}\) 4p\(_{z}\)

[ end of solution ]

The other kind of basis set is the broadly defined extended basis set, which is any kind of basis set other than minimal. This includes split-valence basis sets like 3-21G. “Split valence” implies that core orbitals (closer to the nucleus) are treated with one contracted Gaussian-type function (CGTF), while valence orbitals are treated with more than one. The notation is telling: Core orbitals are represented by one CGTF that is a contraction of three primitives, while valence
orbitals are represented by two CGTFs: one is a contraction of two primitives, and the other has just one primitive.

Unfortunately, when we discuss extended basis sets, we can no longer equate the term “orbital” with CGTF (though be warned that some writers do!). With the 3-21G basis set, the actual $2p_x$ orbital of the carbon atom, which is a valence orbital, is now described in terms of two CGTFs. Throughout this paper, however, and the programming project, we will be using a minimal basis set, so it is appropriate to use the terms interchangeably – i.e., a CGTF is our direct mathematical representation of an orbital (see Eq. 4.22). During the early stages of coming to grips with molecular integrals it provides useful physical intuition to think of it this way. Nevertheless, it is important that the reader be aware that, for extended basis sets, it’s no longer appropriate or meaningful to refer to contracted Gaussian-type functions as orbitals.

Exercise.

In the 3-21G basis set, the hydrogen 1s orbital is considered a valence orbital, so it is treated with two CGTF’s; the oxygen 1s, by contrast, is treated with just one CGTF. What would be the dimensions of an overlap matrix for H₂O, represented by a 3-21G basis set?

Solution.

- Two CGTF’s for each hydrogen 1s orbital (considered valence)
- One CGTF for the oxygen 1s core orbital
- Two CGTF’s for the $2s$, $2p_x$, $2p_y$, and $2p_z$ orbitals

Thus, $2 + 2 + 1 + (2 \times 4) = 13$, producing a $13 \times 13$ matrix. See Fig. 4.13.

[end of solution]
<table>
<thead>
<tr>
<th>(O₁s)</th>
<th>(O₂s')</th>
<th>(O₂pₓ)</th>
<th>(O₂pᵧ)</th>
<th>(O₂s‘)</th>
<th>(O₂pₓ')</th>
<th>(O₂pᵧ')</th>
<th>(O₂s‘')</th>
<th>(O₂pₓ'')</th>
<th>(O₂pᵧ'')</th>
<th>(Hₐ₁s)</th>
<th>(Hₐ₁s')</th>
<th>(Hₐ₁s)</th>
<th>(Hₐ₁s')</th>
</tr>
</thead>
<tbody>
<tr>
<td>(O₁s)</td>
<td>(O₂s')</td>
<td>(O₂pₓ)</td>
<td>(O₂pᵧ)</td>
<td>(O₂s‘)</td>
<td>(O₂pₓ')</td>
<td>(O₂pᵧ')</td>
<td>(O₂s‘')</td>
<td>(O₂pₓ'')</td>
<td>(O₂pᵧ'')</td>
<td>(Hₐ₁s)</td>
<td>(Hₐ₁s')</td>
<td>(Hₐ₁s)</td>
<td>(Hₐ₁s')</td>
</tr>
</tbody>
</table>

Figure 4.13: The basic structure of the water overlap integral matrix \( S \) in the 3-21G basis is a 13 x 13 matrix instead of 7 x 7 as in STO-3G. Valence orbitals are split into two distinct contracted Gaussian-type functions (CGTFs), each having their own set of matrix elements. Clearly, many more terms must be valuated than for the matrix in the STO-3G basis, and so computation with a 3-21G basis set will take longer than those a STO-3G basis set.

**Kinetic energy integrals**

The general form of a kinetic energy matrix element is \((A| - \frac{1}{2} \nabla^2|B)\),

\[
\sum_{a=1}^{K_a} \sum_{b=1}^{K_b} d_a d_b N_a N_b \int \int \int \chi_a(\vec{A}, \alpha_a, l_A, m_A, n_A) \left[ -\frac{1}{2} \nabla^2 \chi_b(\vec{B}, \alpha_B, l_B, m_B, n_B) \right] dx dy dz \quad (4.60)
\]

where

\[
\nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2} \quad (4.61)
\]

**Exercise.**

Show the effect of the \( x \)-component of \( \nabla^2 \) acting on the primitive Gaussian \( \chi_b \), where

\[
\chi_b = x^l y^m z^n e^{-\alpha_b r_B^2} = x^l y^m z^n e^{-\alpha_b (x_B^2 + y_B^2 + z_B^2)}
\]

For now, disregard \( N \), the normalization constant.
Solution.

\[
\frac{\partial}{\partial x} \left( \frac{\partial}{\partial x} \chi_b \right) = \frac{\partial}{\partial x} \left( l_B x^{l_B-1} y^{m_B} z^{n_B} e^{-\alpha_B x^2} - 2\alpha_B x^{l_B+1} y^{m_B} z^{n_B} e^{-\alpha_B x^2} \right)
\]

\[
= \left[ (l_B - 1) l_B x^{l_B-2} y^{m_B} z^{n_B} - x l_B y^{m_B} z^{n_B} 2\alpha_B \left( 2l_B - 2\alpha_B x^2 + 1 \right) \right] e^{-\alpha_B x^2} \tag{4.62}
\]

\[
= l_B (l_B - 1) \chi_b (\vec{B}, \alpha_b, l_B - 2, m_B, n_B) - (4\alpha_B l_B + 1) \chi_b (\vec{B}, \alpha_b, l_B, m_B, n_B)
\]

\[
+ 4\alpha_b^2 \chi_b (\vec{B}, \alpha_b, l_B + 2, m_B, n_B)
\]

[ end of solution ]

Exercise.

Multiply the result of Eq. 4.62 on the left by \( \chi_a \) and integrate over \( dx \). Do not attempt to solve the integral. Collect terms by integrals featuring \( \chi_b \) with different angular momenta (\( l_B, l_B - 2, l_B + 2 \)).

Solution.

\[
\int \chi_a \left( -\frac{1}{2} \nabla^2 \right) \chi_b \, dx = \alpha_b (2l_B + 1) \int \chi_a (\vec{A}, \alpha_a, l_A, m_A, n_A) \chi_b (\vec{B}, \alpha_b, l_B, m_B, n_B) \, dx
\]

\[
-2\alpha_b^2 \int \chi_a (\vec{A}, \alpha_a, l_A, m_A, n_A) \chi_b (\vec{B}, \alpha_b, l_B + 2, m_B, n_B) \, dx \tag{4.63}
\]

\[
-\frac{1}{2} l_B (l_B - 1) \int \chi_a (\vec{A}, \alpha_a, l_A, m_A, n_A) \chi_b (\vec{B}, \alpha_b, l_B - 2, m_B, n_B) \, dx
\]

[ end of solution ]

Exercise.

Repeat the previous exercise for dimensions \( y \) and \( z \). Add together the results (over dimensions \( x \), \( y \), and \( z \)) and collect terms by the three common coefficients

\[
[2(l_B + m_B + n_B) + 3], \quad -2\alpha_b^2, \quad -\frac{1}{2}
\]

89
Finally, lead this multiple integral (over dimensions $x$, $y$, and $z$) expression by the primitive “contraction”

$$
\sum_{a=1}^{K_a} \sum_{b=1}^{K_b} d_a d_b N_a N_b
$$

What do you notice about the result?

**Solution.**

<table>
<thead>
<tr>
<th>Essential Equation</th>
</tr>
</thead>
</table>
| $(|A| - \frac{1}{2} \nabla^2 |B\rangle = \sum_{a=1}^{K_a} \sum_{b=1}^{K_b} d_a d_b N_a N_b \int \int \int \chi_a(\vec{A}, \alpha_a, l_A, m_A, n_A) \left\{ \alpha_b [2(l_B + m_B + n_B) + 3] \chi_b(\vec{B}, \alpha_b, l_B, m_B, n_B) - 
2 \alpha_b^2 \left[ \chi_b(\vec{B}, \alpha_b, l_B + 2, m_B, n_B) + \chi_b(\vec{B}, \alpha_b, l_B, m_B + 2, n_B) + \chi_b(\vec{B}, \alpha_b, l_B, m_B, n_B + 2) \right] - 
\frac{1}{2} \left[ l_B(l_B - 1) \chi_b(\vec{B}, \alpha_b, l_B - 2, m_B, n_B) + m_B(m_B - 1) \chi_b(\vec{B}, \alpha_b, l_B - 2, m_B, n_B) + n_B(n_B - 1) \chi_b(\vec{B}, \alpha_b, l_B, m_B, n_B - 2) \right] \right\} dx dy dz

(4.64)

The entire matrix element for kinetic energy, Eq. 4.64, is effectively just a sum of seven different overlap integrals, with coefficients. By coupling Eq. 4.64 with Eq. 4.49, you have a fully code-able expression for the kinetic energy integral matrix element!

[ end of solution ]
**Electron-nuclear attraction integrals**

The electron-nuclear attraction integral is our first of two potential energy (Coulomb) integrals. But like the overlap and kinetic energy integrals we’ve already dealt with, the electron-nuclear attraction integrals are also two-center integrals (center of $\phi_A$ and center of $\phi_B$). As the name implies, here we are dealing with the potential energy associated with the attraction of an electron to nuclei. The operator is $(\vec{r}_{iC})^{-1}$ where $\vec{r}_{iC} = \vec{r}_i - \vec{C}$, the distance between electron $i$ and nucleus $C$. While we are only dealing with two centers, and hence form two-dimensional matrices, we must form $n$ two-dimensional integrals, one for each nucleus (i.e., atom). Once each matrix is computed, we simply add up these matrices into one two-dimensional matrix $V$:

$$V = \sum_{C} V_C = V_1 + V_2 + \ldots V_n \quad (4.65)$$

The general form of a matrix element belonging to a matrix $V_C$ is

$$(A\mid -\frac{Z_C}{\vec{r}_{iC}} \mid B) = \int \int \phi_A \left( -\frac{Z_C}{\vec{r}_{iC}} \right) \phi_B \, dV_i$$

$$= \sum_{a=1}^{K_a} \sum_{b=1}^{K_b} d_a d_b N_a N_b \int \int \chi_a(\vec{A}, \alpha_a, l_A, m_A, n_A) \left( -\frac{Z_C}{\vec{r}_{iC}} \right) \chi_b(\vec{B}, \alpha_b, l_B, m_B, n_B) \, dV_i \quad (4.66)$$

where we now contend with a third index, $C$. $Z_C$ is a constant, the atomic number of atom $C$. The part of Eq. 4.66 to solve is $V$, a multiple integral over Gaussian primitives $\chi_a$ and $\chi_b$ with the potential energy operator $(-Z_C/\vec{r}_{iC})$.

Spelling out $V$ more explicitly,

$$V = \int_{0}^{\infty} \int_{0}^{\infty} \int_{0}^{\infty} x^{l_A} y^{m_A} z^{n_A} e^{-\alpha_a r_A^2} \left( -\frac{Z_C}{\vec{r}_{iC}} \right) x^{l_B} y^{m_B} z^{n_B} e^{-\alpha_b r_B^2} \, dV_i$$

$$dV_i = dx \, dy \, dz. \quad (4.67)$$
Rearranging by Cartesian coordinates,

\[ V = \int_0^\infty \int_0^\infty \int_0^\infty x^l A x^l B y^m A y^m B z^n A z^n B e^{-\alpha_a r^2_A e^{-\alpha_b r^2_B}} \left( -\frac{Z_C}{\vec{r}_{iC}} \right) \text{d}V_i \]  

(4.68)

The Gaussian product theorem will now serve to reduce the number of centers in our problem from two (\( \vec{A}, \vec{B} \)) to one (\( \vec{P} \)), greatly simplifying the mathematics. We begin with the radial part.

Exercise.

Calculate the Gaussian product for the radial portion of primitives \( \chi_a \) and \( \chi_b \) (each contributing to \( \phi_A \) and \( \phi_B \), respectively) where the radial portion of \( \chi_a \) is

\[ \exp(-\alpha_a r^2_A) \]

and the radial portion of \( \chi_b \) has the same form by analogy. Substitute \( \alpha_a + \alpha_b \) for \( \gamma \).

Solution.

\[ \exp(-\alpha_a r^2_A) \exp(-\alpha_b r^2_B) = \exp\left(-\frac{\alpha_a \alpha_b |\vec{A}\vec{B}|^2}{\gamma}\right) \exp(-\gamma r^2_P) \]

(4.69)

\[ = \tilde{K}_P \exp(-\gamma r^2_P) \]

\[ \gamma = \alpha_a + \alpha_b, \quad \vec{A}\vec{B} = \vec{A} - \vec{B} \]

\[ \vec{P} = \frac{\alpha_a \vec{A} + \alpha_b \vec{B}}{\gamma}, \quad \vec{r}_P = \vec{r}_i - \vec{P}, \quad x_P = x_i - \vec{P}_x \]

Note that \( x_P \) is the \( x \)-component of \( \vec{r}_P \), and that \( x_i \) is the \( x \)-component of \( \vec{r}_i \). In most equations \( \vec{r}_P \) is shown simply as \( r_P \) to reduce notational clutter.

[ end of solution ]

Now we subject the angular part of the integral to the Gaussian product theorem.
Exercise.

Calculate the $x$-component of the product of angular components of primitives $\chi_a$ and $\chi_b$.

$$x^l_A x^l_B$$

Do the same for the $y$ and $z$ components.

Solution. Following the example of Eqs. 4.33, 4.35, and 4.40,

$$x^l_A x^l_B = \sum_{l=0}^{l_A+l_B} c_l(l_A, l_B, P^A_x, P^B_x) x^l_P$$

$$y^m_A y^m_B = \sum_{m=0}^{m_A+m_B} c_m(m_A, m_B, P^A_y, P^B_y) y^m_P$$

$$z^n_A z^n_B = \sum_{n=0}^{n_A+n_B} c_n(n_A, n_B, P^A_z, P^B_z) z^n_P.$$
longer a denominator term. We begin by recasting $\vec{r}_{iC}$, which is the notation for distance between electron $i$ and nucleus of atom $C$, in terms of a third center $\vec{P}$:

$$\vec{r}_{iC} = \vec{r}_i - \vec{C} = (\vec{r}_i - \vec{P}) + (\vec{P} - \vec{C}) = \vec{r}_P + \vec{P}\vec{C}, \quad (4.71)$$

where $\vec{P}\vec{C} = \vec{P} - \vec{C}$, the distance between atom $C$ and the third center $\vec{P}$.

We will perform this integral transformation with a Fourier transform. For an inverse vector (such as our operator) this Fourier transform has the following general form.

<table>
<thead>
<tr>
<th>Essential Equation</th>
</tr>
</thead>
<tbody>
<tr>
<td>[ \frac{1}{\vec{r}} = \frac{1}{2\pi^2} \int \frac{1}{k^2} \exp(i\vec{k} \cdot \vec{r}) , d\vec{k} ] (4.72)</td>
</tr>
</tbody>
</table>

The study of Fourier transforms is a deep one, but we focus on its essential purpose, which as employed here is clear: The transform serves to draw the distance vector out of the denominator, and make the expression more mathematically tractable by passing it into the numerator as part of an exponential function. Unfortunately, we must do so at the price of a new integral and with the introduction of the variable coordinates $\vec{k}$.

$\vec{r}_{iC}$, in the argument of an exponential function, must now be integrated over the “transformation” variable $\vec{k} = \vec{k}_x + \vec{k}_y + \vec{k}_z$. The vector $\vec{k}$ is a vector with Cartesian components, so that the differential volume element of the vector is $d\vec{k} = d\vec{k}_x \, d\vec{k}_y \, d\vec{k}_z$. Take note: the imaginary number $i$ is not the same thing as the letter $i$, which has been used up to this point to represent an electron (and will also be used separately later as an index in summations, because there aren’t enough indices to go around!).
Exercise.

Use the Fourier transform of Eq. 4.72 to transform the electron-nuclear attraction operator of Eq. 4.71.

Solution.

\[
\frac{1}{\vec{r}_i C} = \frac{1}{2\pi^2} \int \frac{1}{\vec{k}^2} \exp \left[ i \vec{k} \cdot (\vec{r}_i C) \right] d\vec{k}
\]

\[
= \frac{1}{2\pi^2} \int \frac{1}{\vec{k}^2} \exp \left[ i \vec{k} \cdot (\vec{PC} + \vec{r}_P) \right] d\vec{k}
\]

\[
= \frac{1}{2\pi^2} \int \frac{1}{\vec{k}^2} \exp \left( i \vec{k} \cdot \vec{PC} \right) \exp \left( i \vec{k} \cdot \vec{r}_P \right) d\vec{k} \tag{4.73}
\]

[ end of solution ]

Exercise.

Substitute the result of Eq. 4.73 into \( V_{RI} \), the last line of Eq. 4.70. In turn, decompose the terms \( \exp \left( i \vec{k} \cdot \vec{r}_P \right) \) and \( \exp \left( -\gamma r_P^2 \right) \) into their Cartesian coordinates, i.e.,

\[
\exp \left( i \vec{k} \cdot \vec{r}_P \right) = \exp \left( i \vec{k}_x \cdot x_P \right) \exp \left( i \vec{k}_y \cdot y_P \right) \exp \left( i \vec{k}_z \cdot z_P \right)
\]

\[
\exp \left( -\gamma r_P^2 \right) = \exp \left( -\gamma x_P^2 \right) \exp \left( -\gamma y_P^2 \right) \exp \left( -\gamma z_P^2 \right)
\]

Leave \( \exp \left( i \vec{k} \cdot \vec{PC} \right) \) unchanged.

Finally, rearrange the expression for \( V_{RI} \) by Cartesian coordinates, and collect exponential functions containing the same Cartesian coordinate into one exponential function, while still leaving \( \exp \left( i \vec{k} \cdot \vec{PC} \right) \) as an independent term. For example,

\[
\exp \left( -a x^2 \right) \exp (b x) = \exp \left( -a x^2 + b x \right).
\]
Solution.

\[ V_{RI} = \frac{1}{2\pi^2} \int \frac{1}{k^2} \exp \left( i \vec{k} \cdot \vec{PC} \right) \exp \left( i \vec{k} \cdot \vec{r}_P \right) d\vec{k} \int x_P y_P^m z_P^n \exp \left( -\gamma x_P^2 \right) dV \]
\[ = \frac{1}{2\pi^2} \int \frac{1}{k^2} \exp \left( i \vec{k} \cdot \vec{PC} \right) \exp \left( i \vec{k}_x \cdot x_P \right) \exp \left( i \vec{k}_y \cdot y_P \right) \exp \left( i \vec{k}_z \cdot z_P \right) d\vec{k} \]
\[ \times \int x_P y_P^m z_P^n \exp \left( -\gamma x_P^2 \right) \exp \left( -\gamma y_P^2 \right) \exp \left( -\gamma z_P^2 \right) dV \]
\[ = \frac{1}{2\pi^2} \int \frac{1}{k^2} \exp \left( i \vec{k} \cdot \vec{PC} \right) \]
\[ \times \int x_P y_P^m z_P^n \exp \left( -\gamma x_P^2 + i \vec{k}_x x_P \right) \exp \left( -\gamma y_P^2 + i \vec{k}_y y_P \right) \exp \left( -\gamma z_P^2 + i \vec{k}_z z_P \right) dV \]
\[ = \frac{1}{2\pi^2} \int \frac{1}{k^2} \exp \left( i \vec{k} \cdot \vec{PC} \right) d\vec{k} \]
\[ \times \int x_P^l \exp \left( -\gamma x_P^2 + i \vec{k}_x x_P \right) dx_P \int y_P^m \exp \left( -\gamma y_P^2 + i \vec{k}_y y_P \right) dy_P \int z_P^n \exp \left( -\gamma z_P^2 + i \vec{k}_z z_P \right) dz_P \]

\[ \text{(4.74)} \]

[end of solution]

We now have four integrals to solve: The terms \( V_{ix}^x, V_{im}^y, \text{ and } V_{in}^z \) are identical in form, distinguished by variable and coordinate. To solve these integrals, of the form

\[ V_{ix}^x = \int_{-\infty}^{\infty} x_P^l \exp \left( -\gamma x_P^2 + i \vec{k}_x x_P \right) dx_P, \quad (4.75) \]

we once again look to a table of integrals.

### Essential Equation

\[ \int_{-\infty}^{\infty} t^n \exp(-at^2 + ibt) \, dt = i^n n! \left( \frac{\pi}{a} \right)^{1/2} \left( \frac{1}{2\sqrt{a}} \right)^n \exp(-c^2) \times \sum_{j=0}^{[n/2]} \frac{(-1)^j}{j!} \left( \frac{(2c)^{n-2j}}{(n-2j)!} \right) \quad (4.76) \]

\[ c = \frac{b}{2\sqrt{a}} \]
Exercise.

Let $\epsilon = (4\gamma)^{-1}$, let $r$ be the summation index (not to be confused with the distance vector), and find the solution to Eq. 4.75 by using Eq. 4.76.

At the end, make the substitution

$$
\left( \frac{4\pi}{\epsilon} \right)^{1/2} = \left( \frac{\pi}{\gamma} \right)^{1/2},
$$

while leaving the other $\epsilon$ terms unchanged. This will prove useful later.

Solution.

$$
V_x^l = i^ll! \left( \frac{\pi}{\gamma} \right)^{1/2} \epsilon^{l/2} \exp(-\epsilon k_x^2) \times \sum_{r=0}^{[l/2]} \frac{(-1)^r (2\sqrt{\epsilon k_x})^{l-2r}}{r!(l-2r)!}
$$

(4.77)

[end of solution]

Exercise.

Calculate the product $V_x^l \times V_y^m \times V_z^n$, and collect terms with powers of $l$, $m$, and $n$ as much as possible. Then, collect the spatially decomposed exponential functions into one $\exp(-\epsilon k^2)$ term.
Solution.

\[ V_l^x \times V_m^y \times V_n^z = \begin{pmatrix} \pi \over \gamma \end{pmatrix}^{3/2} l!m!n!e^{(l+m+n) - \frac{(l+m+n)}{2}} \]

\[ \times \sum_{r=0}^{[l/2]} \sum_{s=0}^{[m/2]} \sum_{t=0}^{[n/2]} \frac{(-1)^{(r+s+t)}(2\sqrt{\epsilon})^{(l+m+n)-2(r+s+t)}}{r!s!t!(l-2r)!(m-2s)!(n-2t)!} \]

\[ \times \exp \left( -\epsilon \vec{k}^2 \right) \vec{k}_x^{l-2r} \vec{k}_y^{m-2s} \vec{k}_z^{n-2t} \]

(4.78)

The form of your solution may differ. Ensure that, with proper algebraic manipulation, your form and that presented here are equivalent.

[end of solution]

We have succeeded in resolving the three integrals of Eq. 4.74 \((V_l^x, V_m^y, V_n^z)\) over spatial coordinates \(dx_P \, dy_P \, dz_P\), as seen by the first two lines of Eq. 4.78. We collect these sums of constants into their own terms \(C_{RI}\), to separate it from the last line of Eq. 4.78, which is an integral we must still resolve.

Exercise.

If

\[ C_{RI} = C_{RI_x} C_{RI_y} C_{RI_z} \]

\[ = \begin{pmatrix} \pi \over \gamma \end{pmatrix}^{3/2} l!m!n!e^{(l+m+n) - \frac{(l+m+n)}{2}} \]

\[ \times \sum_{r=0}^{[l/2]} \sum_{s=0}^{[m/2]} \sum_{t=0}^{[n/2]} \frac{(-1)^{(r+s+t)}(2\sqrt{\epsilon})^{(l+m+n)-2(r+s+t)}}{r!s!t!(l-2r)!(m-2s)!(n-2t)!} \]

which are the first two lines of Eq. 4.78, use Eq. 4.78 to convince yourself that

\[ V_l^x \times V_m^y \times V_n^z = C_{RI} \times \exp \left( -\epsilon \vec{k}^2 \right) \vec{k}_x^{l-2r} \vec{k}_y^{m-2s} \vec{k}_z^{n-2t} \].

98
Solution.

\[ V^x_l \times V^y_m \times V^z_n = i^{(l+m+n)} \left( \frac{\pi}{\gamma} \right)^{3/2} l!m!n!e^{\left( \frac{im+in}{2} \right)} \]

\[ \times \sum_{r=0}^{[l/2]} \sum_{s=0}^{[m/2]} \sum_{t=0}^{[n/2]} \frac{(-1)^{r+s+t}(2\sqrt{\epsilon})^{(l+m+n)-2(r+s+t)}}{r!s!(l-2r)(m-2s)(n-2t)!} \]

\[ \times \exp \left( -\epsilon \frac{k^2}{k^2} \right) k^{2r} y^{2s} z^{2t} \]

\[ = C_{RI} \times \exp \left( -\epsilon \frac{k^2}{k^2} \right) k^{2r} y^{2s} z^{2t} \]

[ end of solution ]

By inserting Eq. 4.80 (i.e., \( V^x_l \times V^y_m \times V^z_n \)) into Eq. 4.74, we have

\[ V_{RI} = \frac{1}{2\pi^2} \cdot C_{RI} \int \frac{1}{k^2} \exp(ik \times PC) \left[ \exp \left( -\epsilon \frac{k^2}{k^2} \right) k^{2r} y^{2s} z^{2t} \right] \, dk \]

\[ = \frac{C_{RI}}{2\pi^2} \cdot V'_{RI}, \tag{4.81} \]

where we have given the name \( V'_{RI} \) to the integral over the transformation variable \( k \) still to be resolved.

In \( V'_{RI} \) of Eq. 4.81, we must address the problematic \( k^{-2} \) term, which when expanded is

\[ \frac{1}{k^2} = \frac{1}{k^2_x + k^2_y + k^2_z}. \tag{4.82} \]

To do so, we will take advantage of yet another identity, though in doing so must we must once again introduce another integral.

**Essential Equation**

\[ \exp \left( -\epsilon \frac{k^2}{k^2} \right) = 2\epsilon \frac{k^2}{u^3} \int_0^1 \frac{du}{u} \exp \left( -\epsilon \frac{k^2}{u^2} \right) \tag{4.83} \]

We refrained from decomposing \( \exp \left( -\epsilon \frac{k^2}{k^2} \right) \) in earlier steps in order to use this identity.
Exercise.

Substitute this identity (Eq. 4.83) into $V_{RI}'$ (Eq. 4.81), and express the result in terms of four integrals: three integrals dedicated to each of the $x$, $y$, and $z$ components of $\vec{k}$ (for all $\vec{k}$ terms, inside and outside of exponential functions), and one integral to the variable $u$. Then collect exponential functions with respect to each coordinate.

Solution.

$$V_{RI}' = \int \frac{1}{\vec{k}^2} \exp\left(i\vec{k} \times \vec{P}\vec{C}\right) \left[\exp\left(-\epsilon\vec{k}^2\right) \vec{k}^{l-2r} \vec{k}^m_{\vec{k}^2} \vec{k}^n_{\vec{k}^2} \right] \vec{k} \, d\vec{k}$$

$$= \int \frac{1}{\vec{k}^2} \exp\left(i\vec{k} \times \vec{P}\vec{C}\right) \left\{2\epsilon\vec{k}^2 \int_0^1 \frac{du}{u^3} \exp\left(-\epsilon\vec{k}^2 u^2\right) \vec{k}^{l-2r} \vec{k}^m_{\vec{k}^2} \vec{k}^n_{\vec{k}^2} \right\} \vec{k} \, d\vec{k} \quad (4.84)$$

The $\vec{k}^2$ term in the denominator is cancelled by the $\vec{k}^2$ in the identity. In turn, decomposing all $\vec{k}$ terms into components, and then collecting terms by these components, gives

$$V_{RI}' = 2\epsilon \int_0^1 \frac{du}{u^3} \times \left(\int_{-\infty}^{\infty} k_{lx}^{l-2r} \exp\left(-\epsilon u^2 \vec{k}_{lx}^2 + i\vec{P}\vec{C}_{lx}\vec{k}_{lx}\right) \, d\vec{k}_{lx}\right)$$

$$\times \left(\int_{-\infty}^{\infty} k_{ly}^m \exp\left(-\epsilon u^2 \vec{k}_{ly}^2 + i\vec{P}\vec{C}_{ly}\vec{k}_{ly}\right) \, d\vec{k}_{ly}\right)$$

$$\times \left(\int_{-\infty}^{\infty} k_{lz}^{n-2t} \exp\left(-\epsilon u^2 \vec{k}_{lz}^2 + i\vec{P}\vec{C}_{lz}\vec{k}_{lz}\right) \, d\vec{k}_{lz}\right) \quad (4.85)$$

Let us define $I_{k_{lx}}$ as one of the three integrals over $d\vec{k}$ in Eq. 4.85, such that

$$I_{k_{lx}} = \int_{-\infty}^{\infty} k_{lx}^{l-2r} \exp\left(-\epsilon u^2 \vec{k}_{lx}^2 + i\vec{P}\vec{C}_{lx}\vec{k}_{lx}\right) \, d\vec{k}_{lx} \quad (4.86)$$

and

$$V_{RI}' = 2\epsilon \int_0^1 \frac{du}{u^3} \times I_{k_{lx}} I_{k_{ly}} I_{k_{lz}} \quad (4.88)$$
You should be able to see that $I_{k_{i}}$ is of the same standard form that was used to evaluate the original “spatial” Cartesian factors (e.g., the integral over $dx \rho$ for $V_{j}^{z}$).
Exercise.

Show that, by applying Eq. 4.76 to $I_k$ as expressed in Eq. 4.85, the solution to $I_k$ is

$$I_k = i^{l-2r} \left( \frac{\pi}{\epsilon} \right)^{1/2} \left( \frac{1}{2} \right)^{l-2r} \left( \frac{1}{\sqrt{\epsilon}} \right)^{l-2r} \sum_{i=0}^{[(l-2r)/2]} \frac{e^{[2(i+1)-l]/2} \cdot P \bar{C}_x^{l-2r-2i} \cdot \frac{(-1)^i(l-2r)\epsilon}{i!(l-2r-2i)!}}{l^{2i} \cdot \epsilon \cdot \epsilon^{-1/2} \cdot \frac{1}{2\sqrt{\epsilon}} \cdot \frac{1}{\sqrt{\epsilon}}} \cdot (l-2r)! \cdot (l-2r-2i)! \cdot \exp(-\gamma P \bar{C}_x^2 \cdot u^2)$$

(4.89)

Note the distinction between the imaginary number $i$ and the index $i$.

Solution.

$$I_k = \int_{-\infty}^{\infty} k_x^{l-2r} \exp(-\epsilon u^{-2} k_x^2 + i P \bar{C}_x k_x) \, dk_x$$

$$= i^{l-2r} (l-2r)! \left( \frac{\pi}{\epsilon u^2} \right)^{1/2} \left( \frac{1}{2\sqrt{\epsilon u^2}} \right)^{l-2r} \exp \left[ -\left( \frac{P \bar{C}_x}{2\sqrt{\epsilon u^2}} \right)^2 \right] \sum_{i=0}^{[(l-2r)/2]} \frac{(-1)^i P \bar{C}_x \cdot \epsilon^{-1/2} \cdot \frac{1}{\sqrt{\epsilon}}}{i!} \cdot \frac{1}{(l-2r)/2} \cdot \frac{(l-2r)!}{(l-2r-2i)!} \cdot \frac{1}{(l-2r-2i)!}$$

$$= i^{l-2r} \left( \frac{\pi}{\epsilon} \right)^{1/2} \left( \frac{1}{2} \right)^{l-2r} \left( \frac{1}{\sqrt{\epsilon}} \right)^{l-2r} \sum_{i=0}^{[(l-2r)/2]} \frac{e^{[2(i+1)-l]/2} \cdot P \bar{C}_x^{l-2r-2i} \cdot \frac{(-1)^i(l-2r)\epsilon}{i!(l-2r-2i)!}}{l^{2i} \cdot \epsilon \cdot \epsilon^{-1/2} \cdot \frac{1}{2\sqrt{\epsilon}} \cdot \frac{1}{\sqrt{\epsilon}}} \cdot (l-2r)! \cdot (l-2r-2i)! \cdot \exp(-\gamma P \bar{C}_x^2 \cdot u^2)$$

[ end of solution ]
The solutions to $I_{k_y}$ and $I_{k_z}$ hold by analogy to Eq. 4.89, with summation indices of $j$ and $k$, respectively. It is noteworthy that in each $I_k$ expression, there are a distinct set of terms that include $u$. 
Exercise.

Extract all factors that include $u$ from $I_{k_x}$, $I_{k_y}$, and $I_{k_z}$, and let $I'_{k_x}$, $I'_{k_y}$, and $I'_{k_z}$ be the equations for $I_{k_x}$, $I_{k_y}$, and $I_{k_z}$ without the $u$ terms. For example,

$$I'_{k_x} = i^{l-2r} \left( \frac{\pi}{\epsilon} \right)^{1/2} \left( \frac{1}{2} \right)^{l-2r} \left( \frac{1}{\sqrt{\epsilon}} \right)^{l-2r}$$

$$\times \sum_{i=0}^{[(l-2r)/2]} \epsilon^{(2r+2i-l)/2} \cdot P\tilde{C}_x^{l-2r-2i} \cdot (-1)^i (l-2r)! \cdot \frac{i!(l-2r-2i)!}{i!}.$$  

Include the $u$ terms removed from $I_k$ as further integrand to the integral

$$\int_0^1 \frac{du}{u^3},$$

which appears in $V_{RI}'$ (Eq. 4.85). That is, fill in the bracketed part of

$$\int_0^1 u^{-3} \cdot \text{[additional } u \text{ terms]} \cdot du,$$

with the $u$ terms that are removed from $I_k$. Simplify the resulting integral, but don’t attempt to solve. Let $\nu$ equal the collection of powers outside of the exponential factors.

Solution.

$$\int_0^1 \frac{du}{u^3} \left[ u \cdot u^{(l-2r)} \cdot u^{(l-2r-2i)} \exp \left( -\gamma P\tilde{C}_x^2 \cdot u^2 \right) \right.$$

$$\times u \cdot u^{(m-2s)} \cdot u^{(m-2s-2j)} \exp \left( -\gamma P\tilde{C}_y^2 \cdot u^2 \right)$$

$$\times u \cdot u^{(n-2t)} \cdot u^{(n-2t-2k)} \exp \left( -\gamma P\tilde{C}_z^2 \cdot u^2 \right) \bigg] =$$

$$\int_0^1 \frac{du}{u^3} \cdot u^{2\nu} \cdot \exp \left[ -\gamma (P\tilde{C}_x^2 + P\tilde{C}_y^2 + P\tilde{C}_z^2) u^2 \right]$$
Simplifying, this is
\[
\int_0^1 u^{2\nu} \exp\left(-\gamma |\vec{PC}|^2 \cdot u^2\right) du,
\] (4.91)
where
\[
\nu = l + m + n - 2(r + s + t) - (i + j + k).
\]

At this point it is worth taking a step back and seeing where our considerable amount of work has taken us. Let us update our expression for \( V_{RI} \), which you should remind yourself makes up part of \( V \) (Eq. 4.70). \( V_{RI} \) was last expressed in full in Eq. 4.81.

\[
V_{RI} = \frac{C_{RI}}{2\pi^2} \cdot V'_{RI}
\]
\[
= \frac{C_{RI}}{2\pi^2} \cdot 2\epsilon \cdot I'_{k_x} I'_{k_y} I'_{k_z} \int_0^1 u^{2\nu} \exp\left(-\gamma |\vec{PC}|^2 \cdot u^2\right) du
\]
\[
= \frac{\epsilon}{\pi^2} \cdot C_{RI_x} C_{RI_y} C_{RI_z} \cdot I'_{k_x} I'_{k_y} I'_{k_z} \int_0^1 u^{2\nu} \exp\left(-\gamma |\vec{PC}|^2 \cdot u^2\right) du
\] (4.92)

We will address the integral over \( du \) momentarily. First, recall that \( C_{RI} \) and \( I'_k \) are simply collections of constants, without any variables, so it would be useful to simplify these constants as much as possible by combining them.

**Exercise.**

It can be seen from Eq. 4.79 that

\[
C_{RI} = C_{RI_x} C_{RI_y} C_{RI_z}
\] (4.93)

such that

\[
C_{RI_x} = i^l \left( \frac{\pi}{\gamma} \right)^{1/2} \epsilon^{l/2} \sum_{r=0}^{l/2} \frac{(-1)^r (2\sqrt{\epsilon})^{l-2r}}{r!(l-2r)!}.
\] (4.94)
Along with Eq. 4.90, show that

\[
C_{RI_x} \times I'_{k_x} = \left( \frac{\pi}{\gamma} \right)^{1/2} \left( \frac{\pi}{\epsilon} \right)^{1/2} \sum_{r=0}^{l/2} \frac{\sum_{i=0}^{[(l-2r)/2]} i^2(l-r) \cdot (-1)^r \cdot i! \cdot P_{C_x}^{l-2r-2i}}{i!(l-2r-2i)!} .
\] (4.95)

**Solution.**

\[
C_{RI_x} \times I'_{k_x} = i^l \left( \frac{\pi}{\gamma} \right)^{1/2} \left( \frac{\pi}{\epsilon} \right)^{1/2} e^{l/2} \sum_{r=0}^{l/2} \frac{(-1)^r (2 \sqrt{\epsilon})^{l-2r}}{r!(l-2r)!}
\times \left( \frac{1}{\epsilon} \right)^{l-2r} \left( \frac{1}{\sqrt{\epsilon}} \right)^{l-2r}
\times \sum_{i=0}^{[(l-2r)/2]} e^{(2r+2i-l)/2} \cdot P_{C_x}^{l-2r-2i} \cdot (-1)^i (l-2r)! \cdot i!(l-2r-2i)!
\]

\[
= \left( \frac{\pi}{\gamma} \right)^{1/2} \left( \frac{\pi}{\epsilon} \right)^{1/2} \sum_{r=0}^{l/2} \frac{\sum_{i=0}^{[(l-2r)/2]} i^2(l-r) \cdot (-1)^r}{i!(l-2r-2i)!}
\times \sqrt{\epsilon}^{l-2r} \times \left( \frac{1}{\sqrt{\epsilon}} \right)^{l-2r} 2^{l-2r} \left( \frac{1}{\epsilon} \right)^{l-2r}
\times e^{l/2} \epsilon^{(2r+2i-l)/2} \cdot (-1)^i P_{C_x}^{l-2r-2i} (l-2r)! \cdot i!(l-2r-2i)!
\]

\[
= \left( \frac{\pi}{\gamma} \right)^{1/2} \left( \frac{\pi}{\epsilon} \right)^{1/2} \sum_{r=0}^{l/2} \sum_{i=0}^{[(l-2r)/2]} \frac{i^2(l-r) \cdot (-1)^r \cdot i! \cdot P_{C_x}^{l-2r-2i}}{i!(l-2r-2i)!}
\]

[ end of solution ]
Exercise.

Construct the analogous \( y \)- and \( z \)-component products of \( C_{RI} \) and \( I'_{k} \). Then, show that to resolve all but the integral over \( du \) in Eq. 4.92, we take the product \( (\epsilon/\pi^2) \times C_{RI} \times I'_{k} \), as follows:

\[
\frac{\epsilon}{\pi^2} \times C_{RI_x} C_{RI_y} C_{RI_z} \times I'_{k_x} I'_{k_y} I'_{k_z} = \frac{2\pi}{\gamma} \sum_{r=0}^{[l/2]} \sum_{i=0}^{[(l-2r)/2]} \frac{(-1)^l (-1)^i \bar{P}_C_{x}^{r-l-2r-2i}}{i!(l-2r-2i)!} \\
\times \sum_{s=0}^{[m/2]} \sum_{j=0}^{[(m-2s)/2]} \frac{(-1)^m (-1)^j \bar{P}_C_{y}^{r-m-2s-2j}}{j!(m-2s-2j)!} \\
\times \sum_{t=0}^{[n/2]} \sum_{k=0}^{[(n-2t)/2]} \frac{(-1)^n (-1)^k \bar{P}_C_{z}^{r-n-2t-2k}}{k!(n-2t-2k)!},
\]

(4.96)
Solution.

\[
\frac{\epsilon}{\pi^2} C_{RI_x} C_{RI_y} C_{RI_z} \times I_{k_x}^\prime I_{k_y}^\prime I_{k_z}^\prime = \frac{\epsilon}{\pi^2} \times \left[ \left( \frac{\pi}{\gamma} \right)^{1/2} \left( \frac{\pi}{\epsilon} \right)^{1/2} \sum_{r=0}^{[l/2]} \sum_{s=0}^{[m/2]} i^{2(l-r)} (-1)^r \frac{l! (-1)^j \rho C_{x}^{l-2r-2i}}{i!(l-2r-2i)!} \right] \\
\times \left[ \left( \frac{\pi}{\gamma} \right)^{1/2} \left( \frac{\pi}{\epsilon} \right)^{1/2} \sum_{j=0}^{[(m-2s)/2]} i^{2(m-s)} (-1)^s \frac{m! (-1)^j \rho C_{y}^{m-2s-2j}}{j!(m-2s-2j)!} \right] \\
\times \left[ \left( \frac{\pi}{\gamma} \right)^{1/2} \left( \frac{\pi}{\epsilon} \right)^{1/2} \sum_{k=0}^{[(n-2t)/2]} i^{2(n-t)} (-1)^t \frac{n! (-1)^k \rho C_{z}^{n-2t-2k}}{k!(n-2t-2k)!} \right]
\]

\[
= \frac{\epsilon}{\pi^2} \left( \frac{\pi}{\gamma} \right)^{3/2} \left( \frac{\pi}{\epsilon} \right)^{3/2} \\
\times \sum_{r=0}^{[l/2]} \sum_{s=0}^{[m/2]} \sum_{t=0}^{[n/2]} (-1)^{l+m+n-(r+s+t)} i^{2(l+m+n-r-s-t)} (-1)^{r+s+t} l! m! n!
\]

\[
\times \sum_{i=0}^{[(l-2r)/2]} \sum_{j=0}^{[(m-2s)/2]} \sum_{k=0}^{[(n-2t)/2]} (-1)^{i+j+k} \rho C_{x}^{i-2r-2i} \rho C_{y}^{m-2s-2j} \rho C_{z}^{n-2t-2k} \\
\frac{i! j! k! (l-2r-2i)! (m-2s-2j)! (n-2t-2k)!}
\]

\[
= \frac{2\pi}{\gamma} \sum_{r=0}^{[l/2]} \sum_{i=0}^{[(l-2r)/2]} \left( (-1)^i \frac{l! (-1)^i \rho C_{x}^{l-2r-2i}}{i!(l-2r-2i)!} \right) \\
\times \sum_{s=0}^{[m/2]} \sum_{j=0}^{[(m-2s)/2]} \left( (-1)^s \frac{m! (-1)^j \rho C_{y}^{m-2s-2j}}{j!(m-2s-2j)!} \right) \\
\times \sum_{k=0}^{[n/2]} \sum_{t=0}^{[(n-2t)/2]} \left( (-1)^t \frac{n! (-1)^k \rho C_{z}^{n-2t-2k}}{k!(n-2t-2k)!} \right).
\]

[end of solution]

Looking to our original expression for \( V \), Eq. 4.70, we can substitute in \( V_{RI} \) (Eq. 4.92) using
\[ V = - Z_C \cdot \vec{K}_P \]
\[
\times \sum_{l_A+l_B} \sum_{m_A+m_B} \sum_{n_A+n_B} c(l_A, l_B, P_A, P_B) c_m(m_A, m_B, P_A, P_B) c_n(n_A, n_B, P_A, P_B) \]
\[
\times \int_{V_{RI}} x^l y^m z^n \exp\left(-\gamma P^2\right) \left(\frac{1}{P_C}\right) dV \]
\[
= - Z_C \times \frac{2\pi}{\gamma} \times \vec{K}_P \]
\[
\times \sum_{l_A+l_B} \sum_{m_A+m_B} \sum_{n_A+n_B} c(l_A, l_B, P_A, P_B) c_m(m_A, m_B, P_A, P_B) c_n(n_A, n_B, P_A, P_B) \]
\[
\times \int_{0}^{1} u^{2\nu} \exp\left(-\gamma |P_C|^2 \cdot u^2\right) du \]
\[ (4.97) \]

And yet there is still an integral! It would seem we are spinning our wheels.

However, this last integral has added significance: It is the Boys function, \( F_\nu(t) \).

\[ F_\nu(t) = \int_{0}^{1} t^{2\nu} \exp\left(-xt^2\right) dt \]  
\[ (4.98) \]

In fact, the Boys function cannot be analytically solved. We can go no further, and numerical approximations of \( F_\nu \) are unavoidable in our solutions. For this reason, considerable research has gone into generating good numerical solutions to this essential function, and fortunately there are many. The one we will implement in Python code (Handout 3) depends on gamma functions.

The method that we will implement in Python depends on gamma functions.
Exercise.

The solution to the Boys function we will implement in Python depends on the incomplete gamma function $\gamma(\nu + \frac{1}{2}, x)$, which is not to be confused with our variable $\gamma = \alpha_a + \alpha_b$. Specifically,

$$F_\nu(t) = \int_0^1 t^{2\nu} \exp(-xt^2) dt = \frac{1}{2x^{(\nu+\frac{1}{2})}} \int_0^x y^{\nu-\frac{1}{2}} e^{-y} dy = \frac{1}{2x^{(\nu+\frac{1}{2})}} \times \gamma(\nu + \frac{1}{2}, x). \quad (4.99)$$

The final two equalities can be found in handbooks of mathematical functions, such as by Abramowitz and Stegun.

Use substitution to show how the Boys Function can be rewritten as the integral that is related to the incomplete gamma function. That is, show that

$$\int_1^0 t^{2\nu} \exp(-xt^2) dt = \frac{1}{2x^{(\nu+\frac{1}{2})}} \int_0^x y^{\nu-\frac{1}{2}} e^{-y} dy. \quad (4.100)$$

(Hint: Let $y = xt^2$.)

Solution.

To begin with,

$$y = xt^2 \implies t = \frac{\sqrt{y}}{\sqrt{x}} = y^{1/2} \cdot x^{-1/2}$$

Thus,

$$dy = 2xt \, dt = 2xy^{1/2}x^{-1/2} = 2x^{1/2}y^{1/2} \, dt.$$  

Rearranging,

$$dt = \frac{1}{2} \cdot x^{-1/2}y^{-1/2} \, dy. \quad (4.100)$$

Clearly, because $y = xt^2$, then for the exponential function

$$e^{-xt^2} = e^{-y}. \quad (4.101)$$
Next, based on our definition of $t$,

$$
t^{2\nu} = \left(\frac{y}{x}\right)^\nu = y^\nu x^{-\nu}. \quad (4.102)
$$

Changing the integration bounds from $t$ to $y$:

$$
\int_{t=\sqrt{y/x}=1}^{t=\sqrt{y/x}=0} \Rightarrow y = x \quad \int_{0}^{x}.
$$

Piecing together each of these substitutions (Eqs. 4.100 – 4.103) into a complete integral gives

$$
\int_{0}^{1} t^{2\nu} e^{-xt^2} dt = \frac{1}{2x^{(\nu+\frac{1}{2})}} \int_{0}^{x} y^{\nu-1/2} e^{-y} dy
$$
as we set out to show.

[ end of solution ]

Now that we have a means of evaluating the Boys function, we have derived a completed solution to Eq. 4.70, and can couple Eq. 4.97 with 4.99. For tidiness of Eq. 4.97, we define

<table>
<thead>
<tr>
<th>Essential Equation</th>
</tr>
</thead>
<tbody>
<tr>
<td>$v_{l,r,i}(l_A, l_B, \vec{A}_x, \vec{B}_x, \vec{C}<em>x, \gamma) = (-1)^i c</em>{l}(l_A, l_B, \vec{PA}_x \vec{PB}_x) (-1)^i l! l! P C_x^{l-2r-2i} e^{r+i} (l-2r-2i)!$</td>
</tr>
</tbody>
</table>

with analogous expressions for $v_{m,s,j}$ and $v_{n,t,k}$, such that
and the complete matrix element, first shown in Eq. 4.70, can be evaluated as

\begin{align}
(A | -Z_C \vec{r}_{iC} | B) &= \sum_{a=1}^{K_a} \sum_{b=1}^{K_b} d_a d_b N_a N_b \times V \\
&= \sum_{l=0}^{l_A+l_B} \sum_{r=0}^{l/2} \sum_{i=0}^{l-2r/2} v_{l,r,i}(l_A, l_B, \vec{A}_x, \vec{B}_x, \vec{C}_x, \gamma) \\
&\times \sum_{m=0}^{m_A+m_B} \sum_{s=0}^{m/2} \sum_{j=0}^{m-2s/2} v_{m,s,j}(m_A, m_B, \vec{A}_y, \vec{B}_y, \vec{C}_y, \gamma) \\
&\times \sum_{n=0}^{n_A+n_B} \sum_{t=0}^{n/2} \sum_{k=0}^{n-2t/2} v_{n,t,k}(n_A, n_B, \vec{A}_z, \vec{B}_z, \vec{C}_z, \gamma) \\
&\times F_{l+m+n-2(r+s+t)-(i+j+k)}(\gamma|\vec{PC}|^2) \\
\end{align}

(4.106)
Electron-electron repulsion integrals

The final operator in the Hamiltonian is, like that of electron-nuclear attraction, corresponds to potential energy. Rather than an operator \( \langle \vec{r}_{iC}^{-1} \rangle \) describing the electrostatic attraction (potential energy) between one electron \( i \) and the nucleus of an atom \( C \), this final operator \( \langle \vec{r}_{ij}^{-1} \rangle \) describes the electrostatic repulsion (potential energy) between two electrons, \( i \) and \( j \). In the electron-nuclear attraction integrals, we had three spatial coordinates to integrate over: \( x, y, \) and \( z \). Electron repulsion integrals (ERI), so-called, necessitate six spatial coordinates to integrate over: the \( x, y, \) and \( z \) coordinates of electron \( i \), and the \( x, y, \) and \( z \) coordinates of electron \( j \).

Just as describing the overlap or kinetic energy of one electron was predicated on two orbitals (for orbitals are the “space” the electrons occupy), it follows that the description of two electrons depends on four orbitals: one electron \( i \) in orbitals \( \phi_A \) and \( \phi_B \), and one electron \( j \) in orbitals \( \phi_C \) and \( \phi_D \).

Because the electron-electron repulsion operator is a potential operator like the electron-nuclear attraction operator, deriving its analytical form is very similar. The techniques employed in the previous section to derive the analytical solution to the electron-nuclear attraction operator are the only ones necessary to derive the analytical solution to the electron-electron repulsion operator. As you might expect, it just happens to involve about twice as much manipulation.

The idea of a rank-4 tensor may initially seem a little abstract, but at least for our purposes it can simply be thought of as a generalization of a matrix. Where \( S, T, \) and \( V \) matrices are \( K \times K \) in dimension, electron-electron repulsion integral matrices \( G \) are \( K \times K \times K \times K \) in dimension. As a simple example, Fig. 4.14 shows the form of the \( 2 \times 2 \times 2 \times 2 \) tensor of electron-electron repulsion for the dihydrogen molecule, which has \( K = 2 \) atomic orbitals; evaluation of each integral shown on the left-hand side of the equivalence sign yields the Hartree energy values shown on the right-hand side.

A more complex example is the water molecule, which has \( K = 7 \) molecules in its basis; the general form of its electron repulsion tensor is shown in Fig. 4.15.

Now consider the general form of a tensor element \( (AB|CD) \), which we will refer to as a matrix
Figure 4.14: The $2 \times 2 \times 2 \times 2$ four-dimensional structure of the electron-electron repulsion tensor for the hydrogen molecule in the STO-3G basis, corresponding to the four indices of $(AB|CD)$. The results are based on a geometry where the interatomic distance is $1.6 \ a_0$; the input .xyz file is available in SI 5.

Figure 4.15: The $7 \times 7 \times 7 \times 7$ four-dimensional structure of the electron repulsion tensor for the water molecule in the minimal basis, corresponding to the four indices of $(AB|CD)$. As this image reveals, the presence of a greater number of orbitals leads rapidly to the growth and complexity of the tensor.

Element of $G$:

\[
(AB|CD) = \int \int \phi_A(\vec{r}_{A1}, \alpha_a, l_a, m_A, n_A)(\vec{r}_{B1}, \alpha_b, l_B, m_B, n_B) \phi_B(\vec{r}_{B1}, \alpha_b, l_B, m_B, n_B) \phi_C(\vec{r}_{C1}, \alpha_c, l_C, m_C, n_C) \phi_D(\vec{r}_{D1}, \alpha_d, l_D, m_D, n_D) \ dV_1 \ dV_2 \\
= \frac{1}{\hat{r}_{ij}} \frac{1}{\hat{r}_{ij}} \frac{1}{\hat{r}_{ij}} \frac{1}{\hat{r}_{ij}} \ \text{as before, we have identified the integral over individual primitives that needs to be analytically resolved, which we term } G. \]

Every “box” of this four-dimensional matrix must include the contraction coefficients and normalization constants for primitives within the four orbitals (or CGTF’s) $\phi_A$, $\phi_B$, $\phi_C$, and $\phi_D$ (three primitives per orbital, in the case of STO-3G). As before, we have identified the integral over individual primitives that needs to be analytically resolved, which we term $G$. By expanding the
expression for $G$ in terms of functional dependencies, we see that there is some new notation to consider.

$$G = \int \int \chi_a(\vec{A}, \alpha_a, l_A, m_a, n_A)(\vec{r}_{Ai})\chi_b(\vec{B}, \alpha_b, l_B, m_B, n_B)(\vec{r}_{Bi})$$

$$\times \left( \frac{1}{\vec{r}_{ij}} \right) \chi_c(\vec{C}, \alpha_c, l_C, m_C, n_C)(\vec{r}_{Cj})\chi_d(\vec{D}, \alpha_d, l_D, m_D, n_D)(\vec{r}_{Dj}) dV_i dV_j$$ \hspace{1cm} (4.108)

Thus, $\vec{r}_{Ai}$ is the distance between the coordinates of electron $i$ (denoted $\vec{r}_i$) and the coordinates $\vec{A}$ corresponding to the center of the atom-centered orbital $\phi_A$. A more explicit expression of what we need to solve is

$$G = \int_0^\infty \int_0^\infty x_A^I x_B^I y_A^I y_B^I z_A^I z_B^I e^{-\alpha_a r_A^2} e^{-\alpha_b r_B^2}$$

$$\times \left( \frac{1}{\vec{r}_{ij}} \right) l_C^I l_D^I c_C^I c_D^I e^{-\alpha_c r_C^2} e^{-\alpha_d r_D^2} dV_i dV_j$$ \hspace{1cm} (4.109)

$$dV_i = dx_i dy_i dz_i$$

$$dV_j = dx_j dy_j dz_j$$

where vector notation on $\vec{r}$ has been removed in the exponential function arguments for clarity. Note that $x_i$ is notation for the $x$-component of $\vec{r}_i$; and so on. As before, $r^2 = x^2 + y^2 + z^2$, and so you should be able to see how exponential functions can be separated based on their Cartesian coordinates.

Dropping the subscripts of the angular terms $x_A^I$, etc., let us rearrange Eq. 4.109 in terms of spatial coordinates, because this will form an important part of how we solve the problem.

$$G = \int_0^\infty x_A^I x_B^I e^{-(\alpha_a + \alpha_b)x_i^2} dx_i \int_0^\infty y_A^I y_B^I e^{-(\alpha_a + \alpha_b)y_i^2} dy_i \int_0^\infty z_A^I z_B^I e^{-(\alpha_a + \alpha_b)z_i^2} dz_i$$

$$dV_i \times \left( \frac{1}{\vec{r}_{ij}} \right) \int_0^\infty x_C^I x_D^I e^{-(\alpha_c + \alpha_d)x_j^2} dx_j \int_0^\infty y_C^I y_D^I e^{-(\alpha_c + \alpha_d)y_j^2} dy_j \int_0^\infty z_C^I z_D^I e^{-(\alpha_c + \alpha_d)z_j^2} dz_j$$ \hspace{1cm} (4.110)

The Gaussian product theorem is once again the gateway to simplification. We can reduce the four centers ($\vec{A}$, $\vec{B}$, $\vec{C}$, $\vec{D}$) to just two, by considering two products of orbitals, $\phi_A\phi_B$, and $\phi_C\phi_D$
– each of which provides the “space” for electrons $i$ and $j$, respectively, to occupy – to a linear combination of these centered on just two points, $\vec{P}$ and $\vec{Q}$. Information about electron $i$ will ultimately be described in terms of center $\vec{P}$, and information about electron $j$ will be described in terms of center $\vec{Q}$. (See Fig. 4.16.)

![Diagram of six centers](image)

Figure 4.16: The six centers involved in a two-electron integral. Centers $\vec{A}$ and $\vec{B}$ have the third center $\vec{P}$, which describes the space for the first electron, $i$; centers $\vec{C}$ and $\vec{D}$ have the third center $\vec{Q}$, which describes the space for the second electron, $j$.

**Exercise.**

Calculate the Gaussian product for the radial portion of primitives $\chi_a$ and $\chi_b$ (each contributing to orbitals $\phi_A$ and $\phi_B$), where the radial portion of $\chi_a$ is

$$\exp(-\alpha_a r_{Ai}^2),$$

and the radial portion of $\chi_b$ has the same form by analogy. Calculate the Gaussian product for the radial portions of primitives $\chi_c$ and $\chi_d$ also. Let $\gamma_P = \alpha_a + \alpha_b$ and $\gamma_Q = \alpha_c + \alpha_d$. 

116
Solution.

\[
\exp(-\alpha_a r^2_{A_i}) \exp(-\alpha_b r^2_{B_i}) = \exp\left(-\frac{\alpha_a \alpha_b |\vec{A}\vec{B}|^2}{\gamma_P}\right) \exp(-\gamma_P r^2_{P_i}) = \tilde{K}_P \exp(-\gamma_P r^2_{P_i})
\]

\[
\exp(-\alpha_c r^2_{C_j}) \exp(-\alpha_d r^2_{D_j}) = \exp\left(-\frac{\alpha_c \alpha_d |\vec{C}\vec{D}|^2}{\gamma_Q}\right) \exp(-\gamma_Q r^2_{Q_j}) = \tilde{K}_Q \exp(-\gamma_Q r^2_{Q_j})
\]

(4.111)

\[
\gamma_P = \alpha_a + \alpha_b, \quad \vec{P} = \frac{(\alpha_a \vec{A} + \alpha_b \vec{B})}{\gamma_P}
\]

\[
\gamma_Q = \alpha_c + \alpha_d, \quad \vec{Q} = \frac{(\alpha_c \vec{C} + \alpha_d \vec{D})}{\gamma_Q}
\]

\[
\vec{A}\vec{B} = \vec{A} - \vec{B}, \quad \vec{C}\vec{D} = \vec{C} - \vec{D}
\]

\[
\vec{r}_{P_i} = \vec{r}_i - \vec{P}, \quad \vec{r}_{Q_j} = \vec{r}_j - \vec{Q}
\]

\[
\vec{x}_{P_i} = x_i - \vec{P}_x, \quad \vec{x}_{Q_j} = x_j - \vec{Q}_x
\]

[ end of solution ]

What of the angular portion of these orbitals? We proceed as we have done before with all of the preceding integrals.
Exercise.

Calculate the $x$-component of the product of angular components of primitives $\chi_a$ and $\chi_b$,

$$x^{l_A}x^{l_B}.$$ 

Do the same for the $y$- and $z$-components.

Solution. Following the example of Eqs. 4.33, 4.35, and 4.40, the angular components over the product of primitives $\chi_a$ and $\chi_b$ are

$$x^{l_A}x^{l_B} = \sum_{l_P=0}^{l_A+l_B} c_{l_P}(l_A,l_B,\vec{P}A_x,\vec{P}B_x)x^{l_P},$$

$$y^{m_A}y^{m_B} = \sum_{m_P=0}^{m_A+m_B} c_{m_P}(m_A,m_B,\vec{P}A_y,\vec{P}B_y)y^{n_P},$$

$$z^{n_A}z^{n_B} = \sum_{n_P=0}^{n_A+n_B} c_{n_P}(n_A,n_B,\vec{P}A_z,\vec{P}B_z)z^{m_P}.$$ 

The new subscripts on the coefficients $c_k$ and coordinates $x^l, y^m, z^n, etc.$, will be necessary to distinguish between centers $\vec{P}$ and $\vec{Q}$.

[ end of solution ]

Exercise.

Repeat the previous exercise for the product of the angular components of the primitives corresponding to electron $j$. 

118
Solution.

The angular components over the product of primitives \( \chi_c \) and \( \chi_d \) are

\[
x^{l_C \neq l_D} = \sum_{l_Q = 0}^{l_C + l_D} c_{l_Q}(l_C, l_D, \vec{P}C_x, \vec{P}D_x)x^{l_Q},
\]

\[
y^{m_C \neq m_D} = \sum_{m_Q = 0}^{m_C + m_D} c_{m_Q}(m_C, m_D, \vec{P}C_y, \vec{P}D_y)y^{m_Q},
\]

\[
z^{n_C \neq n_D} = \sum_{n_Q = 0}^{n_C + n_D} c_{n_Q}(n_C, n_D, \vec{P}C_z, \vec{P}D_z)z^{n_Q}.
\]

[ end of solution ]

Let us consider the expression for \( G \) we’ve derived up to this point, in terms of the \( c_k \) factors we first saw in the electron-nuclear attraction integrals, as well as the formation of the new centers \( \vec{P} \) and \( \vec{Q} \) for the radial components. Because we are integrating over third center coordinates (\( \vec{P} \), \( \vec{Q} \)), we integrate over the range \((-\infty, \infty)\) instead of \([0, \infty)\).

\[
G = \tilde{K}_P \tilde{K}_Q
\]

\[
\times \sum_{l_p = 0}^{l_A + l_B} \sum_{m_p = 0}^{n_A + n_B} c_{l_p}(l_A, l_B, \vec{P}A_x, \vec{P}B_x)c_{m_p}(m_A, m_B, \vec{P}A_y, \vec{P}B_y)c_{n_p}(n_A, n_B, \vec{P}A_z, \vec{P}B_z)
\]

\[
\times \sum_{l_Q = 0}^{l_C + l_D} \sum_{m_Q = 0}^{n_C + n_D} c_{l_Q}(l_C, l_D, \vec{P}C_x, \vec{P}D_x)c_{m_Q}(m_C, m_D, \vec{P}C_y, \vec{P}D_y)c_{n_Q}(n_C, n_D, \vec{P}C_z, \vec{P}D_z)
\]

\[
\times \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} x^{l_p} y^{m_p} z^{n_p} \exp(-\gamma \sqrt{\gamma P_i^2 P_j}) \left( \frac{1}{\vec{r}_{ij}} \right) x^{l_Q} y^{m_Q} z^{n_Q} \exp(-\gamma \sqrt{\gamma Q_i^2 Q_j}) \, dV_i \, dV_j
\]

\[
G_{RI}(4.112)
\]

We’ve succeeded in deriving a general expression for the electron repulsion integral in terms of two centers instead of four. Because we know exactly the form of \( \tilde{K} \) and \( c_k \), the remaining work in resolving the analytical form of \( G \) will be in solving the last line of Eq. 4.112, which we call \( G_{RI} \) to imply the “remaining integral” of \( G \). There is much to be done to resolve \( G_{RI} \).

Perhaps most obviously, there is the matter of the inverse operator. Much as we did with the nuclear-electron attraction integral, we pull \( \vec{r}_{ij} \) from the denominator by subjecting it to a Fourier transform. Before we do, we must recast the denominator of our operator in terms of our two
Exercise.

Expand $\vec{r}_{ij}$ in terms of the two new centers $\vec{P}$ and $\vec{Q}$, analogous to what was done with recasting $\vec{r}_{iC}$ in terms of one new center, $\vec{P}$.

Solution.

$$\vec{r}_{ij} = \vec{r}_i - \vec{r}_j = (\vec{r}_i - \vec{P}) - (\vec{r}_j - \vec{Q}) = \vec{r}_{Pi} - \vec{r}_{Qj} + \vec{PQ} \quad (4.113)$$

$$\vec{PQ} = \vec{P} - \vec{Q}.$$  

[ end of solution ]

Exercise.

Insert the solution of Eq. 4.114 into $G_{RI}$, the last line of Eq. 4.112.
Solution.

\[ G_{RI} = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} x^l y^m z^n \exp\left(-\gamma_p r_i^2\right) \left(\frac{1}{r_{ij}}\right) x^l \sigma y^m z^n \exp\left(-\gamma_q r_j^2\right) \, dV_i \, dV_j \]

\[ = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} x^l y^m z^n \exp\left(-\gamma_p r_i^2\right) \times \left[ \frac{1}{2\pi^2} \int \frac{1}{k^2} \exp\left(ik \cdot r_i\right) \exp\left(ik \cdot r_j\right) \exp\left(ik \cdot P\bar{Q}\right) \, dk\right] x^l \sigma y^m z^n \exp\left(-\gamma_q r_j^2\right) \, dV_i \, dV_j \]

(4.115)

[ end of solution ]

Exercise.

With the exception of the terms \( \left(\frac{1}{k^2}\right) \) and \( \exp\left(ik \cdot P\bar{Q}\right) \), expand \( G_{RI} \) in terms of the \( x, y, \) and \( z \) components of \( \vec{r}_p, \vec{r}_q, \) and \( \vec{k} \). Note that the differential volume elements are

\[ dV_i = dx_P \, dy_P \, dz_P , \]
\[ dV_j = dx_Q \, dy_Q \, dz_Q , \]
\[ d\vec{k} = d\vec{k}_x \, d\vec{k}_y \, d\vec{k}_z . \]

Collect terms by these spatial components. You should end up with one integral that is

\[ \frac{1}{2\pi^2} \int \frac{1}{k^2} \exp\left(ik \cdot P\bar{Q}\right) \, dk , \]

multiplied by six integrals in the general form of

\[ \int_{-\infty}^{\infty} x^n \exp(-ax^2 + bx) \, dx . \]
Solution.

\[ G_{RI} = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} x^p y^m z^n \exp(-\gamma p r^2_{\mathbf{P}_i}) \]
\[ \times \left[ \frac{1}{2\pi^2} \int \frac{1}{k^2} \exp\left(\vec{k} \cdot \vec{r}_{\mathbf{P}_i}\right) \exp\left(\vec{k} \cdot \vec{r}_{Q_j}\right) \exp\left(\vec{k} \cdot \vec{P}_{\mathbf{Q}}\right) d\vec{k} \right] \]
\[ \times \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} x^p y^m z^n \exp(-\gamma p x^2_{\mathbf{P}_i}) \exp(-\gamma p y^2_{\mathbf{P}_i}) \exp(-\gamma p z^2_{\mathbf{P}_i}) \] d\mathbf{P} _i d\mathbf{Q} _j 
\[ = \frac{1}{2\pi^2} \int \frac{1}{k^2} \exp\left(\vec{k} \cdot \vec{r}_{\mathbf{P}_i}\right) \exp\left(\vec{k} \cdot \vec{r}_{Q_j}\right) \exp\left(\vec{k} \cdot \vec{P}_{\mathbf{Q}}\right) d\vec{k} \]
\[ \times \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} x^p y^m z^n \exp(-\gamma p x^2_{\mathbf{P}_i}) \exp(-\gamma p y^2_{\mathbf{P}_i}) \exp(-\gamma p z^2_{\mathbf{P}_i}) \] d\mathbf{P} _i d\mathbf{Q} _j 
\[ = \frac{1}{2\pi^2} \int \frac{1}{k^2} \exp\left(\vec{k} \cdot \vec{P}_{\mathbf{Q}}\right) d\vec{k} \]
\[ \times \int_{-\infty}^{\infty} x^p \exp(-\gamma p x^2_{\mathbf{P}_i}) \exp\left(\vec{k}_x \cdot \mathbf{x}\right) \] d\mathbf{P} _i \times \int_{-\infty}^{\infty} x^q \exp(-\gamma q x^2_{\mathbf{Q}_j}) \exp\left(\vec{k}_x \cdot \mathbf{x}\right) \] d\mathbf{Q} _j 
\[ = \frac{1}{2\pi^2} \int \frac{1}{k^2} \exp\left(\vec{k} \cdot \vec{P}_{\mathbf{Q}}\right) d\vec{k} \]
\[ \times \int_{-\infty}^{\infty} x^p \exp(-\gamma p x^2_{\mathbf{P}_i} + \vec{k}_x \cdot \mathbf{x}\right) \] d\mathbf{P} _i \times \int_{-\infty}^{\infty} x^q \exp(-\gamma q x^2_{\mathbf{Q}_j} - \vec{k}_x \cdot \mathbf{x}\right) \] d\mathbf{Q} _j 
\[ \times \int_{-\infty}^{\infty} y^m \exp(-\gamma p y^2_{\mathbf{P}_i} + \vec{k}_y \cdot \mathbf{y}\right) \] d\mathbf{P} _i \times \int_{-\infty}^{\infty} y^n \exp(-\gamma q y^2_{\mathbf{Q}_j} - \vec{k}_y \cdot \mathbf{y}\right) \] d\mathbf{Q} _j 
\[ \times \int_{-\infty}^{\infty} z^n \exp(-\gamma p z^2_{\mathbf{P}_i} + \vec{k}_z \cdot \mathbf{z}\right) \] d\mathbf{P} _i \times \int_{-\infty}^{\infty} z^n \exp(-\gamma q z^2_{\mathbf{Q}_j} - \vec{k}_z \cdot \mathbf{z}\right) \] d\mathbf{Q} _j 

(4.116)

It will prove useful to use some shorthand notation:

\[ G_{IP}^p = \int_{-\infty}^{\infty} x^p \exp\left(-\gamma p x^2_{\mathbf{P}_i} + \vec{k}_x \cdot \mathbf{x}\right) \] d\mathbf{P} _i 

(4.117)
so that the last line of Eq. 4.116 can be expressed as

\[ G_{RI} = \frac{1}{2\pi^2} \int \frac{1}{\vec{k}^2} \exp\left( i\vec{k} \cdot \vec{P} \right) d\vec{k} \times G^{xP}_{l_P} \times G^{yQ}_{l_Q} \times G^{zP}_{m_P} \times G^{zQ}_{n_P} \times G^{yQ}_{m_Q} \times G^{zQ}_{n_Q} \]  

(4.118)

The form of \( G^{xP}_{l_P} \) as it appears in Eq. 4.117 should be familiar, and we know how to solve it.

**Exercise.**

Let \( \epsilon_P = (4\gamma_P)^{-1} \), and provide the solution to Eq. 4.117, for \( G^{xP}_{l_P} \), using Eq. 4.76. Repeat the exercise for \( G^{xQ}_{l_Q} \).

**Solution.**

\[
G^{xP}_{l_P} = \int_{-\infty}^{\infty} x^{l_P} \exp\left( -\gamma_P x^2_P + i\vec{k} \cdot x_P \right) dx_P \\
= i^l \left( \frac{\pi}{\gamma_P} \right)^{1/2} l_P! \epsilon_P^{l_P/2} \exp\left( -\epsilon_P k_x^2 \right) \times \sum_{r_P=0}^{[l_P/2]} \frac{(-1)^r_P (2\sqrt{\epsilon_P k_x})^{l_P - 2r_P}}{r_P!(l_P - 2r_P)!} 
\]  

(4.119)

\[
G^{xQ}_{l_Q} = \int_{-\infty}^{\infty} x^{l_Q} \exp\left( -\gamma_Q x^2_Q - i\vec{k} \cdot x_Q \right) dx_Q \\
= i^l \left( \frac{\pi}{\gamma_Q} \right)^{1/2} l_Q! \epsilon_Q^{l_Q/2} \exp\left( -\epsilon_Q k_x^2 \right) \times \sum_{r_Q=0}^{[l_Q/2]} \frac{(-1)^r_Q (-1 \cdot 2\sqrt{\epsilon_Q k_x})^{l_Q - 2r_Q}}{r_Q!(l_Q - 2r_Q)!} 
\]  

(4.120)

Take notice that, in addition to different indices, Eq. 4.120 is distinguished from Eq. 4.119 by the presence of a \((-1)^{l_Q - 2r_Q}\) term. Additionally, in the square root fraction, we reverted \( \epsilon / 4 \) to \( \gamma \), which will be useful in the final steps of the derivation of \( G \). [ end of solution ]
Exercise.

Using the form derived in Eqs. 4.119 and 4.120, calculate the product $G_{iP}^{xP}G_{mP}^{yP}G_{nQ}^{zP}G_{lQ}^{wQ}G_{mQ}^{yQ}G_{nQ}^{zQ}$, collecting terms like so:

$$(G_{iP}^{xP}G_{lQ}^{zQ}) \times (G_{mP}^{yP}G_{mQ}^{yQ}) \times (G_{nP}^{zP}G_{nQ}^{zQ})$$

Finally, collect an exp$[-(\epsilon_P + \epsilon_Q)\vec{k}^2]$ term as a multiplicative factor of the remaining, decomposed terms, without decomposing $\vec{k}^2$ into its spatial coordinates.

Solution.

$$
\begin{align*}
\chi^{lP+lQ} & \left( \frac{\pi}{\gamma_P} \right)^{1/2} \left( \frac{\pi}{\gamma_Q} \right)^{1/2} I_{\gamma_P}^{1/2}I_{\gamma_Q}^{1/2} \epsilon_{lP}^{1/2} \epsilon_{lQ}^{1/2} \sum_{t_1 = 0}^{[lP/2]} \sum_{t_2 = 0}^{[lQ/2]} \frac{(-1)^{t_1}I_{\gamma_P}^{2\sqrt{\epsilon_{lP}}}I_{\gamma_Q}^{2\sqrt{\epsilon_{lQ}}} - 2r_P (-1)^{t_2}I_{\gamma_Q}^{2\sqrt{\epsilon_{lQ}}}}{r_P!(l_P - 2r_P)!} \times \frac{(-1)^{t_2}I_{\gamma_Q}^{2\sqrt{\epsilon_{lQ}}}}{r_Q!(l_Q - 2r_Q)!} \sum_{s_1 = 0}^{[mP/2]} \sum_{s_2 = 0}^{[mQ/2]} \frac{(-1)^{s_1}I_{\gamma_Q}^{2\sqrt{\epsilon_{lQ}}}}{s_P!(m_P - 2s_P)!} \sum_{t_1 = 0}^{[nP/2]} \sum_{t_2 = 0}^{[nQ/2]} \frac{(-1)^{t_1}I_{\gamma_P}^{2\sqrt{\epsilon_{lP}}}}{t_P!(n_P - 2t_P)!} \times \frac{(-1)^{t_2}I_{\gamma_Q}^{2\sqrt{\epsilon_{lQ}}}}{t_Q!(n_Q - 2t_Q)!} \chi^{mP+mQ} \left( \frac{\pi}{\gamma_P} \right)^{1/2} \left( \frac{\pi}{\gamma_Q} \right)^{1/2} m_P!m_Q! \epsilon_{lP}^{1/2} \epsilon_{lQ}^{1/2} \times \sum_{t_1 = 0}^{[nP/2]} \sum_{t_2 = 0}^{[nQ/2]} \frac{(-1)^{t_1}I_{\gamma_P}^{2\sqrt{\epsilon_{lP}}}}{t_P!(n_P - 2t_P)!} \frac{(-1)^{t_2}I_{\gamma_Q}^{2\sqrt{\epsilon_{lQ}}}}{t_Q!(n_Q - 2t_Q)!}.
\end{align*}
$$

You may have ordered some specific terms differently than that presented here in Eq. 4.121. If so, ensure that, with appropriate manipulation, the two forms are equivalent.

[ end of solution ]

By this point we have succeeded in integrating over d$V_i$ and d$V_j$, because we solved each $G_i^x$ term and took their product. We can expand out Eq. 4.118 in terms of these solutions, by simply inserting Eq. 4.121 into Eq. 4.118. Even doing so, it remains for us to integrate over d$\vec{k}$, which was introduced in the Fourier transform to draw $\vec{r}_{ij}$ out of the denominator. Since we are interested in those terms involving $\vec{k}$, the following equation features the result of substituting Eq. 4.121 into Eq. 4.118.

$$
G_{RI} = \frac{1}{2\pi^2} \int \frac{1}{k^2} \exp(i\vec{k} \cdot \vec{P}Q) \exp(-\delta \vec{k}^2) \times C_{RI} \times \frac{\Gamma_{lP}^{xP} \Gamma_{mP}^{yP} \Gamma_{nQ}^{zQ} \Gamma_{lQ}^{wQ} \Gamma_{mQ}^{yQ} \Gamma_{nQ}^{zQ}}{k_x^{\gamma_P} - 2r_P k_x^{\gamma_Q} - 2r_Q k_y^{\gamma_P} - 2r_P k_y^{\gamma_Q} - 2s_P k_x^{\gamma_Q} - 2s_Q k_x^{\gamma_Q} - 2t_P k_x^{\gamma_Q} - 2t_Q k_x^{\gamma_Q}} \, d\vec{k} = \frac{C_{RI}}{2\pi^2} \int \frac{1}{k^2} \exp(i\vec{k} \cdot \vec{P}Q) \exp(-\delta \vec{k}^2) \frac{k_x^{\gamma_P} - 2r_P + l_Q - 2r_Q k_y^{\gamma_Q} - 2s_P + m_Q - 2s_Q k_x^{\gamma_Q} - 2t_P + n_Q - 2t_Q}{k_x^{\gamma_Q} - 2t_Q} \, d\vec{k},
$$

(4.122)
where \( C_{RI} \) is a collection of all constant terms in \( G_{RI} \) that are not dependent on \( \vec{k} \),

\[
C_{RI} = i^{p+q} \left( \frac{\pi}{\gamma_p} \right)^{1/2} \left( \frac{\pi}{\gamma_Q} \right)^{1/2} \frac{1}{l_p!l_Q!} \frac{l_p^{l_p/2} l_Q^{l_Q/2}}{m_p!m_Q!} \times \sum_{r_p=0}^{[l_p/2]} \sum_{r_Q=0}^{[l_Q/2]} \frac{(-1)^{r_p} (2\sqrt{\epsilon_p})^{l_p-2r_p} (-1)^{r_Q} (1 - 2\sqrt{\epsilon_Q})^{l_Q-2r_Q}}{r_p!l_p! - 2r_p)!} \frac{1}{r_Q!(l_Q - 2r_Q)!} \\
\times i^{m_p+m_Q} \left( \frac{\pi}{\gamma_p} \right)^{1/2} \left( \frac{\pi}{\gamma_Q} \right)^{1/2} \frac{1}{m_p!m_Q!} \frac{e^{m_p/2} e^{m_Q/2}}{n_p!n_Q!} \times \sum_{s_p=0}^{[m_p/2]} \sum_{s_Q=0}^{[m_Q/2]} \frac{(-1)^{s_p} (2\sqrt{\epsilon_p})^{m_p-2s_p} (-1)^{s_Q} (1 - 2\sqrt{\epsilon_Q})^{m_Q-2s_Q}}{t_p!(m_p - 2s_p)!} \frac{1}{s_q!(m_Q - 2s_Q)!} \\
\times i^{n_p+n_Q} \left( \frac{\pi}{\gamma_p} \right)^{1/2} \left( \frac{\pi}{\gamma_Q} \right)^{1/2} \frac{1}{n_p!n_Q!} \frac{e^{n_p/2} e^{n_Q/2}}{t_p!(n_p - 2t_p)!} \times \sum_{t_p=0}^{[n_p/2]} \sum_{t_Q=0}^{[n_Q/2]} \frac{(-1)^{t_p} (2\sqrt{\epsilon_p})^{n_p-2t_p} (-1)^{t_Q} (1 - 2\sqrt{\epsilon_Q})^{n_Q-2t_Q}}{t_p!(n_p - 2t_p)!} \frac{1}{t_Q!(n_Q - 2t_Q)!} ,
\]

(4.123)

and which can be safely set aside for the moment. Doing so allows us to focus on the manipulations of \( \vec{k} \).

Eq. 4.122 suffers from the presence of \( \vec{k}^2 \) in a denominator. How do you think we might ameliorate this problem? If you have not already guessed as much, we will use the same technique as we did with electron-nuclear attraction integrals to redress this. As before, we do so at the price of a new integral.

---

Exercise.

Employ the identity of Eq. 4.83 in Eq. 4.122. At the end of the exercise, combine the two exponential functions into one.

**Solution.**

\[
G_{RI} = \frac{C_{RI}}{2\pi^2} \int \frac{1}{k^2} \exp \left( i \vec{k} \cdot \vec{P} \vec{Q} \right) \exp \left( -\delta \vec{k}^2 \right) k_x^{2p+2r_p+2-l_Q-2r_Q} k_y^{m_p-2s_p+m_Q-2s_Q} k_z^{m_p-2t_p+n_Q-2t_Q} \, dk \\
= \frac{C_{RI}}{2\pi^2} \int \frac{1}{k^2} \exp \left( i \vec{k} \cdot \vec{P} \vec{Q} \right) 2\delta \vec{k}^2 \int_0^1 \frac{du}{u^3} \exp \left( -\delta \vec{k}^2 u^{-2} \right) \\
\times k_x^{2p-2r_p+l_Q-2r_Q} k_y^{m_p-2s_p+m_Q-2s_Q} k_z^{m_p-2t_p+n_Q-2t_Q} \, dk \\
= \frac{1}{\pi^2} \int_0^1 \frac{du}{u^3} \int_{-\infty}^{\infty} \exp \left( -\delta \vec{k}^2 u^{-2} + i \vec{k} \cdot \vec{P} \vec{Q} \right) k_x^{2p-2r_p+l_Q-2r_Q} k_y^{m_p-2s_p+m_Q-2s_Q} k_z^{m_p-2t_p+n_Q-2t_Q} \, dk \\
\]

(4.124)

Now it should be clear why we had not decomposed \( \exp \left( -\delta \vec{k}^2 \right) \) into its components, so that we could make this identity substitution.

[ end of solution ]
If we consider the components of $\vec{k}$ ($\vec{k}_x$, $\vec{k}_y$, $\vec{k}_z$), then there are now four multiple integrals to be solved in $G_{RI}$.

$$G_{RI} \approx \int_0^1 \frac{du}{u^3} \int_{-\infty}^{\infty} \cdots d\vec{k}_x \int_{-\infty}^{\infty} \cdots d\vec{k}_y \int_{-\infty}^{\infty} \cdots d\vec{k}_z$$

Exercise.

Derive the solution to the integral over $d\vec{k}_x$.

$$\int_{-\infty}^{\infty} \vec{k}_x^{(l_p-2r_r)+(l_Q-2r_q)} \exp\left(-\delta \vec{k}_x^2 u^2 + i \vec{k}_x \vec{PQ}_x\right) d\vec{k}_x$$

(Hint: You’ve solved integrals with this general form before.)

Repeat the exercise for the integrals over $d\vec{k}_y$ and $d\vec{k}_z$.

Solution.

Proceed by using the standard integral of Eq. 4.76.

$$\int_{-\infty}^{\infty} \vec{k}_x^{(l_p-2r_r)+(l_Q-2r_q)} \exp\left(-\delta \vec{k}_x^2 u^2 + i \vec{k}_x \vec{PQ}_x\right) d\vec{k}_x =$$

$$\sum_{i=0}^{[(l_p-2r_r)+(l_Q-2r_q)]/2} \frac{u \cdot u^{(l_p-2r_r)+(l_Q-2r_q)} \cdot u^{(l_p-2r_r)+(l_Q-2r_q)-2i} \cdot \exp\left(-\vec{PQ}_x^2/(4\delta) u^2\right) \cdot i^{(l_p-2r_r)+(l_Q-2r_q)}}{i^{(l_p-2r_r)+(l_Q-2r_q)-2i} \cdot \left(\frac{\pi}{\delta}\right)^{1/2} \left(\frac{1}{2}\right)^{l_p-2r_r} (l_p-2r_r)! \left(\frac{1}{\sqrt{\delta}}\right)^{l_p-2r_r} (l_p-2r_r)!}$$

We have separated out some terms here for future utility. Ensure that, with proper manipulation, your solution agrees with Eq. 4.125.

The solutions over $d\vec{k}_y$ and $d\vec{k}_z$ are entirely analogous, with summation indices $j$ and $k$ in lieu of $i$.

[ end of solution ]
Exercise.

Extract all terms that include contain $u$ from the previous exercise (Eq. 4.125) and introduce them as arguments of the integral over $du$ that appears in Eq. 4.124.

\[ \int_{0}^{1} \frac{du}{u^3} \]

Introduce also the $u$ terms that appear in the analogous solutions to integration over $d\vec{k}_y$ and $d\vec{k}_z$. In other words, fill in the bracketed part of

\[ \int_{0}^{1} u^{-3} \cdot [\text{additional } u \text{ terms}] \cdot du , \]

Simplify the resulting integral, letting $\nu$ equal the collection of powers outside of the exponential factors. Do not attempt to solve the integral.

Solution.

\[ \int_{0}^{1} \frac{du}{u^3} [u \cdot u^{(l_p-2r_p)+(l_Q-2r_Q)} \cdot u^{(l_p-2r_p)+(l_Q-2r_Q)}] \cdot \exp \left[-P\bar{Q}^2/\left(4\delta\right)u^2\right] \]

\[ \times u \cdot u^{(m_p-2s_p)+(m_Q-2s_Q)} \cdot u^{(m_p-2s_p)+(m_Q-2s_Q)} \cdot \exp \left[-P\bar{Q}^2/\left(4\delta\right)u^2\right] \]

\[ \times u \cdot u^{(n_p-2t_p)+(n_Q-2t_Q)} \cdot u^{(n_p-2t_p)+(n_Q-2t_Q)} \cdot \exp \left[-P\bar{Q}^2/\left(4\delta\right)u^2\right] \]

\[ = \int_{0}^{1} \frac{du}{u^3} \cdot u^{2\nu} \cdot \exp \left[-(P\bar{C}^2_x + P\bar{C}^2_y + P\bar{C}^2_z)/(4\delta)u^2\right] \]

where

\[ \nu = l_p + l_Q + m_p + m_Q + n_p + n_Q - 2(r_p + r_Q + s_p + s_Q + t_p + t_Q) - (i + j + k) \]

which simplifies to the Boys Function,

\[ \int_{0}^{1} u^{2\nu} \exp \left[-P\bar{C}^2/\left(4\delta\right)u^2\right] du = F_\nu[P\bar{Q}^2/(4\delta)] \]

[ end of solution ]

Now all terms involving $u$, which was introduced to remove $\vec{k}^2$ from the denominator, are
resolved. Additionally, all terms involving \( \vec{k} \) have been integrated out. All integrals over \( dV_i \) and \( dV_j \) were also solved several exercises before. Thus, all integrals have been solved. What we are left with is the complex assemblage of their solutions, which we will now simplify as fully as possible.

Look again at Eq. 4.125. Since we removed the \( u \) terms to collect the Boys Function (Eq. 4.127), then Eq. 4.125 takes on the following form, which we will term \( I_{k_x} \).

\[
I_{k_x} = \sum_{i=0}^{[(l_P-2r_P+l_Q-2r_Q)/2]} i^{(l_P-2r_P)+(l_Q-2r_Q)} (\frac{\pi}{\delta})^{1/2} (\frac{1}{2})^{(l_P-2r_P)+(l_Q-2r_Q)} \times \left( \frac{1}{\sqrt{\delta}} \right)^{(l_P-2r_P)+(l_Q-2r_Q)-2i} \times \vec{P}^2 \frac{Q_x}{\delta}^{(l_P-2r_P)+(l_Q-2r_Q)-2i} (-1)^i (l_P - 2r_P + l_Q - 2r_Q)! \frac{1}{i!}(l_P - 2r_P + l_Q - 2r_Q - 2i)! \right)
\]

(4.128)

We know there are analogous expressions for the \( y \) and \( z \) components, which we will term \( I_{k_y} \) and \( I_{k_z} \).

Consider \( I_{k_x} \) (Eq. 4.128) in the context of \( G_{RI} \), which we last saw expressed in Eq. 4.124.

\[
G_{RI} = \frac{\delta \cdot C_{RI}}{\pi^2} \times I_{k_x} I_{k_y} I_{k_z} \times F_{\nu} \frac{\vec{P}^2 Q}{(4\delta)} \]

(4.129)

where \( C_{RI} \) is expressed in Eq. 4.123, and \( C_{RI} = C_{RI_x} C_{RI_y} C_{RI_z} \) is the product of the spatial decomposition of \( C_{RI} \), just as each line of Eq. 4.123 clearly shows the delineated \( x \), \( y \), and \( z \) contributions. (Take a look again at Eqs. 4.112 and 4.108 to remind yourself of \( G_{RI} \) in the context of \( G \) and \( (AB|CD) \), respectively.)

We proceed with the simplification of \( G_{RI} \) by considering the \( x \) component product of \( C_{RI} \) and \( I_k \).

---

**Exercise.**

Using the \( x \)-component of \( C_{RI} \) shown in Eq. 4.123, and \( I_{k_x} \) as expressed in Eq. 4.128, show that
the product of $C_{RI_x}$ and $I_{k_z}$ is

$$C_{RI_x} \times I_{k_z} = \left( \frac{\pi}{\gamma_P} \right)^{1/2} \left( \frac{\pi}{\gamma_Q} \right)^{1/2} \left( \frac{\pi}{\delta} \right)^{1/2} \times \sum_{r_P=0}^{[l_P/2]} \sum_{r_Q=0}^{[l_Q/2]} \sum_{i=0}^{[(l_P-2r_P+l_Q-2r_Q)/2]} \frac{l_P! \gamma_{l_P-r_P}!}{r_P!(l_P-2r_P)!} \times \frac{l_Q! \gamma_{l_Q-r_Q}!}{r_Q!(l_Q-2r_Q)!} \times \frac{(-1)^r (2\delta)^{2(r_P+r_Q)} (l_P+l_Q-2r_P-r_Q)! \delta^i P_{P_Q}^{l_P+l_Q-2(r_P+r_Q+i)}}{(4\delta)^{l_P+l_Q+i}! (l_P+l_Q-2r_P-2r_Q-2i)!}$$
Solution.

\[
C_{\text{RP}} \times I_{\text{RP}} = \frac{\pi}{\gamma_p P} \left( \frac{\pi}{\gamma_q Q} \right)^{1/2} \left( \frac{\pi}{\gamma_q Q} \right)^{1/2} \left( \frac{\pi}{\gamma_q Q} \right)^{1/2} \left( \frac{\pi}{\gamma_q Q} \right)^{1/2} \times \sum_{r_p=0}^{(p/2)} \sum_{r_q=0}^{(q/2)} \frac{(-1)^{r_p} \left(2\sqrt{r_q}P\right)^{r_p-2r_q} \left(-1\right)^{r_q} \left(1 - 2\sqrt{r_q}Q\right)^{r_q-(q-2r_q)}}{r_p! (l_p - 2r_p)!} \frac{\left(1 - \sqrt{r_q}Q\right)^{l_q-(q-2r_q)}}{r_q!} \\
\times \left( \frac{\pi}{\gamma_p} \right)^{1/2} \left( \frac{\pi}{\gamma_q} \right)^{1/2} \left( \frac{\pi}{\gamma_q} \right)^{1/2} \left( \frac{\pi}{\gamma_q} \right)^{1/2} \left( \frac{\pi}{\gamma_q} \right)^{1/2} \sum_{i=0}^{(l_p-2r_p)+(l_q-(q-2r_q)/2)} \frac{\left(1 \right)^{i}}{i!} \left(l_p - 2r_p + l_q - 2r_q - 2i\right)! \\
\times \frac{(1/2)^{l_p+q-(2r_p+2r_q)}}{\sqrt{Q}^{l_p-2r_p} \sqrt{Q}^{Q-2r_q}} \delta^i \delta^2 \frac{l_p! \left(l_p - 2r_p + l_q - 2r_q\right)\left(-1\right)^{r_p} \left(-1\right)^{r_q} \left(-1\right)^{l_p+q-(2r_p+2r_q)}}{r_p! (l_p - 2r_p)! r_q! (l_q - 2r_q)!} \\
= \left( \frac{\pi}{\gamma_p} \right)^{1/2} \left( \frac{\pi}{\gamma_q} \right)^{1/2} \left( \frac{\pi}{\gamma_q} \right)^{1/2} \left( \frac{\pi}{\gamma_q} \right)^{1/2} \sum_{r_p=0}^{(p/2)} \sum_{r_q=0}^{(q/2)} \frac{(-1)^{r_p+q-r_q}}{r_p! (l_p - 2r_p)! r_q! (l_q - 2r_q)!} \\
\times \frac{\left(1/2\right)^{l_p+q-(2r_p+2r_q)}}{\sqrt{Q}^{l_p-2r_p} \sqrt{Q}^{Q-2r_q}} \delta^i \delta^2 \frac{l_p! \left(l_p - 2r_p + l_q - 2r_q\right)\left(-1\right)^{r_p} \left(-1\right)^{r_q} \left(-1\right)^{l_p+q-(2r_p+2r_q)}}{r_p! (l_p - 2r_p)! r_q! (l_q - 2r_q)!} \\
\times \frac{\left(1/2\right)^{l_p+q-(2r_p+2r_q)}}{\sqrt{Q}^{l_p-2r_p} \sqrt{Q}^{Q-2r_q}} \delta^i \delta^2 \frac{l_p! \left(l_p - 2r_p + l_q - 2r_q\right)\left(-1\right)^{r_p} \left(-1\right)^{r_q} \left(-1\right)^{l_p+q-(2r_p+2r_q)}}{r_p! (l_p - 2r_p)! r_q! (l_q - 2r_q)!} \\
= \left( \frac{\pi}{\gamma_p} \right)^{1/2} \left( \frac{\pi}{\gamma_q} \right)^{1/2} \left( \frac{\pi}{\gamma_q} \right)^{1/2} \left( \frac{\pi}{\gamma_q} \right)^{1/2} \sum_{r_p=0}^{(p/2)} \sum_{r_q=0}^{(q/2)} \frac{(-1)^{r_p+q-r_q}}{r_p! (l_p - 2r_p)! r_q! (l_q - 2r_q)!} \\
\times \frac{\left(1/2\right)^{l_p+q-(2r_p+2r_q)}}{\sqrt{Q}^{l_p-2r_p} \sqrt{Q}^{Q-2r_q}} \delta^i \delta^2 \frac{l_p! \left(l_p - 2r_p + l_q - 2r_q\right)\left(-1\right)^{r_p} \left(-1\right)^{r_q} \left(-1\right)^{l_p+q-(2r_p+2r_q)}}{r_p! (l_p - 2r_p)! r_q! (l_q - 2r_q)!} \\
\times \frac{\left(1/2\right)^{l_p+q-(2r_p+2r_q)}}{\sqrt{Q}^{l_p-2r_p} \sqrt{Q}^{Q-2r_q}} \delta^i \delta^2 \frac{l_p! \left(l_p - 2r_p + l_q - 2r_q\right)\left(-1\right)^{r_p} \left(-1\right)^{r_q} \left(-1\right)^{l_p+q-(2r_p+2r_q)}}{r_p! (l_p - 2r_p)! r_q! (l_q - 2r_q)!} \\
= \left( \frac{\pi}{\gamma_p} \right)^{1/2} \left( \frac{\pi}{\gamma_q} \right)^{1/2} \left( \frac{\pi}{\gamma_q} \right)^{1/2} \left( \frac{\pi}{\gamma_q} \right)^{1/2} \sum_{r_p=0}^{(p/2)} \sum_{r_q=0}^{(q/2)} \frac{(-1)^{r_p+q-r_q}}{r_p! (l_p - 2r_p)! r_q! (l_q - 2r_q)!} \\
\times \frac{\left(1/2\right)^{l_p+q-(2r_p+2r_q)}}{\sqrt{Q}^{l_p-2r_p} \sqrt{Q}^{Q-2r_q}} \delta^i \delta^2 \frac{l_p! \left(l_p - 2r_p + l_q - 2r_q\right)\left(-1\right)^{r_p} \left(-1\right)^{r_q} \left(-1\right)^{l_p+q-(2r_p+2r_q)}}{r_p! (l_p - 2r_p)! r_q! (l_q - 2r_q)!}
\]
Finally, \( C_{R I_x} \times I_{k_z} = \left( \frac{\pi}{\gamma_P} \right)^{1/2} \left( \frac{\pi}{\gamma_Q} \right)^{1/2} \left( \frac{\pi}{\delta} \right)^{1/2} \times \sum_{r_P=0}^{[l_P/2]} \sum_{r_Q=0}^{[l_Q/2]} \sum_{i=0}^{(l_P-2r_P+l_Q-2r_Q)/2} \frac{l_P! \gamma_{r_P-l_P}}{r_P!(l_P-2r_P)!} \times \frac{l_Q! \gamma_{r_Q-l_Q}}{r_Q!(l_Q-2r_Q)!} \times (-1)^i (2\delta)^{2(r_P+r_Q)} (l_P + l_Q - 2r_P - r_Q)! \delta^i P Q x^{l_P+l_Q-2(r_P+r_Q+i)} (4\delta)^{l_P+l_Q} (l_P + l_Q - 2r_P - 2r_Q - 2i)! \right) \) \( (4.130) \)

[ end of solution ]

Exercise.

From Eq. 4.130 of the previous exercise, extract the first line of square root constants. These will have identical form in \( C_{R I_y} \times I_{k_y} \) and \( C_{R I_z} \times I_{k_z} \). Thus, across \( C_{R I_x} I_{k_z} C_{R I_y} I_{k_y} C_{R I_z} I_{k_z} \), we will have the factor

\[
\left( \frac{\pi}{\gamma_P} \right)^{3/2} \left( \frac{\pi}{\gamma_Q} \right)^{3/2} \left( \frac{\pi}{\delta} \right)^{3/2}.
\]

Show that the product of this and the first factor that appears in the last line of \( G_{RI} \) (Eq. 4.129), \((\delta/\pi^2)\), is

\[
\frac{2\pi^2}{\gamma_P \gamma_Q} \left( \frac{\pi}{\gamma_P + \gamma_Q} \right)^{1/2}.
\]

Solution.

\[
\left( \frac{\pi}{\gamma_P} \right)^{3/2} \left( \frac{\pi}{\gamma_Q} \right)^{3/2} \left( \frac{\pi}{\delta} \right)^{3/2} \times \frac{\delta}{\pi^2} = \pi^3 (\gamma_P \gamma_Q)^{-3/2} \pi^{3/2} \cdot \delta^{-3/2} \frac{\delta}{\pi^2} = \pi (\gamma_P \gamma_Q)^{-3/2} \pi^{3/2} \cdot \delta^{-3/2} \frac{\delta}{\pi^2} \]

Because

\[
\delta = \left( \frac{1}{4\gamma_P} \right) + \left( \frac{1}{4\gamma_Q} \right) = \frac{4(\gamma_P + \gamma_Q)}{16\gamma_P \gamma_Q} = \frac{\gamma_P + \gamma_Q}{4\gamma_P \gamma_Q},
\]

then

\[
\delta^{-1/2} = \left( \frac{4\gamma_P \gamma_Q}{\gamma_P + \gamma_Q} \right)^{1/2}.
\]

131
Thus,

\[
\pi (\gamma_P \gamma_Q)^{-3/2} \pi^{3/2} \left( \frac{4 \gamma_P \gamma_Q}{\gamma_P + \gamma_Q} \right)^{1/2} = \frac{2 \pi^2}{\gamma_P \gamma_Q \left( \gamma_P + \gamma_Q \right)^{1/2}}.
\]

[ end of solution ]

Consider \( C_{RI_k} \times I_{k_x} \) (Eq. 4.130) with the factors featuring \( \pi \) removed, as in the last exercise. When we multiply this by the \( x \)-component coefficients \( c_{l_P} \) and \( c_{l_Q} \), which are factors of \( G_{RI} \) in the expression for \( G \) (Eq. 4.108), then we have what we might call \( C'_{RI_k} \times I'_{k_x} \):

\[
C'_{RI_k} \times I'_{k_x} = \sum_{r_P=0}^{[l_P/2]} \sum_{r_Q=0}^{[l_Q/2]} \sum_{i=0}^{[(l_P-2r_P+l_Q-2r_Q)/2]} \frac{l_P! \gamma_P^{r_P-l_P} \cdot c_{l_P}}{r_P! (l_P - 2r_P)!} \times \frac{l_Q! \gamma_Q^{r_Q-l_Q} \cdot c_{l_Q}}{r_Q! (l_Q - 2r_Q)!} \\
\times \frac{(-1)^{l_P} (2\delta)^{2(r_P+r_Q)} (l_P + l_Q - 2r_P - r_Q)! \delta^l \bar{P} Q_x^{l_P+l_Q-2(r_P+r_Q+i)}}{(4\delta)^{l_P+l_Q} i! (l_P + l_Q - 2r_P - 2r_Q - 2i)!}
\]

\[
= \sum_{r_P=0}^{[l_P/2]} \sum_{r_Q=0}^{[l_Q/2]} \sum_{i=0}^{[(l_P-2r_P+l_Q-2r_Q)/2]} \theta_P \times \theta_Q \\
\times \frac{(-1)^{l_P} (2\delta)^{2(r_P+r_Q)} (l_P + l_Q - 2r_P - r_Q)! \delta^l \bar{P} Q_x^{l_P+l_Q-2(r_P+r_Q+i)}}{(4\delta)^{l_P+l_Q} i! (l_P + l_Q - 2r_P - 2r_Q - 2i)!}
\]

where

\[
\theta = \frac{l! \gamma_P^{r_l} \cdot c_{l}}{r! (l - 2r)!}
\]

(4.132)
As such, by incorporating the results of the previous exercise and the new definitions $C'_{RI}$ and $I'_k$,

$$G = \tilde{K}_P \tilde{K}_Q$$

\[
\times \sum_{l_p=0}^{l_A+l_B} \sum_{m_p=0}^{m_A+m_B} \sum_{n_p=0}^{n_A+n_B} c_{l_p} c_{m_p} c_{n_p} \\
\times \sum_{l_q=0}^{l_A+l_B} \sum_{m_q=0}^{m_A+m_B} \sum_{n_q=0}^{n_A+n_B} c_{l_q} c_{m_q} c_{n_q} \\
\times G_{RI}
\]

\[
= \frac{2\pi^2}{\gamma_P \gamma_Q \left( \frac{\pi}{\gamma_P + \gamma_Q} \right)^{1/2}} \tilde{K}_P \tilde{K}_Q \\
\times \sum_{l_p=0}^{l_A+l_B} \sum_{l_q=0}^{l_A+l_B} C'_{RI_x} \times I'_k \\
\times \sum_{m_p=0}^{m_A+m_B} \sum_{m_q=0}^{m_A+m_B} C'_{RI_y} \times I'_k \\
\times \sum_{n_p=0}^{n_A+n_B} \sum_{n_q=0}^{n_A+n_B} C'_{RI_z} \times I'_k \\
\times F_\nu \left( \overrightarrow{PQ}^2 / (4\delta) \right)
\]

(4.133)

Only a few steps remain.

**Exercise.**

Complete the derivation of $G$ by performing the following rearrangements.

1. Extract the summation symbols from every $C'_{RI}$ and $I'_k$, and fuse the remaining part of $C'_{RI}$ and $I'_k$ to create expressions labelled $g_x$, $g_y$, and $g_z$.

2. Move the summation symbols into Eq. 4.133, alongside $g_x$, $g_y$, and $g_z$, so that there are five summation symbols per coordinate for total of 15.

3. Collect the terms containing $\pi$ and $\tilde{K}$ (i.e., the first line in the last equality of Eq. 4.133) into a new term called $\Omega$.

4. Show the final expression for $G$ in terms of this rearrangement of Eq. 4.133 and its new set of definitions.
5. As a separate, one-line equation, show the expression for $(AB|CD)$ in terms of $G$.

Solution.

**Essential Equation**

$$(AB|CD) = \sum_{a=1}^{K_a} \sum_{b=1}^{K_b} \sum_{c=1}^{K_c} \sum_{d=1}^{K_d} d_a d_b d_c d_d N_a N_b N_c N_d \times G$$

**Essential Equation**

$$G = \Omega \sum_{l_p=0}^{l_A+l_B} \sum_{l_Q=0}^{l_C+l_D} \sum_{r_p=0}^{l_p+l_Q} \sum_{r_Q=0}^{l_A+l_B} \sum_{i=0}^{l_p+l_Q-2r_p-2r_Q} g_x$$

$$\times \sum_{m_p=0}^{m_A+m_B} \sum_{m_Q=0}^{m_C+m_D} \sum_{s_p=0}^{m_p+l_Q} \sum_{s_Q=0}^{l_A+l_B} \sum_{j=0}^{l_p+l_Q-2r_p-2r_Q} g_y$$

$$\times \sum_{n_p=0}^{n_A+n_B} \sum_{n_Q=0}^{n_C+n_D} \sum_{t_p=0}^{n_p+l_Q} \sum_{t_Q=0}^{l_A+l_B} \sum_{k=0}^{l_p+l_Q-2r_p-2r_Q} g_z$$

$$\times F_\nu(|\vec{PQ}|^2/4\delta)$$

$$\nu = l_p + l_Q + m_p + m_Q + n_p + n_Q - 2(r_p + r_Q + s_p + s_Q + t_p + t_Q) - (i + j + k)$$

$$\delta = \frac{1}{4\gamma_P} + \frac{1}{4\gamma_Q}$$

where

**Essential Equation**

$$\Omega = \frac{2\pi^2}{\gamma_P \gamma_Q} \left( \frac{\pi}{\gamma_P + \gamma_Q} \right)^{1/2} \exp \left( -\frac{\alpha_a \alpha_b \vec{A}^2}{\gamma_P} - \frac{\alpha_c \alpha_d \vec{C}^2}{\gamma_Q} \right)$$

and for the $x$-components
\[ g_x \equiv g_{l_P,l_Q,r_P,r_Q,i}(l_A,l_B,\vec{A}_x,\vec{B}_x,\vec{P}_x,\vec{Q}_x,\gamma_P;l_C,l_D,\vec{C}_x,\vec{D}_x,\vec{Q}_x,\gamma_P) \]

\[ = (-1)^{l_P} \cdot \theta(l_P,l_A,l_B,\vec{P}A_x,\vec{P}B_x,r,\gamma_P) \cdot \theta(l_Q,l_C,l_D,\vec{Q}C_x,\vec{Q}D_x,r_Q,\gamma_Q) \]

\[ \times \left( -1 \right)^i \left( 2\delta \right)^2 (l_P + l_Q - 2r_P - 2r_Q)\delta^i \left( -1 \right)^{l_P + l_Q - 2(r_P + r_Q + i)} \]

\[ \frac{(4\delta)^{l_P + l_Q + i} l_P! l_Q!}{(l_P + l_Q - 2(r_P + r_Q + i))!} \]

\[ \theta(l,l_A,l_B,a,b,\gamma) = c_i(l_A,l_B,a,b)^{l!} \gamma^{r-1} \frac{l!(l-2r)!}{r!(l-2r)!} \]  

(4.138)

with analogous expressions for the other two \( g \) factors.

[ end of solution ]

Congratulations! You have derived the code-able solutions to the molecular integrals. You are ready to proceed to Handout 2 (or Handout 3 if you are an experienced programmer) and implement them in computer code.
As a first program, create a new file called `myfirstprogram.py`, and then in either your text editor or IDE type in the following line of code:

```python
print('Hello, world!')
```

Run `myfirstprogram.py` by clicking the “Run” button in your IDE. Alternatively, double click on the saved file from a folder. If you work in a terminal, then enter the following at the `$` prompt.

```
$ python myfirstprogram.py
```

You should see
printed to your screen. Congratulations, you have coded a complete, functioning program!

Now things get a little more complicated with it comes to implementing molecular integrals. However, we will present only the minimum, essential features of Python that you need to complete the programming projects. There are many techniques and functionalities of Python that would allow one to complete the project in a more sophisticated manner (tuples, object-oriented programming, etc.); the reader familiar with such techniques is encouraged to employ them.

Introducing the Python Types, Nested Arrays, and Printing

For the molecular integrals and Hartree–Fock programming projects, you will need to store different kinds of data. Python provides several types of data. The data types that we will be concerned with, and some examples of each, are:

- **Integer**: Numbers with no decimal point.
  
  3, -2, 118

- **Float**: Numbers with a decimal point.
  
  -1.00, 3.14159, 6.022e23, 6.626e-34

- **String**: Non-numerical characters. (Notice that '1234' is a string and not an integer, because it is in quotes.)
  
  'H', 'Li', '1234'

- **Dictionary**: In curly brackets, a dictionary is made up of keys and values separated by a colon.
  
  {'H': 1.0094, 'He': 1.9998, 'Li': 3.001}

- **Array**: A collection marked by square brackets, made up of elements separated by commas.
  
  [0.0004, 1.0, 0.34523425]
The fact that arrays themselves are types means that arrays can be **nested** within arrays, like so:

\[
[[0.0000, 0.0000, 0.45403], [0.0000, 2.70672, -1.81610], [0.0000, -2.70672, -1.81610]]
\]

This **nested array** is the set of \( x, y, z \) coordinates of water, as shown in the input .xyz file in Fig. 4.17. The first element of the outer array is itself an “inner” array made up individual elements giving the \( x \), \( y \), and \( z \) positions (in units Bohr), respectively, of the oxygen atom; the second element is an array made up of the \( x \), \( y \), and \( z \) positions of the first hydrogen atom; the third element contains an array for the \( x, y, z \) coordinates of the second hydrogen atom.

<table>
<thead>
<tr>
<th>3</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>O</td>
<td>0.00000 0.00000 0.45403</td>
</tr>
<tr>
<td>H</td>
<td>0.00000 2.70672 -1.81610</td>
</tr>
<tr>
<td>H</td>
<td>0.00000 -2.70672 -1.81610</td>
</tr>
</tbody>
</table>

Figure 4.17: The first line of an .xyz file is the number of atoms present in the input molecule. The second line is either empty or holds text, such as **this is the coordinates of water in units Bohr**. The remaining lines feature a different atom on each line represented in the first column by the atom’s atomic symbol, followed by their \( x \), \( y \), and \( z \) coordinates.

Each of these described types can be assigned to **variables**. For example, to **declare** an array-assigned variable \( d \), which will hold nested arrays of contraction coefficients that form part of the STO-3G basis set, the code would be:

```python
# STO-3G contraction coefficients pulled from EMSL; applies to all atoms
d = [[0.44463454, 0.53532814, 0.15432897], [0.70011547, 0.39951283, -0.09996723],
     [0.39195739, 0.60768372, 0.15591627]]
```

We have added a **comment** to a line before the array. Any code or text that appears after a pound sign is inactive. This is very useful for writing notes to remind yourself what certain code means. (It may be obvious what the code means to you now, but come back to it in two weeks and you may be scratching your head.) Here we remind ourselves that \( d \) is the nested array holding contraction coefficients, which apply to all atoms. We can also reformat the code like so:

```python
# STO-3G contraction coefficients pulled from EMSL; applies to all atoms
d = [[0.44463454, 0.53532814, 0.15432897], # contraction coefficients for 1s orbitals
     [0.70011547, 0.39951283, -0.09996723], # contraction coefficients for 2s orbitals
     [0.39195739, 0.60768372, 0.15591627]] # contraction coefficients for 2p orbitals (x, y, or z)
```
Parentheses or brackets permit breaking up on continuous line of code over several lines, which can aid visualization. We also remind ourselves, with comments, that the first element of the outer array is the three contraction coefficients for the 1s orbitals, that the second element of the outer array contains the contraction coefficients for the 2s orbitals, and so on.

You can print a string of the second element of the alpha array by including in your code the following line:

```python
print(d[1])
```

Running this Python program will then display

```
[0.70011547, 0.39951283,-0.09996723]
```

Note that Python uses zero-based indexing: It counts beginning at the number 0, rather than the number 1. Hence, d[1] is actually the second element of d. You can also write

```python
print(d[1,2])
```

to print just the third contraction coefficient of the 2s orbitals.

We can also generate a more detailed, user-friendly output by adding together strings in the print command argument (content within parenthese).

```python
print("The third contraction coefficient for a 2s orbital is: " + str(d[1,2]) )
```

Because we coupled a string (The third contraction coefficient ...) with data that is not of string-type (the array d), we needed to add the str keyword around d[1,2]. Running this program will then display

```
The third contraction coefficient for a 2s orbital is: -0.09996723.
```

When building the atomic orbital basis, regular use will made be of dictionaries, one of the Python types introduced earlier. We will create a dictionary that converts atomic symbols (which appear at the beginning of all but the first two lines of .xyz input files; see Fig. 1) to atomic numbers:
Here, the key 'Li' has a value of 3. If one wished to create an integer variable called `atnumBoron` holding the atomic number of boron, we would write the following code:

```python
atnumBoron = Z['B']
```

Try printing `atnumBoron`, and confirm it gives the atomic number of boron.

Python is not generally sensitive to spacing. For example,

```python
d = [[0.44463454, 0.53532814, 0.15432897],
     [0.70011547, 0.39951283, -0.09996723],
     [0.39195739, 0.60768372, 0.15591627]]  # contraction coefficients for 2p orbitals (x, y, or z)
```

functions the same as

```python
d = [
    [0.44463454, 0.53532814, 0.15432897],
    [0.70011547, 0.39951283, -0.09996723],
    [0.39195739, 0.60768372, 0.15591627]
]
```

One important exception to Python's non-restrictive treatment of spacing are functions, which will be discussed later.

**Handling Matrices with NumPy**

While it is possible to use a Python two-dimensional array (or nested array) as a matrix, matrices and matrix operations are most easily and powerfully handled with the NumPy library. Computational quantum chemistry is replete with matrices: the overlap and one-electron integral matrices are two-dimensional matrices; the electron repulsion integral matrices are of four-dimensions. To use NumPy (which needs to have been installed on your computer), the following line is added to the top of a code file:

```python
import numpy as np  # import the NumPy library
```

By importing NumPy, we can also calculate exponential functions and use a precise value of π. For example, to calculate the value $e^\pi$ would be `np.exp(np.pi)`. 
Here is how to create a two-dimensional matrix. The following simple example features the number 1 along the diagonal, with all non-diagonal elements being equal to 0.

```python
# construct a matrix with diagonal elements = 1
mymatrix = np.matrix([[1,0,0],[0,1,0],[0,0,1]])
print(mymatrix)
```

Running this code will display

```
[[1 0 1]
 [0 1 0]
 [0 0 1]]
```

Typically, we will create an empty matrix the dimensions of which are determined by the number $K$ of atomic orbitals present in the basis. For water, the minimal basis configuration describes

$$K = \phi_{H1s} + \phi_{H1s} + \phi_{O1s} + \phi_{O2s} + \phi_{O2p_x} + \phi_{O2p_y} + \phi_{O2p_z} = 7$$

atomic orbitals in the basis. Thus, the overlap integral matrix $S$ is $K \times K = 7 \times 7$ in dimension.

The corresponding code for generating an empty $S$ matrix of $K \times K$ dimension is

```python
K = 7 # the number of atomic orbitals in the basis
S = np.zeros((K,K)) # create an empty 7x7 matrix to be filled in later
```

The elements of $S$ will later be accessed by loops and modified by functions. The two-electron integrals make up a rank-4 tensor of $K \times K \times K \times K$ dimensions.

```python
# K was already set to 7, so the following line will create a 7x7x7x7 tensor filled with zeros
G = np.zeros((K,K,K,K))
```

For an even simpler molecular system like $H_2$, for which 2 atomic orbitals ($K = 2$) form the basis when using a minimal basis set like STO-3G, then

```python
print(S)
```

gives
and

```python
print(G)
```
gives

```
[[[[ 0.  0.]
[ 0.  0.]]

[[ 0.  0.]
[ 0.  0.]]

[[ 0.  0.]
[ 0.  0.]]

[[ 0.  0.]
[ 0.  0.]]

[[ 0.  0.]
[ 0.  0.]]
```

We will now discuss how to access individual matrix elements, so that we can calculate their values using the proper formulas.

**Functions and Loops**

Loops allow us to perform repetitive tasks without having to repeat code, which can be error-prone. A simple example is having our quantum chemistry program calculate how many electrons are present in the molecule. (Remember that the orbitals of our minimal basis provide the space for electrons to occupy, and orbitals are used in the molecular integral calculations, but it is in the Hartree–Fock method that electrons are effectively “inserted” into these atomic orbitals, which are then mixed to form molecular orbitals. The Hartree–Fock procedure requires knowing the number of electrons to form the molecular orbitals from the atomic orbitals.) We can calculate the number
of electrons assuming a neutrally charged molecule, so that the number of electrons is simply the sum of atomic numbers.

We earlier created a dictionary $Z$ containing keys that are atomic symbols (strings), and values that are atomic numbers (integers). Assume that we have an array called `atoms`, the array’s elements of which are the atomic symbol strings for each atom present in the input molecule. For example, having read the input file for water, shown in Fig. 4.17, the `atoms` array would be

```python
atoms = ['O', 'H', 'H']
```

This array would be automatically constructed while the input file is being read in by the code. (Reading input files will be addressed in the Guided Exercises.)

We can take advantage of the fact that each element of `atoms` is a string that corresponds to some key within $Z$. We can then use the value corresponding to that key, which is the atomic number integer, to sum the total number of electrons. This is shown in the following code.

```python
atoms = ['O', 'H', 'H']
n = 0 # this integer variable will sum the atomic numbers
for A in atoms:
    n = n + Z[A] # add to whatever n already equals the atomic number of index A
print(n)
```

While $n = n + Z[A]$ is of course not algebraically correct, in computer code it serves to update the variable $n$.

Notice that the code within the for loop is indented; this is necessary to designate those lines which are part of the loop, and separate it from those that are not. The first line following the for line that is not indented (here, the `print` statement) then lies outside of whatever task the loop is executing. (See for yourself what happens when the print statement is included inside the loop versus when it is outside the loop.)

For the current step in the loop through each element of `atoms`, $A$ is a variable equal to that element. So in the first loop through atoms, $A = 'O'$. Thus, $Z[A]$ is equivalent to saying $Z['O']$, and this returns the integer 8. The integer 8 is added to $n$ (which initially equals 0), and then the next cycle is begun. $A$ now equals 'H', and so $Z['H']$ returns 1, and this is added to $n$, for a total of 9. The final step in the cycle adds the remaining 1 to $n$. With `atoms` now exhausted, the for loop
ends, having determined for us that the number of protons (and hence the number of electrons) in our neutrally charged molecule is 10. This is what would be printed to the computer screen.

Here is another, slightly different way of using a loop. We will often need to compute the distance between two atoms. Let R be a nested array holding the coordinates of the H₂ molecule: Element one for the first hydrogen atom, and element two for the other hydrogen. (Normally, R will automatically be created while the .xyz input file is being read in, as would atoms. For now we assign the coordinates directly.)

```
# define coordinates of the H2 molecule
R = [[0.0, 0.0,-0.8],
     [0.0, 0.0, 0.8]]
RA = R[0] # coordinates of the first hydrogen
RB = R[1] # coordinates of the second hydrogen
```

In this code we also instantiated (or “created”) the variables RA and RB to separately hold the coordinates of the individual atoms, which will make this demonstration a little clearer. As we saw in the main article and in Handout 1, the equation for the squared distance between orbitals φ_A (centered on atom A at \(\vec{A}\)) and φ_B (centered at \(\vec{B}\)), is

\[
|\vec{AB}|^2 = |\vec{A} - \vec{B}|^2 = (\vec{A}_x - \vec{B}_x)^2 + (\vec{A}_y - \vec{B}_y)^2 + (\vec{A}_z - \vec{B}_z)^2 .
\] (4.139)

One straightforward way of computing this would be

```
ABsquared = (RA[0]-RB[0])**2 + (RA[1]-RB[1])**2 + (RA[2]-RB[2])**2
```

Recall that RA[0] would be the x-coordinate of atom A, RA[1] the y-coordinate, and RA[2] the z-coordinate; likewise for RB. ABsquared is a variable that holds a float, a number that is the result of using Eq. 4.139 as implemented in the code. The distance between \(\vec{A}\) and \(\vec{B}\) for each coordinate is in parentheses, and the squaring of each coordinate difference is achieved with **2. (Cubing can be achieved with **3, finding the square root with **(1/2), and so on.)

Alternatively, we could employ a for loop.
What distinguishes this for loop from our previous example is that the index \( i \) is simply a counting, integer variable, rather than being equal to an element of an array. Thus, in the range from 0 up to (but not including!) 3, \( i \) will be equal to: 0 in the first pass, and then 1 in the second pass, and end with 2. If we didn’t know that the “length” (the number of elements) of the array was 3, we could have also written

```python
ABsquared = 0.0  # instantiate the variable ABsquared, its quantity to be calculated in a for loop
for i in range(0, len(RA)):  # instantiate the variable ABsquared, its quantity to be calculated in a for loop
    ABsquared = ABsquared + (RA[i] - RB[i])**2
```

We’re assuming here that \texttt{len(RA)} – that is, the length of RA – is equal to RB.

Finally, loops also allow us to systematically access the elements of a matrix (as of \( S, T, V, \) and \( G \)). To demonstrate this, we will prepare a simple program that computes the overlap integrals for \( \text{H}_2 \) using the STO-1G basis set. Before showing how loops allow us to access matrix elements, we first consider some details of what the program will entail, and discuss functions.

As Fig. 4.18 below shows, the overlap matrix will have four matrix elements: the overlap of orbital \( \phi_A \) (on one hydrogen) with itself; the overlap of orbital \( \phi_A \) with orbital \( \phi_B \) (which is on the other hydrogen); the overlap of orbital \( \phi_B \) with orbital \( \phi_A \) (this will give the same value as the overlap of orbital \( \phi_A \) with orbital \( \phi_B \)); and the overlap of orbital \( \phi_B \) with itself. These integrals correspond to specific values of the \( S \) (or coded \( S \)) matrix.

<table>
<thead>
<tr>
<th>( \phi_A )</th>
<th>( \phi_B )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \phi_A )</td>
<td>( \int \phi_A \phi_A , d\tau )</td>
</tr>
<tr>
<td>( \phi_B )</td>
<td>( \int \phi_B \phi_A , d\tau )</td>
</tr>
</tbody>
</table>

\( \phi_A \) \quad \phi_B \quad \phi_A \quad \phi_B \equiv \quad \phi_A \quad S[0,0] \quad S[0,1] \quad \phi_B \quad S[1,0] \quad S[1,1]

Figure 4.18: The \( 2 \times 2 \) overlap integral matrix \( S \) for the 1s orbital \( \phi_A \) on hydrogen atom \( A \), and the 1s orbital \( \phi_B \) on hydrogen atom \( B \).

Because orbitals are normalized, we should expect that the overlap integrals of orbitals with themselves (which make up the diagonal elements of the matrix) equal 1; the rest will be some value between, or equal to, 0 and 1. Knowing these facts will form a good basic check test of our
code. In the STO-1G basis set for hydrogen, there is only one contraction coefficient \(d_p = 1.0\), and only one \(\alpha_p = 0.28294\).

The equation for an overlap integral matrix element of two identical, 1s orbitals can easily be derived from Eqs. 4.144 and 4.143 of Handout 1. It is

\[
S_{AB} = N^2 \times \exp \left[ - \left( \frac{\alpha^2}{2\alpha} \right) |\vec{AB}|^2 \right] \times \left( \frac{\pi}{2\alpha} \right)^{3/2}, \tag{4.140}
\]

where the contraction coefficient \(d = 1.0\) is implied. The normalization constant \(N\) is

\[
N = \left( \frac{2\alpha}{\pi} \right)^{3/4}. \tag{4.141}
\]

We discussed earlier the mathematical form of \(|\vec{AB}|^2\), and how to code it.

Let’s define a function that allows us to call the calculation of \(AB^2\) at any time, so that if we need to we can use it more than once. We then use the function to find the distance between the two atoms specified in \(R\). Finally, we will print the result.

```python
def compute_ABsq(RA, RB):
    # This function compares the square of the distance between two atoms, or two atom-centered orbitals.
    # It follows Eq. 4.141 of Handout 2. RA and RB are arrays of the coordinates of atoms (or orbitals) A and B, 3 elements each.
    ASquared = 0.0
    for i in range(len(RA)):  # equivalent to range(0, len(RA))
        ASquared += (RA[i] - RB[i])**2
    return ASquared

H2distance = compute_ABsq(R[0], R[1])  # equivalent to passing RA and RB to the function
print("Distance between the two atoms of dihydrogen = " + str(H2distance))
```

We created the function with the “define function” keyword `def`. The keyword is followed by what we want to call the function, `compute_ABsq`. We then say the function has two arguments, \(RA\) and \(RB\), which are passed in to the function for the function to use.

We also have an expanded comment type, created by blocking text between three quotation marks. Here the comment indicates that \(RA\) and \(RB\) are both expected to be arrays; if they are not, the function will return an error when running the program. For example, if we accidentally pass in a float instead of an array, we would get the error
That is, it has encountered a Python data type error. On the other hand, if we do pass in an array, but it doesn’t have enough elements for what the loop “expects”, we will get the following error:

```
Index Error: list index out of range
```

A comment after the first line of the for loop indicates that the 0 at the beginning of `range` is 0 by default and not necessary to explicitly include. Of course, any lower bound of a range other than 0 must be explicitly declared.

Just as with a loop, every line that’s indented after `def compute_ABsq(RA, RB):` is a part of the function. When the text returns to non-indented form, it is outside the function. The line `H2distance = compute_ABsq(R[0], R[1])` is outside the function, and is *calling* the function `compute_ABsq`. It is passing into `compute_ABsq` the positions of the first hydrogen atom and the second hydrogen we defined for our H₂ molecule in R, which you will recall is a nested array. What is *returned* by the function (ABsquared) is stored in the variable `H2distance`. The print statement results in the display

```
Distance between the two atoms of dihydrogen = 1.6
```

when the code is run.

Similarly, the functions for computing the normalization constant and the overlap integral would be

```python
def compute_norm(alphaconstant):
    """
    Calculate and return the normalization constant N, according to Eq. of Handout 2.
    """
    N = ((2*alphaconstant / np.pi)**(3/4))
    return N

def compute_overlap(a,RA,RB):
    """
    Calculate a matrix element of the S matrix by calling on compute_norm and compute_ABsq.
    """
    Selement = (compute_norm(a)**2 * np.exp(-a**2 / (2*a)) * compute_ABsq(RA,RB)) * (np.pi / (2*a)**(3/2))
    return Selement
```
This code also reveals that one can call another function from within a function. In the function `compute_overlap` we call `compute_norm`, passing into it whatever value for `alpha` we have. We also call the distance function `compute_ABSq`, passing into it the $x, y, z$ coordinates of atom A (in the form of the array RA) and the $x, y, z$ coordinates of atom B (array RB). Note that the variable name for $\alpha$ is `alphaconstant` in the `compute_norm` function, but `a` in the `compute_overlap` function. This is perfectly acceptable, as long as the same variable name is used within a function. Make sure that you can see the direct connections between this code and Eqs. 4.140 and 4.141.

It only remains to fill the empty matrix $S$ using these functions. We will do so with a loop. Let’s first create a simple matrix other than $S$ for demonstration, and call it `mymatrix`. Assuming we’ve imported NumPy, the code

```python
mymatrix = np.matrix([[1,2],[3,4]])
print(mymatrix)
```

gives

```
[[ 1 2 ]
 [ 3 4 ]]
```

If we apply a single loop to `mymatrix`, and have an integer variable called `counter` to keep track of which loop cycle we are in, then,

```python
counter = 1 # track where we are in the loop
for A in range(0, len(mymatrix)):
    print("Loop " + str(counter) + " : mymatrix[" +str(i) + "] = " + str(mymatrix[i]))
    counter = counter + 1
```

Notice that we set the upper bound on our range to the length of the matrix, which would be just one dimension of the matrix (the number of columns). This is okay, and it will be for all of our molecular integral matrices, because they are square, $K \times K$ matrices (the number of rows is equal to the number of columns). The display of this computation would be

```
Loop 1: [[ 1. 2.]]
Loop 2: [[ 3. 4.]]
```

This would be fine if we wanted entire rows of matrices. But usually we want individual matrix elements. To access elements, we must nest loops.
mymatrix = np.matrix([[1,2],[3,4]])

counter = 1 # track where we are in the nested loop
for A in range(len(mymatrix)):
    for B in range(len(mymatrix)):
        print("Loop" + str(counter) + ": mymatrix[" + str(A)+ "," + str(B) + "] = " + str(mymatrix[A,B]))
        counter = counter + 1

As you can see, deeper indenting is required when nesting loops: one indentation per loop. In this
looping scheme, the row of mymatrix is accessed with index A, the first row being A = 0. Then
within this row, each element of row A is accessed with index B. A matrix element corresponding
to row A and column B, then, is specified by mymatrix[A,B]. Once every possible element (or
“column”) B is accessed within row A, then the next row (A + 1) is accessed. Thus, the output is

<table>
<thead>
<tr>
<th>Loop 1: mymatrix[0,0] = 1.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Loop 2: mymatrix[0,1] = 2.0</td>
</tr>
<tr>
<td>Loop 3: mymatrix[1,0] = 3.0</td>
</tr>
<tr>
<td>Loop 4: mymatrix[1,1] = 4.0</td>
</tr>
</tbody>
</table>

Additionally, we can perform tasks at different levels of the nested loop. So, if we wanted to print
that the end of a row of mymatrix was reached, we would do so by writing code at the appropriate
level of indentation, that of the “outer” loop.

counter = 1 # track where we are in the nested loop
for A in range(len(mymatrix)):
    for B in range(len(mymatrix)):
        print("Loop " + str(counter) + ": mymatrix[" + str(A)+ "," + str(B) + "] = " + str(mymatrix[A,B]))
        counter = counter + 1
    print("The end of row " + str(A+1) + " has been reached.") # A+1 to account for zero-based indexing

The output for this code is
To access elements of an $N$-dimensional matrix, we require the nesting of $N$ loops: two loops for a two-dimensional matrix, one loop for a one-dimensional matrix (or vector), four loops for a four-dimensional matrix (or rank-4 tensor), and so on.

Now that we’ve illustrated accessing matrix elements, we will apply this technique to filling in our empty $S$ overlap integral matrix. Recalling that hydrogen has one $\alpha = 0.28294$ in the STO-1G basis set, and that we already constructed a function for computing a matrix element, then

$$H_{i\alpha} = 0.28294 \quad \# \text{exponential factor for hydrogen 1s orbital in a STO-1G basis}$$

```
for A in range(len(S)):
    for B in range(len(S)):
        S[A,B] = compute_overlap(H1salpha,R[A],R[B])
```

First, the overlap when $A = B = 0$ is calculated: This is the overlap of $\phi_A$ with itself. Then, with $A$ unchanged, $B$ is incremented to 1, and the overlap of $\phi_A$ with $\phi_B$ is calculated. $B$’s length does not go beyond 1, so then $A$ is incremented by 1 (to $A = 1$) and $B$ begins again at 0: The overlap of $\phi_B$ with $\phi_A$ is calculated. Finally, with $A$ still at $A = 1$, $B$ is incremented to 1, such that $A = B = 1$ and the overlap of $\phi_B$ with itself is computed.

Let’s put this discussion in context by presenting all of the code constituting a complete program that computes the overlap integral matrix of dihydrogen with the STO-1G basis set. Dropping some of the intermediate print statements we included for demonstration earlier, this complete program would be
import numpy as np

R = [[0.0, 0.0, -0.8],
     [0.0, 0.0, 0.8]]

def compute_ABsq(RA, RB):
    
    This function compares the square of the distance between two atoms, or two atom-centered
    orbitals.
    It follows Eq. 4.141 of Handout 2. RA and RB are arrays of the coordinates of
    atoms (or orbitals) A and B, 3 elements each.
    
    ABsquared = 0.0 # instantiate the variable ABsquared, its quantity to be calculated in a for
    loop
    for i in range(0,len(RA)):
        ABsquared = ABsquared + (RA[i] - RB[i])**2
    return ABsquared

def compute_norm(a):
    
    Calculate and return the normalization constant N, according to Eq. 4.143 of Handout 2.
    
    N = (2*a/np.pi)**(3/4)
    return N

def compute_overlap(a,RA,RB):
    
    Calculate a matrix element of the S matrix by calling on compute_norm and compute_ABsq.
    Follows Eq. 4.142 of Handout 2.
    
    S = (compute_norm(a))**2 * np.exp(-(a**2/(2*a)) * compute_ABsq(RA,RB)) * (np.pi/(2*a))**(3/2)
    return S

H1salpha = 0.28294
S = np.zeros((2,2)) # create an empty 2x2 S matrix

# access and calculate each matrix element
for A in range(len(S)):
    for B in range(len(S)):
        S[A,B] = compute_overlap(H1salpha,R[A],R[B])

print("Overlap integral matrix (S) =
" + str(S))

The \n in the print statement pushes whatever comes next to the following line, so that the printout of this code looks like

\begin{verbatim}
Overlap integral matrix (S) =
[[ 1.  0.91343706]
 [ 0.91343706 1. ]]\end{verbatim}

This reveals the overlap integrals for H\textsubscript{2} in the STO-1G basis, and can be represented as shown in Fig. 4.19.

<table>
<thead>
<tr>
<th>( \phi_A )</th>
<th>( \phi_B )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \phi_A )</td>
<td>( \int \phi_A \phi_A , d\tau )</td>
</tr>
<tr>
<td>( \phi_B )</td>
<td>( \int \phi_B \phi_A , d\tau )</td>
</tr>
</tbody>
</table>

\( \rightarrow \)

<table>
<thead>
<tr>
<th>( \phi_A )</th>
<th>( \phi_B )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \phi_A )</td>
<td>1.000</td>
</tr>
<tr>
<td>( \phi_B )</td>
<td>0.913</td>
</tr>
</tbody>
</table>

Figure 4.19: The 2 \times 2 overlap integral matrix S for the dihydrogen molecule, with a STO-1G basis set, and interatomic distance of 1.6 \( a_0 \).

As expected, the diagonal elements representing self-overlap are equal to exactly 1. All values of
this matrix can be considered as percentages of overlap, so that while an orbital obviously overlaps 100% with itself, one hydrogen 1s orbital overlaps with 91.3% of the other hydrogen’s 1s orbital. This high degree of overlap is due to the identical orbital symmetry, as well as the fact that each orbital has the same energy. (Atomic orbitals with very different energies, such as between the core s orbital of a sulfur atom and one of its valence p orbitals have little to no overlap due to the disparities in energy, as well as different symmetries.)

The Final Skill: Conditional Logic (If Statements)

Most of the molecular integrals programming project is dedicated to accessing matrix elements with loops (as shown earlier) and calling on functions (also discussed earlier) to execute some task that is frequently used, like calculating a matrix element. There is one final tool we need in our programming toolkit to implement all of the molecular integrals.

If statements are a powerful means of giving the programmer greater control over their code. It allows them to execute line(s) of code only under the condition that some criterion is met: If something is true, then perform this action. (By implication, if that something is not true, then don’t perform the action.) If statements are one kind of conditional logic that programmers frequently use.

There are only a few cases where we will use if statements in our programming projects. Here is one example. Recall from the discussion of the Hartree–Fock method in the main article that it is iterative: The molecular orbital coefficients \( \mathbf{C} \) are tweaked step-by-step (in a loop) until the change in total energy of the molecular system drops below some threshold. The energy changes usually get smaller with every step. We call this threshold, where the change in energy is sufficiently small, convergence. Usually it is set to a very small number like \( 1.0 \times 10^{-7} \) Hartrees.
def compute_HF(electroncount, basissize, S, T, V, G):
    
    Calculate electronic energy iteratively by the Hartree-Fock method. 
    Depends on number of electrons, the number of orbitals in the basis (basissize), and the 
    molecular integrals.
    
    # code to set up HF iteration as a for loop
    convergence = 1.0e-7
    for iteration in range(0, 20):  # stop looping after 20 iterations if HF doesn’t converge
        # code to varationally optimize the molecular orbitals
        Etot = Eel + Enn  # sum electronic energy and nuclear-nuclear repulsion energy
        if abs(Etot - old_Etot) < convergence:
            break

You will learn in the Hartree–Fock programming project (Handout 4) all of the details that go 
into this code, including what goes into # code to .... We are presently concerned with just two 
lines:

    if abs(Etot - old_Etot) < convergence:
       break

These two lines of code command the program to “break” out of the Hartree–Fock for loop if the 
absolute difference in energy between the present iteration energy (Etot) and the previous iteration 
energy (old_Etot) is less than $1.0 \times 10^{-7}$ Hartrees. By implication, if the change in energy is greater 
than or equal to $1.0 \times 10^{-7}$ Hartrees, continue looping.

Here is another important example. At the beginning of the programming project, our (minimal) 
atomic orbital basis will be built by passing through the list of atoms (which are represented by 
string atomic symbols) specified in the atoms array, drawn from the input .xyz file. For each atom, 
we append, or add, to an array called sto3Gbasis, where each element is an “atomic orbital” in the 
form of a dictionary. The dictionary will contain the orbital’s atom-centered coordinates, the atomic 
umber of the atom it’s centered on, the orbital angular momentum ($p_y$ implies $l = 0, m = 1, n = 0$), 
and the orbital’s basis set values for $\alpha_p$ and $d_p$. Because each dictionary contains all of the orbital 
information we need for calculating molecular integrals, each orbital is effectively represented by 
this dictionary. Each atom is assigned orbitals by its atomic orbital “basis configuration,” which 
we discussed in Handout 1 and will now briefly re-define.

Any atom in the first row of the periodic table has the (minimal) atomic orbital basis config- 
uration of, simply, $1s$ – just one orbital is available for the 1 or 2 electrons of hydrogen (electron 
configuration $1s^1$) or helium (electron configuration $1s^2$) to occupy. Any atom in the second row
of the periodic table is endowed with orbitals according to the the minimal basis configuration \(1s\ 2s\ 2p_x\ 2p_y\ 2p_z\), whether it be lithium (which has the ground state electron configuration \(1s^22s^1\)), or carbon (which has the ground state electron configuration \(1s^22s^22p^2\)), or neon (\(1s^22s^22p^6\)).

Basis configurations can be associated with particular atoms using another dictionary we call \texttt{orbitalconfiguration}.

```python
# minimal orbital basis configurations through the second row
orbitalconfiguration = {
    'H': ['1s'],
    'He': ['1s'],
    'Li': ['1s', '2s', '2p_x', '2p_y', '2p_z'],
    'Be': ['1s', '2s', '2p_x', '2p_y', '2p_z'],
    'B': ['1s', '2s', '2p_x', '2p_y', '2p_z'],
    'C': ['1s', '2s', '2p_x', '2p_y', '2p_z'],
    'N': ['1s', '2s', '2p_x', '2p_y', '2p_z'],
    'O': ['1s', '2s', '2p_x', '2p_y', '2p_z'],
    'F': ['1s', '2s', '2p_x', '2p_y', '2p_z'],
    'Ne': ['1s', '2s', '2p_x', '2p_y', '2p_z'],
}
```

By using the dictionary \texttt{orbitalconfiguration}, we can append orbitals to an array representing our basis (\texttt{sto3Gbasis}) as each atom of the array \texttt{atoms} is accessed during the loop. As described earlier, the orbitals themselves are constructed as dictionaries containing all of the relevant orbital information, such as orbital angular momentum and basis set values.

```python
# the basis will be constructed as an array;
# it is initially empty, but built up as orbitals (dictionaries) are appended to it in a later
# for loop
sto3Gbasis = []

# looping through the atoms array, append orbitals (dictionaries) to the sto3gbasis array,
# where each element of the atoms array is termed atom; # each atom has an orbital configuration associated with it, so loop through these too
for atom in atoms:
    # for orbital in orbitalconfiguration(atom):
        if orbital == '1s':
            sto3Gbasis.append( {1s orbital dictionary} )  # add the atom's 1s orbital to basis
        elif orbital == '2s':
            sto3Gbasis.append( {2s orbital dictionary} )  # add the atom's 2s orbital to basis
        elif orbital == '2px':
            sto3Gbasis.append( {2px orbital dictionary} )  # add the atom's 2px orbital to basis
        elif orbital == '2py':
            sto3Gbasis.append( {2py orbital dictionary} )  # add the atom's 2py orbital to basis
        elif orbital == '2pz':
            sto3Gbasis.append( {2pz orbital dictionary} )  # add the atom's 2pz orbital to basis
```

The nature of orbitals as dictionaries will be fleshed out in the Guided Exercises. For now, the thing to pay attention to are the \texttt{if statements}. The code inside the \texttt{if} block will only be executed if the condition of \texttt{orbital == '1s'} is met. (Note that there are two equals signs, not just one. With two equals signs we’re not setting \texttt{orbital} equal to the string value \texttt{'1s'}, rather we are seeing if it is equal to the string value \texttt{1s}.) Sometimes one if statement is adequate, as we saw with the Hartree–Fock convergence condition. But in this case it is not, because we are interested in more
than the condition that the orbital is 1s. We are also interested in the possible condition that the orbital is 2s, 2p_x, and so on. For this, we follow if statements with elif, or “else-if”, statements, as shown.

**Guided Exercises**

In the following exercises you will be carefully guided through the beginning of the molecular integrals programming project, in which you will develop large parts of three of the five interacting .py files that will form the foundation of your quantum chemistry code. (It is certainly possible to have all of your code in one file, but management is considerably easier if it is broken up.) The five files we will prepare are:

1. **basis.py**: Basis set information and construction of the STO-3G basis (with the function build_sto3Gbasis) will be located here. You will develop and complete this code during these Guided Exercises.

2. **oei.py**: This will contain the code for calculating matrix elements of the one-electron integral matrices S, T, and V. You will be guided through coding the calculation of matrix elements of S during the guided exercises.

3. **eri.py**: The code for calculating matrix elements of the two-electron integrals that make up G will be in this file. This code will not be discussed here.

4. **hf.py**: This will contain the code for calculating the electronic energy, which depends on S, T, V, and G. Other than the Hartree–Fock example provided earlier in this document, we will not be discussing hf.py in these Guided Exercises.

5. **main.py**: This central code file will integrate, or “drive”, the four other code files, beginning with reading in an input .xyz file. Then the basis will be built by accessing functions within basis.py. This is followed by constructing empty NumPy matrices, and calling on relevant functions in oei.py and eri.py to calculate the matrix elements. Finally, the matrices are passed into hf.py, where the total electronic energy of the molecule is calculated by the Hartree–Fock method. We will construct part of main.py in this set of exercises so we can
build \textbf{S} and print the overlap integral matrix. (\texttt{h2o.xyz} as presented in Fig. 4.17 in this document will be our test case.)

Completing these exercises will provide practice in all of the coding techniques you will employ for the remainder of the project (building \textbf{T}, \textbf{V}, \textbf{G}, and computing the Hartee–Fock energy).

Throughout the following exercises, be sure to continually describe your code with detailed comments. Here are a couple \textit{debugging} tips to keep in mind as you get started:

1. Errors you encounter can often be fixed by addressing addressing a flagged line of bad code that Python reports when you try to run your code.

2. Be sure to regularly \texttt{print} any variables (numbers, matrices, \texttt{etc.}) that are formed during the development of your code, as this can help you infer where the offending code exists.

---

1. In a directory that will hold all of your quantum chemistry code, create a file called \texttt{basis.py}.

2. In \texttt{basis.py}, we will not need a preamble to import any libraries. Type in the code to create a dictionary called \texttt{Z} that holds the atomic symbols (keys) and atomic numbers (values) of the first ten elements of the periodic table, as demonstrated earlier.

3. After an empty line following the code of the previous step, type in the necessary code to create a nested array called \texttt{d}, which will hold the STO-3G \textit{dp} contraction coefficients for 1s, 2s, and 2p orbitals. This also was demonstrated earlier in this Handout. (Don’t forget comments!)

4. Create the dictionary \texttt{orbitalconfiguration}, as described earlier.

5. Create a nested array featuring the \textit{α} factors. This will involve not just one nesting within the outer array, as with \texttt{d}, but two nestings, because \textit{each atom} has its own, unique set of \textit{α_p} factors. (The set of contraction coefficients in a STO-NG basis apply to all atoms). Like the contraction coefficients, values for \textit{2p} apply to \textit{2p_x}, \textit{2p_y}, and \textit{2p_z}. Note also that \textit{2s} and \textit{2p} orbitals have the same \textit{α_p} factors. Here’s part of this code:
Fill in the values corresponding to the elements between lithium and neon, as directed in dark blue text. Go to https://bse.pnl.gov/bse/portal, select the elements you need, and then select STO-3G from the scrolling menu on the left side of the page. From the “Format:” drop down menu below the periodic table, select Turbomole, because this is the format we presented in the main article. Type in the $\alpha$ values you see on the website in to your alpha array; the $\alpha$ values are in the left column, as was demonstrated in Fig. 7 of the main article, and duplicated here for convenience.

Figure 4.20: An example of the structure of the tabulated STO–3G basis set for the carbon atom. Many table formats are found; this is “Turbomole” selected from the drop-down menu (with “Optimized General Contractions”) of the EMSL Basis Set Exchange, with “carbon” highlighted in the website’s periodic table. The red text was added in this figure to highlight salient features of the table. $n$ is the principal quantum number.
The contraction coefficients are in the right column, and you should confirm this by noting that they are the same as those you already typed in for the d array.

As you enter in the α values, take care to heed the order of the basis set numbers! Compare the numbers shown in the starting examples with those printed on the website. The numbers going left to right in alpha are the same as those going bottom to top on the EMSL website. Double (and triple!) check that you enter every digit correctly, and that you have every closing bracket for every opening bracket. If there is a typo, it could lead to numerical errors later on that would be very difficult to trace back to a simple transcription error from this admittedly tedious task.

This would be a good time to try running your program. At this point it won’t return anything, but if running it produces an error, then there may something as trivial as a missing closing bracket in your code. The printed error will often direct you to the line that can be found. It is good practice to frequently check that your code is running, even well before it is complete. The more code you write in, the more likely are errors to be produced, and the harder it becomes to track and fix them.

6. We now define a function build_sto3Gbasis that will house the code we introduced in our discussion of conditional logic. This will append to an array sto3Gbasis every atomic orbital in the molecule, based on the atomic orbital basis configuration. Type in the following code, and complete the unfinished code indicated in dark blue.
sto3Gbasis = [] # instantiate the array representing our basis

def build_sto3Gbasis(atoms, R):
    
    This function depends on the atoms array, which lists strings of atomic symbols of the
    atoms of the input
    molecule, in the same order as the input .xyz file.
    This function additionally depends on the molecule's coordinates R (a nested array),
    where each element is
    an array holding the coordinates of each atom, also in the same order as atoms.
    
    K = 0 # instantiates atomic orbital counter, which will be used for matrix dimensions
    # loop through the atoms array, and append orbitals (dictionaries) to the basis array
    # each atom (in atoms) has an orbital configuration associated with it, so loop
    through these too
    for i, atom in enumerate(atoms):
        for orbital in orbitalconfiguration[atom]:
            if orbital == '1s':
                sto3Gbasis.append(
                    {
                    'Z': Z[atom], # atom name --> atomic number
                    'o': orbital, # append the orbital-type string ('1s','2s',
                    etc.)
                    'R': R[i], # get array [x,y,z] of i-th atom coordinates
                    'l': 0, # s orbital --> 0 angular momentum
                    'm': 0,
                    'n': 0,
                    'a': a[(Z[atom]-1)][0], # append list of 1s orbital
                    exponential factors
                    'd': d[0] # append list of 1s orbital contraction
                    coefficients
                    }
                )
                K = K + 1 # increment orbital counter
            elif orbital == '2s':
                # append 2s orbital dictionary to sto3Gbasis
                # increment orbital counter K by one
                sto3Gbasis.append(
                    {
                    'Z': Z[atom],
                    'o': '2s',
                    'R': R[i],
                    'l': 1, # 2p orbital has angular momentum in x direction
                    'm': 0,
                    'n': 0,
                    'a': a[(Z[atom]-1)][1], # 2p orbital exponent = 2s
                    orbital exponent
                    'd': d[2] # append list of 2p orbital contraction
                    coefficients
                    }
                )
                # append 2py orbital under the condition that orbital == '2py'; increment K
                # append 2pz orbital under the condition that orbital == '2pz'; increment K
                # when this function is called, return both the basis and the orbital count
    return sto3Gbasis, K


While typing in this code, you will have noticed that the code for the function build_sto3Gbasis has changed just slightly from our earlier presentation of function build_sto3Gbasis. Namely, we have adopted the keyword enumerate in the for loop. We do so because this is a convenient way of conjoining the two kinds of loops we discussed earlier: One in which the loop index is simply a counting integer variable (i = 0, 1, 2, ..., upper bound of range), and one in which the index actually equals the element of whatever array is being looped through (for atom
in atoms: ... atom = 'O', atom = 'H', ...).

This is useful because $\mathbf{R}$ is a nested array, and the $i$-th atom’s coordinates are those corresponding to whatever $\text{atom}$ is being looped through. For example, if $\text{atoms} = ['H', 'F']$, which has the corresponding molecular coordinates $\mathbf{R} = [[0.00, 0.00, 0.00], [0.00, 0.00, 1.79]]$, then when in the second loop that $\text{atom} = 'F$', then $i = 1$, and $\mathbf{R}[i] = \mathbf{R}[1] = [0.00, 0.00, 1.79]$.

Pay careful attention to the indentations of for loops and if statements, because it is easy to throw an error if any of them are not aligned correctly. Continue to ensure you are closing every bracket or parentheses that you open. And include comments!

This completes $\text{basis.py}$. Confirm there are no obvious errors in your code by running it.

7. Create a file in your programming project directory called $\text{oei.py}$. In this file we will write code to calculate the overlap integral ($\mathbf{S}$) matrix elements.

8. Import the NumPy library, and three modules of the SciPy library, by typing in the following code in the preamble (beginning lines) of $\text{oei.py}$.

   ```python
   import numpy as np # to handle matrices, calculate pi, exponential functions, and more
   from scipy import misc, special, linalg # to handle other specific math operations
   ```

9. Define a function for computing normalization constants, and call it $\text{compute norm}$. This function will be employed during the calculation of matrix elements of $\mathbf{S}$, $\mathbf{T}$, $\mathbf{V}$ and $\mathbf{G}$. The formula was derived in Handout 1, and is found there as Eq. 4.56. Type the following code in, and take care to notice the relationship between Handout 1 Eq. 4.56 and the code.
def compute_norm(a, l, m, n):
    """
    Compute the normalization constant, using Eq. 4.50 from Handout 1.
    Applies to S, V, T, and G.
    """
    N = (2*a/np.pi)**(3/2) * (?)**(?)
        / (misc.factorial2(2*l-1) * misc.factorial2(?) * misc.factorial2(?))
    N = N**(1/2)
    return N

There are several things to notice.

N is initially equated to just the bracketed part of the normalization constant \( N \) of Eq. 4.56. Then, in the next line of code, the square root of that result is taken. As before, the line \( N = N**(1/2) \) is not algebraically correct, but in code it serves to update the variable \( N \), to reflect the full form of the normalization constant \( N \) as it appears in Handout 1 Eq. 4.56.

Another thing to notice is that because the equation for \( N \) is quite long, we break it up over several lines of code by adding a backslash at the end of the first line of the coded equation.

Additionally, double factorials are computed by including in the parentheses of \( \text{misc.factorial2()} \) whatever we want the double factorial of – that is, the argument of the function goes in the parentheses. Recall that we imported the \text{misc} module of the SciPy library in our preamble.

Complete the \texttt{compute_norm} function by substituting the dark blue placeholder text (question marks) with functioning code.

10. We will frequently need to compute third centers between some extant centers – e.g., the third center \( \vec{P} \) between \( \vec{A} \) and \( \vec{B} \). Let’s create a function based on Eq. 4.145 of Handout 1, and let \( \alpha_a = a, \alpha_b = b, \vec{A} = \vec{RA}, \text{and } \vec{B} = \vec{RB} \)

   def compute_third_center(a, RA, b, RB):
       """
       Compute a Gaussian product third center (P between A and B, or Q between C and D), according to Eq. 4.34 of Handout 1.
       """
       # code to calculate P in terms of the function's arguments
       return P

where of course \( P = \vec{P} \). Implement Handout 1 Eq. 4.145 in the blue block of code.

11. Type in the function \texttt{compute_ABsq} for computing the distance between two orbitals (or atoms), as described earlier.
12. Now we write a function for 4.144 of Handout 1. This is among the most complex functions, but once you have a good grasp on it, by understanding its application to the matrix $\mathbf{S}$, you will be able to apply it to the other matrices ($\mathbf{T}$, $\mathbf{V}$, $\mathbf{G}$). Type in the following code.

```python
def build_S(basis, S):
    
    # create an empty $K \times K$ NumPy matrix called $S$
    for A, bA in enumerate(basis): # access row $A$ of $S$ matrix; call basis info for orbital $A$
        for B, bB in enumerate(basis): # access column $B$ of $S$ matrix; call basis info for orbital $B$
            # code to manipulate matrix element $S[A,B]$
    return S
```

You know how to create empty NumPy matrices of any dimensions, so type in the code for instantiating $S$ where you are directed to. We will fill in # code ... shortly, but first let’s consider the nested for loop.

This nested loop is very similar to the example we provided earlier for computing the overlap integral matrix of dihydrogen with a STO-1G basis set. However, because we are now using a STO-3G basis, and want to generalize the code to atoms other than hydrogen, and to molecules of any size, we will loop through the array `basis` instead of the array `atoms`. This gives us access to the detailed basis set information for each orbital (which is represented by three Gaussian primitives now, not just one), which could be a $1s$ orbital of H, a $2p_y$ orbital on F, and one of many other possibilities. Once again we adopt `enumerate` loops. $A$ is an integer counting variable, and $bA$ is equal to an element of the array `basis`. $bA$, then, will be a dictionary because, as we’ve already discussed, each element of the array `basis` is an “orbital dictionary.”

Initially, $A$ and $B$ are equal to 0, and $bA$ and $bB$ will both be the first dictionaries of the basis array – that is, both reference the same atom-centered orbital. Once code to manipulate matrix $S$ on this upper-left-most matrix element (overlap of $\phi_A$ with $\phi_A$) is complete, then, with $bA$ still at 0 (which is the first row of matrix $S$), $bB$ will be incremented by one, accessing the second element (or “column”) of $S$ in the first row. $bB$ will increment as many times
as there are elements of basis, before bA is incremented by one, and bB is again at 0 – the beginning of the second row of S.

This nested loop scheme gives us access to all elements of S.

However, we need to nest more loops within these two loops, precisely because each orbital is made up of three Gaussian primitives. Within each matrix element, then, a Gaussian overlap integral must be calculated three times, corresponding to each primitive of each orbital. The code is therefore expanded as follows:

```python
def build_S(basis, K):
    
     # create an empty K x K NumPy matrix called S
    # for A, bA in enumerate(basis): # access row A of S matrix; call basis info for orbital A
    # for B, bB in enumerate(basis): # access column B of S matrix; call basis info for orbital B
    # pull basis set information for orbital A, B
    for a, dA in zip(bA['a'], bA['d']): # a is alpha, dA is contraction coefficient
        # analogous for loop for basis set information of orbital B
        RA = bA['R'] # collect atom-centered coordinates of orbital A from basis
        RB = bB['R'] # collect atom-centered coordinates of orbital B from basis
        lA, mA, nA = bA['l'], bA['m'], bA['n'] # collect orbital A angular momentum
        RP = compute_third_center(a,b,RA,RB)
        S[A,B] = # Eq. 4.49
    return S
```

The new for loops serve to access individual elements of the basis set (\(\alpha_p\) factors and \(d_p\) contraction coefficients). To do so, we have employed a new functionality of Python loops called `zip`. (Don’t worry, this is the last variation on loops needing to be introduced!) Looping with `zip` allows one to loop through two arrays in tandem: As the first element of the value of key 'a' from dictionary bA is being accessed with a, so too is 'd' with dA. The first elements of each array are accessed simultaneously, the second elements simultaneously, and the third elements simultaneously. This is essential, because the first \(\alpha\) coefficient can only be paired with the first \(d_p\) contraction coefficient, the second \(\alpha_p\) with the second \(d\), and the third \(\alpha_p\) with the third \(d_p\). Any another combination is meaningless.

Recall that the keys 'a' and 'd', in each orbital dictionary of basis, have “values” that are
arrays. In STO-3G, there are three elements in both 'a' and 'd'. If we had chosen STO-6G, there would be 6 elements in each of these arrays.

New variables, such as RA, RB, lA, lB, and so on, are instantiated to pull coordinate and angular momentum information from bA and bB.

Having accessed a matrix element, and having pulled relevant information about the individual orbitals from the basis, Eq. 4.143 can be evaluated. Using the fact that

\[ S[A,B] = S[A,B] + \ldots \]

is equivalent to

\[ S[A,B] += \ldots \]

let’s flesh out this line of code.

```python
np.exp(-(a+b/(a+b)) * compute_ABsq(arguments)**2) * \ 
computeSi(lA,lB,RP[?]-RA[?],RP[?]-RB[?],a+b) * # Sx
computeSi(nA,nB,RP[?]-RA[?],RP[?]-RB[?],a+b) * # Sy
computeSi(mA,mB,RP[?]-RA[?],RP[?]-RB[?],a+b) # Sz
```

Complete the code. Note that

\[ \vec{PA} = \vec{P} - \vec{A} \]

such that

\[ \vec{PA}_x = \vec{P}_x - \vec{A}_x, \quad \vec{PA}_y = \vec{P}_y - \vec{A}_y, \quad \vec{PA}_z = \vec{P}_z - \vec{A}_z \]

We have called a function `compute_Si`, which we have not yet coded. But it makes sense to code a generic \( S_i \), because for each Gaussian primitive overlap integral we must compute something very similar three times (\( S_x, S_y, S_z \)).
13. Referring to Eq. 4.144 of Handout 1, enter and complete the following code:

```python
def compute_Si(lA,lB,PA,PB,\gamma):
    ""
    Calculate the i-th coordinate contribution to the matrix element S[A,B].
    ""
    for k in range( int((lA + lB)/2)+1 ): # note range is up to and INCLUDING floor of (lA +lB)/2, hence +1
        Si += compute_ck(2*k,lA,lB,PA,PB) * np.sqrt( np.pi / \gamma ) * \n            misc.factorial2(2*k-1) / (?)**k
```

The floor function, ⌊⌋ depicted in Eq. 4.144 is equivalent to the integer `int()` function of Python. $S_i$ depends on one final function, $c_k$, which we implement next.

14. We now define another function for computing the the constant $c_k$. Referring to Eq. 4.41 of Handout 1, and calling the `binomial` function of the `special` module (part of the SciPy library), type in the following code:

```python
def compute_ck(arguments):
    for i in range(l+1): #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.

We are now finished with `oei.py` for the Guided Exercises, having implemented all of the code for computing the overlap integral matrix.

We need to tie together `basis.py` and `oei.py` with `main.py`.

15. Create a file in your programming project directory called `main.py`.

16. In the preamble of `main.py`, import the NumPy library, as you did for `oei.py`. Also import the library `argparse`. Finally, import `oei` and import `basis`, which will allow us to access all functions of these code files, which we prepared in the previous exercises.
17. After an empty line, type in the following code:

```python
parser = argparse.ArgumentParser()
parser.add_argument('coords_file', metavar='coordinates.xyz', type=str)
args = parser.parse_args()
inputfile = open(args.coords_file,'r') # open .xyz file without being able to modify it (read-only)
```

This code will allow the user of main.py to specify an input file when running the program, like so:

```bash
$ python main.py [.xyz file]
```

The variable inputfile, then, will hold contents of the .xyz input file (e.g., h2o.xyz). (At some point the reader should explore features of argparse functionality, because with additional add_arguments, one can include greater input functionality. For example, you might wish to allow the user to control the molecule’s charge or multiplicity if open-shell functionality is added to the closed-shell Hartree–Fock code that is presented in Handout 4.)

18. Now we will begin generating code to automatically read in the .xyz files and generate the atoms and coordinates array R. Construct a function called read_input_file to read an inputfile line by line, at each line extracting atomic symbol and the atom’s coordinates.

```python
def read_input_file(inputfile):
    """Builds two arrays from .xyz input file:
    1) atomarray, containing elements of atomic symbols (strings) in order of input file
    2) coords, a nested array containing elements of coordinates (1 coordinates array per
    atom) also in order
    """
    atomarray = [] # this array will hold atomic symbols in order of input file
    coords = [] # in same order as atomarray, each element will be an array of i-th atom's
    coordinates
    # read through inputfile line by line
    for line in inputfile.readlines()[2:]: # [2:] means read only from the third line of
    the input file onwards
        atom, x, y, z = line.split() # from each atom line collect atomic symbol, x, y, z
        atomarray.append( atom ) # append to atomarray the contents of the string variable
        atom
        coords.append( [float(x),float(y),float(z)] ) # append to coords current atom's
        coordinates (as array)
    return atomarray, coords
```

19. Call the function read_input_file developed in the previous step, and pass into it inputfile, to return an array atoms and coordinates R.
20. Build the STO-3G basis by typing in and completing

```python
# build the STO-3G basis using these inputs
orbitalbasis, basissize = basis.build_sto3G_basis(?, ?) # basis size (number of AOs) ==> dimensions of matrix
```

If you are unsure what the arguments to the function call should be, return to the part of your code in `basis.py` where you defined the `build_sto3Gbasis` function, and use this information to determine what to pass into it. Also make sure you understand the relationship between `orbitalbasis` and `basissize`, and what is returned by the function `build_sto3Gbasis`.

21. Finally, pass in to the `build_S` function (which is in `oei.py`) the array `orbitalbasis` and the matrix dimension integer `basissize`, which we collected in the previous exercise.

```python
# the result of calling on build_S is the overlap integral matrix
S = oei.build_S(?, ?)
```

As the comment states, the result of calling `build_S`, in the form of `S` here, should be the complete overlap integral matrix `S`.

22. The exercises are complete! All that remains is to `print(S)`, and ensure that the result is identical to that shown in Fig. 4.21, for the input file also shown there.

A reminder: Errors you may encounter in your code are often fixed by addressing a flagged line of bad code that Python reports when you try to run your code. An additional debugging practice includes printing variables and results at several different stages throughout your code. For example, if everything prints as expected up until `Sx`, `Sy`, and/or `Sz` is being calculated, then there is very likely an error in your `compute_Si` function. Did you type in everything exactly as shown in this document? Are you sure you filled in the incomplete code correctly?

You have now practiced every coding skill you will require to complete the remainder of the coding projects in Handout 3 and Handout 4. Any other programming techniques you will need are described in those documents.
Other Resources

One’s programming skills can always be improved through practice and exposure to new techniques. Further and broader introduction to programming in Python can be found in many excellent books and online introductory guides. At the time of this manuscript’s writing, two excellent online introductory guides are CodeAcademy (www.codeacademy.com) and Learn Python the Hard Way (www.learnpythonthehardway.org).

The Internet is a spectacular resource for answering coding questions. If you’re having a problem, or need to know if there is a way to perform some task in Python, a good search engine query can often yield results that provide exactly the information you are looking for (e.g., “how do I diagonalize a matrix in python with numpy”). Additionally, take advantage of colleagues that have coding experience. Ask them how they would go about coding something (but don’t let them do the work for you!).

---

<table>
<thead>
<tr>
<th></th>
<th>O1s</th>
<th>O2s</th>
<th>O2px</th>
<th>O2py</th>
<th>O2pz</th>
<th>H1s</th>
<th>H2s</th>
</tr>
</thead>
<tbody>
<tr>
<td>O1s</td>
<td>1.00000</td>
<td>0.23670</td>
<td>0.00000</td>
<td>0.00000</td>
<td>0.00000</td>
<td>0.00678</td>
<td>0.00678</td>
</tr>
<tr>
<td>O2s</td>
<td>0.23670</td>
<td>1.00000</td>
<td>0.00000</td>
<td>0.00000</td>
<td>0.00000</td>
<td>0.09883</td>
<td>0.09883</td>
</tr>
<tr>
<td>O2px</td>
<td>0.00000</td>
<td>0.00000</td>
<td>1.00000</td>
<td>0.00000</td>
<td>1.00000</td>
<td>0.00000</td>
<td>0.00000</td>
</tr>
<tr>
<td>O2py</td>
<td>0.00000</td>
<td>0.00000</td>
<td>0.00000</td>
<td>1.00000</td>
<td>0.00000</td>
<td>0.08159</td>
<td>-0.08159</td>
</tr>
<tr>
<td>O2pz</td>
<td>0.00000</td>
<td>0.00000</td>
<td>0.00000</td>
<td>0.00000</td>
<td>1.00000</td>
<td>-0.06843</td>
<td>-0.06843</td>
</tr>
<tr>
<td>H1s</td>
<td>0.00678</td>
<td>0.09883</td>
<td>0.00000</td>
<td>0.08159</td>
<td>-0.06843</td>
<td>1.00000</td>
<td>0.02420</td>
</tr>
<tr>
<td>H2s</td>
<td>0.00678</td>
<td>0.09883</td>
<td>0.00000</td>
<td>-0.08159</td>
<td>-0.06843</td>
<td>0.02420</td>
<td>1.00000</td>
</tr>
</tbody>
</table>

Figure 4.21: The outcome of evaluating every element of the overlap integral matrix \( \mathbf{S} \) in the minimal STO-3G basis, with the .xyz coordinates (units Bohr) shown below it.
4.5 Handout 3: Implementation of the Molecular Integrals

Prepare your integrals code

It is at the reader or instructor’s discretion whether to include all molecular integrals and Hartree–Fock code in one file, or to break the code up over several files. In Handout 2 (introduction to programming), the latter was recommended because it makes managing the code easier. The suggested file structure was as follows: basis.py (to hold STO-3G basis set information and construct the atomic orbital basis), oei.py (to calculate one-electron integrals $S$, $T$, $V$), eri.py (to calculate two-electron integrals $G$), and main.py (to “drive” the first three files and pass results into the Hartree–Fock code hf.py described in the next Handout). It was discussed in Handout 2 how, from within one .py file, one call call a function from another .py file.

In most of these code files you will need to import a few modules from the SciPy library to perform some specialized mathematical operations (e.g., double factorial). Additionally, you may need to import the NumPy library, because both simple (e.g., matrix construction, matrix multiplication) and sophisticated (e.g., diagonalization) linear algebra operations are easily handled with NumPy, and will prove especially useful when coding the Hartree–Fock method. In the current project, we will use NumPy to build empty two-dimensional matrices (each of which will hold the overlap, kinetic energy, and electron-nuclear attraction integrals), as well as a rank-4 tensor (essential just a four-dimensional matrix) to hold the electron-electron repulsion integrals; the elements will then be filled in using the appropriate matrix element equations.

Read the input .xyz file

In the main.py file ...

1. Read in the .xyz file. (The test case for this project is h2o.xyz, shown in Fig. 3 of the main article.)
   
   - Form a one-dimensional list itemizing atoms (e.g., ['O', 'H', 'H']).
   - Form a two-dimensional nested list of the atom’s $x, y, z$ coordinates ([[0.000, 0.000, 0.000], ...]).

   If the coordinates aren’t already in units Bohr, make sure to convert them.
Build the STO-3G basis

In basis.py ...

1. Construct two arrays that hold the basis set information (we will use the minimal basis set STO-3G). One array should be allocated for \( \alpha_p \) exponential factors, and the other for \( d_p \) contraction coefficients (here, \( p \) is a general index, which is \( a \) for \( A \) orbitals, \( b \) for \( B \) orbitals, and so on). The \( \alpha_p \) containing array will need to be nested two times in order to hold values for distinct atoms, distinct subshells within atoms, and the three individual values. The \( d_a \) array will only need to be nested once in order to hold values for different shells, because the same contraction coefficients apply to all atoms. It is suggested that you include basis set information for at least the first ten atoms of the periodic table, which would mean returning the length of \( \alpha_p \) yields the value 10.

The required STO-3G basis set values for \( \alpha_p \) and \( d_p \) can be found at https://bse.pnl.gov/bse/portal. (See Fig. 7 of the main article for downloading and interpreting basis set values.)

2. Create a dictionary `orbitalconfiguration` that returns the minimal basis atomic orbital configuration of any atom through the second row of the periodic table (the first ten atoms).

   - For example, passing in '0' or 'C' returns ['1s', '2s', '2px', '2py', '2pz']

3. Looping through the atoms, and then through the orbitals of each atom ...

   - Append to a `sto3Gbasis` array, such that
     - each element is a dictionary, and
     - the keys of each dictionary host ...
       (a) List of \( \alpha_p \) exponential factors (depends on orbital type)
       (b) List of \( d_p \) contraction coefficients (depends on orbital type)
       (c) List of atom coordinates \((x, y, z)\)
       (d) \( l \) (= 0 if 1s, 2s, 2py, 2pz; = 1 if 2px)
       (e) \( m \) (= 0 if 1s, 2s, 2px, 2pz; = 1 if 2py)
       (f) \( n \) (= 0 if 1s, 2s, 2px, 2py; = 1 if 2pz)
• By way of example, calling information on just the $2p_x$ orbital of the oxygen atom should return something close to

$\{ 'a': [0.380389, 1.1695961, 5.0331513], 'd': [0.39195739, 0.60768372, 0.15591627], 'R': [0.0, 0.0, 0.454025755], 'l': 1, 'm': 0, 'n': 0}$$

• Tip: When looping through the atoms, you will need not only the atom identity (e.g., hydrogen), but also its index (e.g., row 2 of the $x,y,z$ coordinates in the input file). Consider using

```python
for i, atom in enumerate(atoms):
```

to return both, where $i$ is the numerical index.

4. The basis is now built. When you call `sto3Gbasis`, built for the input `h2o.xyz`, seven items should be returned. Why is this?

**Overlap integrals**

In `oei.py` ...

1. Build an empty $K \times K$ matrix called $S$, where $K$ is the number of atomic orbitals included in the basis. Use `np.zeros` to generate an empty 2D NumPy matrix:

```python
S = np.zeros((K, K))
```

This will be your overlap integral matrix $S$, or $S_{A,B} \in \mathbb{R}^{K \times K}$. The equation for any individual matrix element $(A|B)$ of $S$ is

\[
(A|B) = \int \int \int \phi_A \phi_B \, dx \, dy \, dz = \sum_{a=1}^{K_a} \sum_{b=1}^{K_b} d_a d_b N_a N_b \int \int \int \chi_a(\vec{A}, \alpha_a, l_A, m_A, n_A) \chi_b(\vec{B}, \alpha_b, l_B, m_B, n_B) \, dx \, dy \, dz
\]

(4.142)

where $\chi_a$ is the Gaussian primitive for orbital $A$. The linear combination of these primitives form our atomic orbital $\phi_A$, therefore we must sum up to $K_a$, which is the number of primitives per orbital (3 for `sto3Gbasis`, because we take a linear combination of three Gaussian
functions to emulate one Slater-type orbital). Recall that $\alpha_a$ is the exponential factor for the current primitive (of 3 total) of orbital $A$, $d_a$ is the contraction coefficient, $\vec{A}$ are the atomic $x,y,z$ coordinates of orbital $A$ (because the orbitals are centered on the atoms), and $l_A, m_A,$ and $n_A$ are the angular momenta corresponding to the $x, y,$ and $z$ directions, respectively.

This should provide a fairly concrete sense of how a matrix element is evaluated, however we still require analytical expressions for the multiple integral component in order to code it up. We will come to that shortly.

2. Loop over each element $(A|B)$ of $S$ in order to access and modify that element.

   - Do so with two nested loops, with each loop corresponding to orbital index $A$ or $B$.
   - While looping over an orbital index, you’ll need to pull its corresponding orbital information from the `sto3Gbasis` array. *(Tip: Use the `enumerate` functionality.)*

3. Now that we’ve “accessed” a matrix element with our two nested loops, we need to access elements within the basis that we’ve pulled for that index. For example, for the STO-3G basis set, there are three $\alpha_a$ exponent factors and three $d_a$ contraction coefficients that we will be passing on to another function.

   - Once again, we need to obtain the basis components for each index we’re looping over. In other words, we need yet two more nested loops within the nested loops we’ve already formed.
   - *Tip:* We can simultaneously retrieve the pair of $\alpha_a$ and $d_a$ values necessary for the current primitive by using another useful Python function called `zip`. For example, when looping within index $A$, the basis for which we might name $bA$, we can call the $\alpha_a$ and $d_a$ elements as follows:

   ```python
   for a, da in zip(bA['a'], bA['d']):
   ```

4. Now that we’ve accessed specific $\alpha_p$ and $d_p$ values to calculate the integral corresponding to one Gaussian primitive, we will need these values, the position coordinates $(x,y,z)$ of each atom that the orbitals we’re evaluating correspond to, and each orbital’s angular momentum
values \((l, m, n)\). We then plug these into our analytical expression for the overlap integral:

\[
\int \int \int \chi_a(\vec{A}, \alpha_a, l_A, m_A, n_A) \chi_b(\vec{B}, \alpha_b, l_B, m_B, n_B) \, dx \, dy \, dz = \exp \left( \frac{-\alpha_a \alpha_b}{\gamma} |\vec{A}\vec{B}|^2 \right) \times S_x \times S_y \times S_z
\]

(4.143)

where

\[
S_x = \sqrt{\frac{\pi}{\gamma}} \sum_{k=0}^{[\frac{(l_A+l_B)/2]}} c_{2k}(l_A, l_B, \vec{P}^A_x, \vec{P}^B_x) \frac{(2k-1)!!}{(2\gamma)^k},
\]

(4.144)

where \([x]\) is the floor function (conveniently, the \texttt{int(x)} function of Python returns the largest integer value less than or equal to the argument, which is the definition of the floor function), \(l_A\) and \(l_B\) are the angular momenta of orbital indices \(A\) and \(B\), respectively, and where

\[
\vec{P} = \frac{\alpha_a \vec{A} + \alpha_b \vec{B}}{\alpha_a + \alpha_b},
\]

(4.145)

\[
\vec{P}^A = \vec{P} - \vec{A},
\]

(4.146)

\[
|\vec{A}\vec{B}|^2 = (\vec{A}_x - \vec{B}_x)^2 + (\vec{A}_y - \vec{B}_y)^2 + (\vec{A}_z - \vec{B}_z)^2,
\]

(4.147)

\[
\gamma = \alpha_a + \alpha_b,
\]

(4.148)

\[
c_k(l, m, a, b) = \sum_{i=0}^{l} \sum_{j=0}^{m} \binom{l}{i} \binom{m}{j} a^{l-i} b^{m-j}; \quad i + j + 1 = k,
\]

(4.149)

Note that notation such as \(\vec{P}_x\) refers specifically to the \(x\) component of the position vector \(\vec{P}\).

The function \(c_k\) is a generic expression that will be used by more than just equation 4.144 – hence, the generic form of its arguments. The exclamation point above the equals sign indicates that the function is evaluated only when the sum of indices \(i\) and \(j\) is equal to the argument function \(k\).

Additionally, SciPy includes a function to calculate a double factorial:
The double factorial function can be applied to non-integer arguments, and cost-saving approximations made with \texttt{exact=False}. But the function is fastest if it is applied to exact integers, which is all that we are dealing with – hence, \texttt{exact} is set to \texttt{True}.

Binomial coefficients can be computed as:

\[ \texttt{special.binom(m,j)} \]

5. This Gaussian primitive is almost fully evaluated. However, we need to multiply it by its contraction coefficients \(d_a\) and \(d_b\), as well as its normalization constants \(N_a\) and \(N_b\), the latter of which are evaluated as follows:

\[
N(\alpha, l, m, n) = \left[ \frac{\left(\frac{2\alpha}{\pi}\right)^{3/2}}{(2l-1)!!(2m-1)!!(2n-1)!!} \right]^{1/2}
\]

(4.150)

6. Having looped through every orbital for both indices, and in turn every primitive for every orbital, the evaluation of the overlap integrals is complete!

\section*{Kinetic energy integrals}

In \texttt{oei.py} ...

1. The evaluation of the kinetic energy integrals proceeds in much the same manner as it does for the overlap integrals. There are simply more terms that go into each matrix element, which has the general form \((A| - \frac{1}{2} \nabla^2 | B)\); the operator features the Laplacian \(\nabla^2\), which is the sum of second partial derivatives with respect to \(x\), \(y\), and \(z\), individually. Form another \(K \times K\) matrix called \(T\).

2. As before, loop through every orbital pair to access a specific matrix element, and implement the following working equation:
\[
(A|\frac{1}{2}\nabla^{2}|B) = \sum_{\alpha}^{\alpha_0} \sum_{\beta}^{\beta_0} d_\alpha d_\beta N_\alpha N_\beta \int \int \chi_\alpha(\vec{A}, \alpha_\sigma, \Lambda; \Lambda) \left\{ a_\alpha [2(l_B + m_B + n_B) - 3] \chi_\beta(\vec{B}, \alpha_\sigma, \Lambda; \Lambda) - 2a_\alpha^2 \left[ \chi_\beta(\vec{B}, \alpha_\sigma, l_B + 2, m_B, n_B) + \chi_\beta(\vec{B}, \alpha_\sigma, l_B + 2, m_B, n_B) + \chi_\beta(\vec{B}, \alpha_\sigma, l_B, m_B, n_B + 2) \right] - \frac{1}{2} [l_B(l_B - 1) \chi_\beta(\vec{B}, \alpha_\sigma, l_B - 2, m_B, n_B) + m_B(m_B - 1) \chi_\beta(\vec{B}, \alpha_\sigma, l_B, m_B - 2, n_B) + n_B(n_B - 1) \chi_\beta(\vec{B}, \alpha_\sigma, l_B, m_B, n_B - 2)] \right\} dx dy dz
\]

(4.151)

While the presence of the integral signs might lead you to believe we haven’t presented a code-able expression, you should be able to see that this equation consists of the sum of seven distinct overlap integrals, with some leading factors consisting of constant terms \(\alpha_p\), angular momentum components \(l, m, n\), etc.. As such, this matrix element depends heavily on Eq. 4.143. The kinetic energy integrals are, mathematically, nothing more than a sum of overlap integrals with coefficients!

*Electron-nuclear attraction integrals*

In oe.py ...

1. The evaluation of the electron-nuclear attraction integrals is distinct from the construction of overlap and integral matrices in that we will now sum \(N\) two-dimensional matrices, each of which we call \(V_C\) corresponding to the index \(C\) for each atom, to form our matrix \(V\), as follows:

\[
V = V_1 + V_2 + \ldots + V_N \tag{4.152}
\]

where \(N\) is the number of atoms.

We will need to collect atomic number and coordinates for this new, third index \(C\). However, we do not need to access, for matrix \(V_C\), the \(\alpha\) exponential factors and contraction coefficients of \(C\).

2. The matrix element for any individual matrix element of \(V_C\) has the general form

\[
(A|\frac{-Z_C}{\vec{r}_C}|B) = \int \int \int \phi_A \left( \frac{-Z_C}{\vec{r}_C} \right) \phi_B \, dx \, dy \, dz \tag{4.153}
\]
where the operator features the atomic number $Z$ of the nucleus $C$; the denominator indicates the distance between electron $i$ and nucleus $C$.

The working equation for the matrix element is:

$$
(A | \frac{-Z_C}{\vec{r}_i} | B) = \sum_{a=1}^{K_a} \sum_{b=1}^{K_b} \sum_{l_A+l_B \geq \lfloor l/2 \rfloor} d_a d_b N_a N_b \times \left\{ -Z_C \times \frac{2\pi}{\gamma} \times \exp \left( \frac{-\alpha_a \alpha_b}{\gamma} |\vec{A}\vec{B}|^2 \right) \right\} 
\times \sum_{l=0}^{l_A+l_B \geq \lfloor l/2 \rfloor} \sum_{r=0}^{l-2r} \sum_{i=0}^{\lfloor (l-2r)/2 \rfloor} v_{l,r,i}(l_A, l_B, \vec{A}_x, \vec{B}_x, \vec{C}_x, \gamma) 
\times \sum_{m=0}^{m_A+m_B \geq \lfloor m/2 \rfloor} \sum_{s=0}^{m-2s} \sum_{j=0}^{\lfloor (m-2s)/2 \rfloor} v_{m,s,j}(m_A, m_B, \vec{A}_y, \vec{B}_y, \vec{C}_y, \gamma) 
\times \sum_{n=0}^{n_A+n_B \geq \lfloor n/2 \rfloor} \sum_{t=0}^{n-2t} \sum_{k=0}^{\lfloor (n-2t)/2 \rfloor} v_{n,t,k}(n_A, n_B, \vec{A}_z, \vec{B}_z, \vec{C}_z, \gamma) 
\times F_{l+m+n-2(r+s+t)-(i+j+k)}(\gamma |\vec{P}\vec{C}|^2) \right\} 
\left(4.154\right)
$$

where

$$
v_{l,r,i}(l_A, l_B, \vec{A}_x, \vec{B}_x, \vec{C}_x, \gamma) = (-1)^n \epsilon^i c_l(l_A, l_B, \vec{P}_A \vec{x} \vec{P}_B \vec{x}) \frac{(-1)^i l! \ell! \vec{P}^i_{\vec{C}_x} \ell^2 r^2 e^{\epsilon i}}{r! l! (l-2r-2i)!} \left(4.155\right)
$$

$$
\epsilon = \frac{1}{4\gamma} \left(4.156\right)
$$

Here, $\vec{P} \vec{C}_x$ is the $x$ component of a new vector $\vec{P} \vec{C} = \vec{P} - \vec{C}$, the distance between the midpoint position $\vec{P}$ and the third index position $\vec{C}$. It’s worth trying to see the pattern in the many indices present in the summations – for example, the floor functions constituting the upper range of certain sums involve indices being counted over from sums earlier in the same line of the equation. This interdependence of sums is largely due to the treatment of the product of polynomials that form the angular momentum terms of the contracted Gaussian type orbitals (see Handout 1). These summation indices, in turn, are arguments for the function $v$.

Single factorials can be calculated similarly to double factorials:
The Boys function for $x$ greater than zero

\[ F_{\nu}(x) = \int_{0}^{1} t^{2\nu} \exp\left(-xt^2\right) dt = \frac{1}{2x^{\nu+\frac{1}{2}}} \times \Gamma\left(\nu + \frac{1}{2}\right) \]  

(4.157)

includes both a lower incomplete gamma function $\gamma$ (not to be confused with our variable $\gamma = \alpha_a + \alpha_b$), and a normal Gamma function $\Gamma$, the latter of which was not discussed in the derivation, but is necessary because Python normalizes the lower incomplete gamma function $\gamma$ by dividing it by $\Gamma$. Evaluation of this in Python relies on two special SciPy functions:

\[
F = 0.5 \times x**(-(nu + 0.5)) \times \text{special.gammainc}(nu + 0.5, x) \times \text{special.gamma}(nu + 0.5)
\]

For very small $x$ (say, $x < 1 \times 10^{-6}$), the equation breaks down. This is because, as $x$ approaching zero in the denominator of the first quotient of the expression, the equation “blows up”. However, we can use a Taylor expansion of the function and truncate at second-order:

\[ F_{\nu}(x) \approx \frac{1}{2\nu + 1} - \frac{x}{2\nu + 3}\]  

(4.158)

where $x$ has been removed from the denominator. An if statement should be employed to allow both argument types.

3. Notice that there are three summation symbols per $x$, $y$, and $z$ direction of the function $v$. Thus, implementation of the equation for evaluating integration over one Gaussian primitive requires nine loops. Obviously, then, to fully evaluate a matrix element as a contraction of the three primitives (in STO-3G), these nine loops are within the two blocks of nested loops needed to access the matrix elements and the $\alpha_p$ and $d_p$ values.

Electron repulsion integrals

In eri.py ...

177
1. Having conquered the electron-nuclear attraction integrals, the two-electron repulsion integrals are effectively just an extension of these—i.e., more loops! There is one key difference: We must form a rank-4 tensor instead of just a two-dimensional matrix. So instead of looping over two indices to access a two-dimensional matrix element, we must loop over four in order to access any one element of this $K \times K \times K \times K$ “matrix”. Nevertheless, the looping technique is just the same as before.

Use NumPy to form $G$.

2. The two-electron operator is $(1/\vec{r}_{ij})$, and describes the inverse distance between electrons $i$ and $j$. This operator acts on the product of orbitals $C$ and $D$; after we’ve multiplied this result by the product of orbitals $A$ and $B$, we integrate over all space to get the matrix element $(AB|CD)$.

The working equation for the matrix element $(AB|CD)$ is

\[
G = \int \int \int \phi_A \phi_B \left( \frac{1}{\vec{r}_{ij}} \right) \phi_C \phi_D \, dx \, dy \, dz
\]

\[
= \sum_{a=1}^{K_a} \sum_{b=1}^{K_b} \sum_{c=1}^{K_c} \sum_{d=1}^{K_d} d_a d_b d_c d_d N_a N_b N_c N_d
\]

\[
\times \left\{ \frac{2\pi^2}{\gamma_P \gamma_Q} \left( \frac{\pi}{\gamma_P + \gamma_Q} \right)^{1/2} \times \exp \left( -\frac{\alpha_a \alpha_b |AB|^2}{\gamma_P} \right) \times \exp \left( -\frac{\alpha_c \alpha_d |CD|^2}{\gamma_Q} \right) \right\}
\]

\[
\times \sum_{l_P=0}^{l_A+l_B} \sum_{l_Q=0}^{l_C+l_D} \sum_{r_P=0}^{l_P-1/2} \sum_{r_Q=0}^{l_Q-1/2} \sum_{i=0}^{l_A+l_B} g_x
\]

\[
\times \sum_{m_P=0}^{m_A+m_B} \sum_{m_Q=0}^{m_C+m_D} \sum_{s_P=0}^{m_A+m_B} \sum_{s_Q=0}^{m_C+m_D} \sum_{j=0}^{l_A+l_B} g_y
\]

\[
\times F_\nu(|\vec{PQ}|^2/4\delta)
\]

where we have a new third center $\vec{Q}$ located between orbitals $C$ and $D$ (see Eq. 4.145); additionally,

\[
\gamma_P = \alpha_a + \alpha_b, \quad \gamma_Q = \alpha_c + \alpha_d
\]
\[ \delta = \frac{1}{4\gamma_P} + \frac{1}{4\gamma_Q} \]  

\[ \nu = l_P + l_Q + m_P + m_Q + n_P + n_Q - 2(r_P + r_Q + s_P + s_Q + t_P + t_Q) - (i + j + k) \]

(4.161)  

(4.162)

where

\[ g_x \equiv g_{l_P,l_Q,r_P,r_Q,i}(l_A,l_B,\vec{A}_x,\vec{B}_x,\gamma_P;l_C,l_D,\vec{C}_x,\vec{D}_x,\gamma_P) \]

\[ = (-1)^{l_P} \cdot \theta(l_P, l_A, l_B, P\vec{A}_x, P\vec{B}_x, r_P, \gamma_P) \cdot \theta(l_Q, l_C, l_D, Q\vec{C}_x, Q\vec{D}_x, r_Q, \gamma_Q) \]

\[ \times \frac{(-1)^i(2\delta)^2(r_P+r_Q)(l_P + l_Q - 2r_P - 2r_Q)!\delta^iP_{x}^{l_P+l_Q-2(r_P+r_Q+i)}}{(4\delta)^{l_P+l_Q}![(l_P + l_Q - 2(r_P + r_Q + i))!]}} \]

\[ \theta(l,l_A,l_B,a,b,\gamma) = c_l(l_A,l_B,a,b)\frac{l!\gamma^{r-l}}{r!(l-2r)!} \]

(4.163)

(4.164)

3. For anything but simple, light molecules such as H\(_2\), NumPy will produce some hard-to-read output when you print the massive electron-electron repulsion integral four-dimensional matrix. Here are a few things you can add to the beginning of your code to improve the appearance of printed matrices:

- `np.set_printoptions(threshold=np.inf)` # suppresses "...", printing all elements
- `np.set_printoptions(precision=5)` # reduces number of decimal places per element to 5
- `np.set_printoptions(linewidth=200)` # lengthens printable line-width from 75 characters
- `np.set_printoptions(suppress=True)` # suppresses hard-to-read exponential notation

**Storing integrals**

The process of calculating molecular integrals should be initiated from `main.py` by calling the appropriate functions for building the atomic orbital basis from `basis.py`. The basis can then be passed into `oei.py` to calculate \( S \), \( T \), and \( V \), and `eri.py` to calculate \( G \).

It can take several minutes to calculate these molecular integrals. The electron-electron repulsion integrals are particularly time-consuming. It is therefore worth saving the generated matrices to files that can later be opened, such as for when you’re coding and debugging the Hartree–Fock procedure. (You don’t want to have to re-compute the integrals for water every time you test your Hartree–Fock code.)
Let's say you had allocated a variable `fname` at the beginning of your code that holds the characters of the input file name preceding the `.xyz` extension – e.g., `h2o` from `h2o.xyz`. You can use `fname` to save the $S$ NumPy matrix for the overlap integrals of water:

```python
S_file = open('S.' + fname + '*.npy', 'w')
np.save(S_file, S)
```

The `.npy` is important for NumPy’s recognition of the file. In the future, then, you can easily load this saved NumPy matrix with

```python
S = np.load('S.h2o.npy')
```
4.6 Handout 4: Implementation of the Hartree–Fock Method

While clear and detailed expositions to molecular integrals are hard to find in the literature (hopefully until now) there are many, excellent resources that go into some detail about the Hartree–Fock–Roothan method. (Perhaps the most complete and excellent description of Hartree–Fock theory at a level of rigor consistent with the present paper is offered in the classic textbook by Szabo and Ostlund.)

Equipped with the values computed by the methods presented in Handout 3, one is ready to finish computing the solution to the Schrödinger equation. This is done by the self-consistent field (a.k.a., Hartree–Fock) method – it is a looping “recipe” whereby molecular orbitals are constructed from the atomic orbitals we have been working with this entire paper. The looping is necessary because the first guess at the nature of the molecular orbitals (which must be made to begin iteration) tends to be crude, but over a series of iterations these orbitals are tweaked, step-by-step, until there is little change in the orbitals between steps. The HF procedure is much easier to code than than the molecular integrals it depends on.

Implementing the Hartree–Fock method involves just a few additional NumPy techniques, namely computing the inverse of a matrix, taking dot products, and computing eigenvalues and eigenvectors of matrices. These will be introduced in individual steps of the algorithm we now present.

The Hartree–Fock algorithm is as follows:

1. Specify a molecule (as an .xyz file containing atom coordinates and atomic number) and a basis set (in the present study, STO-3G). This step was done when computing molecular integrals, as described in Handout 3.

2. Calculate all required molecular integrals: $S, T, V,$ and $G$. (Completed in Handout 3.)

3. Sum the kinetic and electron-nuclear attraction integral matrices to form the core Hamiltonian:

$$H_{\text{core}} = T + V.$$  \hspace{1cm} (4.165)
With NumPy, matrix addition is as simple as the following line:

\[
\text{Hcore} = T + V
\]

4. Generate the orthonormal transformation matrix \( \mathbf{X} \) from \( \mathbf{S} \) with the following equation:

\[
\mathbf{X} = \mathbf{S}^{-1/2}.
\]  

\[
\mathbf{X} = \text{linAlg.sqrtm(linAlg.inv(S))}
\]

5. Guess the initial density matrix \( \mathbf{D} \). The simplest guess is the null (zero) matrix. Hence, let \( \mathbf{D} = 0 \), where \( \mathbf{D} \) is \( K \times K \) in dimension, and \( K \) is the number of atomic orbitals in the basis.

6. Calculate the two-electron contribution \( \mathbf{P} \) to the Fock matrix with

\[
P_{\mu\nu} = \sum_\lambda \sum_\sigma D_{\lambda\sigma} \left( G_{\mu\nu\sigma\lambda} - \frac{1}{2} G_{\mu\lambda\sigma\nu} \right)
\]  

\[\text{(4.167)}\]

# K is the basis size (e.g., K = 7 for H2O in the STO-3G basis)
for m in range(K):
    for n in range(K):
        P[m,n] = 0.0
for l in range(K):
    for s in range(K):
        P[m,n] += D[l,s] * (G[m,n,s,l] - 0.5*G[m,l,s,n])

7. Add \( \mathbf{P} \) to the core Hamiltonian to obtain the Fock matrix:

\[
\mathbf{F} = \mathbf{H}^\text{core} + \mathbf{P}.
\]  

\[\text{(4.168)}\]

8. Transform the Fock matrix to the orthonormal basis with dot products involving the original Fock matrix \( \mathbf{F} \), and the transformation matrix \( \mathbf{X} \):

\[
\mathbf{F}' = \mathbf{XFX}.
\]  

\[\text{(4.169)}\]

It is easy to calculate dot products of matrices with the “at” symbol:
\[ F_p = X \odot F \odot X \]

9. Diagonalize \( F' \) to obtain its eigenvalues \( \epsilon \) and eigenvectors \( C' \).

\[ \epsilon, C_p = \text{linalg.eigh}(F_p) \]

10. Calculate the molecular orbitals \( C \):

\[ C = XC'. \] (4.170)

11. Form a new density matrix \( D \) from \( C \) using

\[ D_{\mu\nu} = \frac{2}{N/2} \sum a \, C_{\mu a} C_{\nu a}^*. \] (4.171)

# K is the basis size (e.g., \( K = 7 \) for \( H_2O \) in the STO-3G basis)
# N is the number of electrons (e.g., \( N = 10 \) for neutral water)

```
for m in range(K):
    for n in range(K):
        D[m,n] = 0.0
for a in range(int(N/2)):
    D[m,n] += 2 * (C[m,a] * C[n,a])
```

12. Determine whether self-consistency has been achieved (i.e., whether the Hartree–Fock electronic energy has converged). Do so by determining whether the Hartree–Fock energy expectation value \( E_e \)

\[ E_e = \frac{1}{2} \sum_{\mu} \sum_{\nu} D_{\nu\mu} (H_{\mu\nu}^{\text{core}} + F_{\mu\nu}) \] (4.172)

is the same as the energy expectation value of the previous iteration within a specified criterion (e.g., \( 10^{-7} E_h \)). If the procedure has not converged, return to step 6, passing it the newest density matrix \( D \), from step 11.

13. If the procedure has converged, append the nuclear-nuclear repulsion to the converged electronic energy \( E_e \) (from Step 12).

\[ E_{\text{tot}} = E_e + \sum_A \sum_{B > A} \frac{Z_A Z_B}{R_{AB}} \] (4.173)

183
You have solved Schrödinger’s equation for the system of interest! Specifically, you’ve found the total energy of the system (an expectation value), the eigenvector set of molecular orbitals, and the molecular orbital energies.
CHAPTER 5

CONCLUSION

The present dissertation began in Chapter 1 with a general overview of computational quantum chemistry, particularly with respect to those methods employed in the research found herein. A summary and conclusions for the research on phosgene (Chapter 2), the sulfur analogue of the Criegee intermediate (Chapter 3), and the evaluation of molecular integrals (Chapter 4) is now presented.

In Chapter 2, a high-level *ab initio* study of the sulfur analog of the Criegee intermediate (thioformaldehyde S-sulfide), its valence isomer (dithiirane), and the transition state linking these molecules was reported. The fundamental frequencies of thioformaldehyde S-sulfide and dithiirane, which had heretofore been reported using argon matrix isolation techniques, were predicted in the gas phase for the first time using coupled-cluster theory [CCSD(T)] and second-order vibrational-perturbation theory (VPT2). Vibration-rotation interaction constants ($\alpha^B$) and rotational constants ($B_e$ and $B_0$) were also reported. Relative electronic energies between stationary points on the isomerization pathway were reported at the complete basis set (CBS) limit of coupled-cluster theory with full quadruple excitations (CCSDTQ) by way of a focal point extrapolation. These results indicated that dithiirane is 7.2 kcal mol$^{-1}$ lower in energy than thioformaldehyde S-sulfide, and that there is a 27.0 kcal mol$^{-1}$ barrier to cyclization. Analysis of atomic charges, resonance structures, and chemical bonding were facilitated by Natural Bond Orbital (NBO) methods. The resonance contributors of the $C_s$-symmetric thioformaldehyde S-sulfide structure are predominantly characterized by the presence of a double-bond bond between the carbon and proximal sulfur atoms, and a third lone pair on the terminal sulfur atom, lending it a negative charge. The $C_{2v}$-symmetric structure dithiirane is formed by a ring-closing, electrocyclic mechanism, and features lengthened S—S and C—S bonds relative to the antecedent structure. Dithiirane’s S—S bond is almost exclusively covalent and of single-bond character, while its two C—S bonds are 88%
covalent. The geometric parameters of the transition state generally bear more resemblance to the thioformaldehyde $\text{S}$-sulfide reactant than the dithiirane product. The computed fundamental frequencies were in very good agreement with the argon matrix experiments of Maier and colleagues (2001), and the reported predictions of the currently unknown gas-phase infrared bands should be reliable. Comparisons to prior theoretical research were made where possible. It is hoped that it will be of utility to future experimental efforts to characterise the molecule’s gas-phase infrared spectrum.

In Chapter 3, phosgene was examined with high-level ab initio methods, building upon the extant theoretical literature by providing equilibrium geometry and fundamental vibrational frequencies of phosgene computed at significantly higher levels of electronic structure theory. Geometry optimization using first-derivatives extrapolated to the complete basis set (CBS) limit of coupled-cluster theory with full triple and perturbative quadruple excitations [CCSDT(Q)] yielded an equilibrium geometry ($\text{C}—\text{O} = 1.1768 \text{ Å}, \text{C}—\text{Cl} = 1.7373 \text{ Å}, \text{ and } \angle_{\text{O-C-Cl}} = 124.03^\circ$) in excellent agreement with the experimentally-derived values of Kuchitsu and coworkers, with an average deviation in bond length of 0.0005 Å from experiment. Fundamental frequencies were predicted using a “hybrid” approach that combines a CCSDT(Q)/CBS quadratic force field with CCSD(T)/cc-pCVQZ cubic and quartic force-fields to reduce the overall computational cost and leverage the rapid convergence of higher-order force constants with regard to level of theory and size of basis set. Anharmonic corrections to the vibrational transitions were predicted using CCSD(T) and the cc-pCVQZ, core-valence basis set. The predicted transitions displayed excellent agreement with the experimental values of Schnöckel, demonstrating the accuracy of this approach; however, further experimental examination of $\nu_4$ and $\nu_6$ is suggested. Vibration-rotation interaction constants ($\alpha_r^B$) and rotational constants ($B_e$) were presented. NBO computations show that the central carbon atom is electron deficient, with a natural valence of less than 3.5, which contributes to its activity as an electrophile. The dominant resonance structure (70%) features the expected CO double bond and CCl single bonds; the two minor resonance contributors (14% each) include a CO single bond and CO triple bond, respectively. While this work is now complete, it is hoped that experimental investigators will carefully examine modes $\nu_4$ and $\nu_6$.

A detailed, pedagogical introduction to molecular integrals constitutes Chapter 4. This was written, and submitted to Journal Chemical of Education, in order to ameliorate the often vague
conception of molecular integrals that even professional computational quantum chemists often suffer. It was designed to elucidate precisely what molecular integrals are, how their mathematical form is derived, and how exactly to implement them in code. It provides a careful introduction to a subject that most literature treats at an esoteric level and that tends to lack the clear detail most readers require.

The chapter features many characteristics that distinguish it from the few other introductory treatments of molecular integrals. First, it is an active-learning based introduction to the subject, having readers take part in the manipulation of equations and implementation in code. Secondly, it assumes no more than a sequence in physical chemistry, included the associated mathematics (calculus, but not linear algebra). It is written to be fully accessible to the undergraduate student. Finally, whereas the few accessible introductions to molecular integrals that exist are constrained to orbitals of low or no angular momentum for purposes of simplification, the present article details methods that are applicable to orbitals of arbitrary angular momentum, for all atoms, of any molecule, with any geometry.

The article’s structure is intended to have flexible utility to instructors and/or general readers. The programming project can be completed independently of the derivation. The reader who has programming experience and previously coded up Hartree–Fock need not refer to two of the four sub-chapters, but rather simply modify their code to read in their implementation of molecular integrals. The background piece provides a general introduction to computational quantum chemistry and the Hartree–Fock method that alone may be useful to some readers.

Early in the planning stages of this project, editors of *Journal of Chemical Education* were consulted for their perspective and advice. Among other suggestions, it was recommended that, rather than reporting quantitative, statistical data about student performance, a qualitative description of the student experience in engaging with the project would enrich the discussion. To this end, four students (two beginning graduate, two undergraduate) were guided through the molecular integrals programming project. The qualitative results stemming from a survey of their efforts are featured. Additionally provided are suggestions to instructors for scaffolding the material for students with different levels of experience.
BIBLIOGRAPHY


[130] Harris, D. C.; Bertolucci, M. D. Symmetry and Spectroscopy; Dover, 1989.


