The Dissertation Committee for Scott Michael Rabidoux
certifies that this is the approved version of the following dissertation:

Extending the Reach of Algorithms for the Calculation of Molecular Vibronic Spectra

Committee:

John Stanton, Supervisor

Victor Eijkhout, Co-Supervisor

Todd Arbogast

Robert van de Geijn

Dmitrii Makarov

Extending the Reach of Algorithms for the Calculation of Molecular Vibronic Spectra

by

Scott Michael Rabidoux, B.S., M.S.C.S.E.M.

DISSERTATION

Presented to the Faculty of the Graduate School of
The University of Texas at Austin
in Partial Fulfillment
of the Requirements
for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT AUSTIN
December 2016

Acknowledgments

I would like to first thank my advisor Dr. John Stanton for his guidance and support over the course of this work. I greatly appreciate his patience and understanding in guiding a Mathematical Economics major through a dissertation on computational spectroscopy.

I would like to also thank my co-advisor Dr. Victor Eijkhout for sharing with me his expertise in sparse linear algebra and high-performance computing. His support has played a major role in my overall development as a computational scientist.

In addition, I would like to thank Dr. Robert van de Geijn for welcoming me into his research group, introducing me to both John and Victor, and providing guidance through the CSEM program.

A special thanks to Dr. Robert Cave for his help in understanding the complexities of the *trans*-1,3-butadiene molecule.

Finally, I would like to thank my family and friends for their emotional support throughout my graduate studies.

Extending the Reach of Algorithms for the Calculation of Molecular Vibronic Spectra

by

Scott Michael Rabidoux, Ph.D.
The University of Texas at Austin, 2016

Supervisors: John Stanton
Victor Eijkhout

Theoretical spectroscopy is an important field of chemistry that can help extract useful information about the properties of a molecule from experimental spectral data. *Ab initio* calculations of molecular spectra can be performed and compared against experimental data to determine the validity of various calculated molecular properties. Unfortunately, the computational cost of these spectral simulations rises quickly with the number of atoms in the molecule of interest. As a result, current techniques for simulating molecular spectra are often limited for use with only the smallest of molecules. The main purpose of this work is to develop new computational tools in an effort to extend the reach of current state-of-the-art spectral simulation algorithms and allow for the spectroscopic study of larger molecules than is currently feasible.

The calculation of vibronic spectra requires the solution of the time-independent Schrödinger equation to obtain the vibronic energy levels of a molecule and their

corresponding transition intensities. When the Born-Oppenheimer approximation is applicable for the solution of the time-independent Schrödinger equation, the vibrational energy levels of a molecule can be easily determined analytically, if the harmonic approximation is used. What remains, then, for a spectral simulation, is the calculation of the transition intensities associated with each energy level. Under the harmonic approximation, the transition intensities (also known as Franck-Condon factors) can be calculated via a set of recurrence equations developed by Doktorov, Malkin, and Manko. The implementation of these recurrence equations, though, can be computationally intensive for medium-to-large molecules, especially for finite-temperature simulations. In this work, I present a new algorithm for the calculation of Franck-Condon factors via the Doktorov recurrence equations that achieves significantly better computational performance than existing implementations, with speedups of roughly thirty times on a single processor.

When the Born-Oppenheimer approximation is *not* applicable, vibronic coupling effects must be accounted for to achieve an accurate spectral simulation. A common approach for treating vibronic coupling effects is to solve the time-independent Schrödinger equation using a model Hamiltonian developed by Köppel, Domcke, and Cederbaum (KDC). Using the KDC approach, the problem of solving the Schrödinger equation becomes a problem of solving for the eigenstates of a large, sparse matrix. The computational difficulty of this problem is then dependent on the size of the matrix. Unfortunately, the size of the matrix at hand grows exponentially with the number of vibrational modes of the molecule of interest, and matrix dimensions can easily reach upwards of one billion and beyond.

In an attempt to make tractable problems involving very large matrices, I present in this work a distributed-memory parallelization strategy for the KDC approach. The resulting parallel algorithm achieves impressive parallel scalability and has been used to study several previously intractable spectroscopic problems.

I conclude this work by presenting *ab initio* calculations and spectral simulations for the molecule *trans*-1,3-butadiene. The spectroscopy of butadiene has been studied by expermentalists and theorist alike, but a complete KDC spectral model has yet to be achieved due in part to the large size of the resulting matrices. Using the newly-developed parallel algorithm described above, I am able to present spectra from simulations using the most complete KDC model for butadiene to date and discuss what these results may tell us about butadiene's electronic structure.

Table of Contents

List of Tables

List of Figures

Introduction

0.1 Spectroscopy

Spectroscopy is an important field of chemistry concerned with the understanding and measurement of spectra produced by the interaction between matter and electromagnetic radiation. Experimentally, the spectrum of a molecular system is measured by exposing the system to radiation of various energies and measuring the amount of absorption that occurs as a function of energy. The resulting graph of absorption versus energy is called an *absorption spectrum* and is an important measurable that contains a wealth of information about the molecule. The shape of a molecule's spectrum is determined by various molecular properties such as the vertical energies of the molecule's electronic states, the shape of each electronic state's potential energy surface, and the minimum-energy geometries of electronic states. Unfortunately, most of this information that is encapsulated in an experimental spectrum is not always easily interpretable.

Fortunately, theoretical models can be employed to help better understand experimental data and uncover much of this hidden information. *Ab initio* calculations of many of the above properties can be performed and used in spectral models in an attempt to recreate the experimental data. The accuracy of the computed spectrum can provide evidence for the validity of the calculated molecular properties (i.e. if the computed spectrum agrees with the experimental data, it's reasonable

to assume that the molecular properties used to create the computed spectrum are accurate as well). In addition to helping to better understand experimental data, theoretical spectroscopic models can also help predict the spectrum of molecules that have not yet been studied experimentally.

0.1.1 Theory of spectroscopy

The absorption of radiation causes an atom or molecule to *transition* from an initial energy state to a higher energy state. The energy states of a molecule are quantized, meaning only certain discrete values of energy are possible to attain (i.e. there is not a continuous spread of energy levels available). These discrete values of energy, E, are called *energy levels*, and each energy level has associated with it a quantum mechanical *state* (sometimes more than one if the energy level is degenerate). Each state is described by a wave function $\Psi(\mathbf{r}, \mathbf{R})$ that depends on electronic ($\mathbf{r}$) and nuclear ($\mathbf{R}$) coordinates.

The absorption of radiation by a molecule can only occur if the energy of the radiation is **exactly** equal to the energy difference between the molecule's current state and an attainable higher-energy state. Absorption is not guaranteed, however, even if the available radiation has energy appropriate for a possible transition. Rather, each possible transition has some probability of occurring, given that the appropriate radiation is available for absorption. The likelihood of a transition occurring (through absorption) is called its *transition intensity* and is calculated using the formula

$$I = P^2 = [\langle \Psi' | \, \boldsymbol{\mu} \, | \Psi'' \rangle]^2 = \left[\int \Psi'^* \boldsymbol{\mu} \Psi'' \, \mathrm{d}\tau \right]^2, \tag{1}$$

where $\boldsymbol{\mu}$ is the molecular dipole operator

$$\boldsymbol{\mu} = \boldsymbol{\mu}_e + \boldsymbol{\mu}_n = -e\sum_i \mathbf{r}_i + e\sum_j Z_j \mathbf{R}_j, \tag{2}$$

the functions Ψ' and Ψ'' are molecular wave functions for the final and initial states, respectively, and $\mathrm{d}\tau$ is a volume element in nuclear and electronic coordinate space.

The possible energy levels and corresponding wave functions of a molecule can be solved for using the time-independent Schrödinger equation,

$$\hat{H}\Psi_k(\mathbf{r}, \mathbf{R}) = E\Psi_k(\mathbf{r}, \mathbf{R}), \tag{3}$$

where $\hat{H}$ is the molecular Hamiltonian operator, $\Psi_k(\mathbf{r}, \mathbf{R})$ is a molecular wave function that depends on electronic and nuclear coordinates, and E is the total energy of the system described by $\Psi_k(\mathbf{r}, \mathbf{R})$. The electronic coordinates $\mathbf{r}$ and nuclear coordinates $\mathbf{R}$ are both $3N$-dimensional Cartesian coordinates, where N is the number of atoms in the molecule.

The molecular Hamiltonian can be written as the sum of five operators:

$$\hat{H} = \hat{T}_n + \hat{T}_e + \hat{U}_{en} + \hat{U}_{ee} + \hat{U}_{nn}. \tag{4}$$

The operator $\hat{T}_n$ describes the kinetic energy of each nucleus in the system, $\hat{T}_e$ describes the kinetic energy of each electron in the system, and $\hat{U}_{en}$, $\hat{U}_{ee}$, and $\hat{U}_{nn}$ describe the Coulombic potential energy of all electron-nucleus, electron-electron, and nucleus-nucleus interactions, respectively. Let us define an electronic Hamiltonian operator, $\hat{H}_e$, as

$$\hat{H}_e = \hat{T}_e + \hat{U}_{en} + \hat{U}_{ee} + \hat{U}_{nn}. \tag{5}$$

The electronic Hamiltonian describes the energy of the electrons in the electrostatic field of the nuclei, where the nuclei are assumed to be stationary. The full molecular Hamiltonian can then be written as

$$\hat{H} = \hat{T}_n + \hat{H}_e. \tag{6}$$

The molecular Schrödinger equation is far too complex to be solved exactly, save for the very simplest of molecular systems. Approximations must be made for the problem to become tractable. Two types of approximations are considered in this work: The Born-Oppenheimer approximation (Chapter 1) and an approximate vibronic coupling model developed by Köppel, Domcke, and Cederbaum (KDC) (Chapter 2).

0.2 Computational challenges in theoretical spectroscopy

Even when approximations are made, the solution of the Schrödinger equation and the calculation of transition intensities represent major computational challenges for all but the smallest of molecules. When the Born-Oppenheimer approximation is applicable, the vibronic energy levels of a molecule can be easily determined analytically under the harmonic approximation. What remains, then, is the calculation of the corresponding transition intensities (also known as Franck-Condon factors). The calculation of Franck-Condon factors commonly involves an implementation of a set of recurrence equations developed by Doktorov *et al.*[28]. As the molecules that we wish to study increase in size, the number of values that need to be determined via the recurrence equations grows approximately exponen-

tially with the number of vibrational modes in the system. Much research has been done to reduce the computational workload of these calculations [39, 23, 22, 73, 6, 86, 87, 48, 42], but the preferred computer implementation of the recurrence equations is still up for debate.

When the Born-Oppenheimer approximation breaks down, vibronic coupling effects must be considered. A common approach for treating vibronic coupling effects is to solve the time-independent Schrödinger equation using a model Hamiltonian developed by Köppel, Domcke, and Cederbaum (KDC). Computing a spectrum using the KDC model involves using a variational approach that requires the eigensolution of large, sparse matrices. As with the Franck-Condon factor calculations, the size of the KDC model calculations grows exponentially with the number of vibrational modes in the system and can easily result in matrices of dimension one billion and beyond. Much effort has been made to reduce the computational workload of these calculations as well [90, 89], and I aim to improve upon this work by developing a novel approach for the parallelization of the KDC algorithm.

More detail regarding the exact computational costs of each of these of approaches will be given in Chapters 1 and 2. Improving the computational performance of these types of algorithms can help advance the field of theoretical spectroscopy by allowing for the study of molecules too large to be considered currently. Achieving this goal is the primary purpose of this work.

0.3 Dissertation outline

In Chapter 1, I will introduce the Born-Oppenheimer (BO) approximation for solving the Schrödinger equation and explain how it can be used to compute a vibronic spectrum. I will discuss commonly used approaches for calculating transition intensities under the BO approximation and their computational inefficiencies. I will propose a new algorithm for calculating transition intensities that alleviates many of these inefficiencies and achieves significant computational speedup over existing approaches.

In Chapter 2, I will discuss under what circumstances the Born-Oppenheimer approximation breaks down and the treatment of vibronic coupling effects in the solution of the Schrödinger equation becomes necessary. I will introduce the KDC vibronic coupling model and how it is used and implemented for the calculation of vibronic spectra. I will then motivate and describe, in detail, my approach to the parallelization of the computation of vibronic spectra using the KDC model.

In Chapter 3, I will introduce the problem of simulating vibronic spectra for the lowest-energy electronic transition of the molecule 1,3-*trans*-butadiene. I will report the results of *ab initio* calculations of the potential energy surfaces related to this electronic transition and report results of the calculation of the vibronic spectrum.

Chapter 1

Born-Oppenheimer approximation

The Born-Oppenheimer approximation has been a staple of computational chemistry for decades. It simplifies the solution of the Schrödinger equation by allowing for the separation of electronic and nuclear coordinates (**r** and **R**), breaking down the full problem of solving the Schrödinger equation (equation 3) into two smaller (and much more manageable) sub-problems:

1. Fix the nuclear geometry **R** and solve for the electronic wave functions $\psi_i(\mathbf{r};\mathbf{R})$ by solving for the eigenstates of electronic Hamiltonian,

$$\hat{H}_e\psi_i(\mathbf{r};\mathbf{R}) = V_i(\mathbf{R})\psi_i(\mathbf{r};\mathbf{R}). \tag{1.1}$$

   Electronic wave functions defined in this way are called *adiabatic electronic wave functions*. Repeat this step for varying geometries **R**, and sew together the resulting values of $V_i(\mathbf{R})$ to determine the energy of the electronic wave

---

Parts of this chapter are based on a prior publication:

Rabidoux, S.M.; Eijkhout, V.; Stanton, J.F. A Highly-Efficient Implementation of the Doktorov Recurrence Equations for Franck-Condon Calculations. *Journal of Chemical Theory and Computation*, 12(2), **2016**, 728-739.

The co-authors Stanton, J.F. and Eijkhout, V. supervised the project.

function as a function of nuclear geometry. These functions, $V_i(\mathbf{R})$, are called *adiabatic potential energy surfaces*.

2. Solve for the nuclear wave functions, $\Omega_j^{(i)}(\mathbf{R})$, within each electronic state of interest by solving the nuclear Schrödinger equation,

$$[\hat{T}_n + V_i(\mathbf{R})]\Omega_j^{(i)}(\mathbf{R}) = E_{ij}\Omega_j^{(i)}(\mathbf{R}). \tag{1.2}$$

The full molecular wave functions are then taken to be products of electronic and nuclear wave functions,

$$\Psi_{ij}(\mathbf{r}, \mathbf{R}) = \psi_i(\mathbf{r}; \mathbf{R})\Omega_j^{(i)}(\mathbf{R}). \tag{1.3}$$

In the Born-Oppenheimer framework, the discrete energy levels that a molecule can attain can be written as a sum of contributions of electronic and nuclear components:

$$E = E_{\text{electronic}} + E_{\text{vibrational}} \tag{1.4}$$

Any other contributions (rotational, translational, etc.) to the total energy are too small for the scope of this work and are neglected. The contributions to the total energy can be categorized in a hierarchical structure. The largest contribution comes from the *electronic state* of the molecule (described by $\psi_i(\mathbf{r}; \mathbf{R})$), which corresponds to a certain configuration of electrons. Within each electronic state are many *vibrational states* (described by $\Omega_j^{(i)}(\mathbf{R})$), which correspond to various states of internal nuclear motion. A pictorial description of electronic and vibrational energy levels is shown in Figure 1.1.

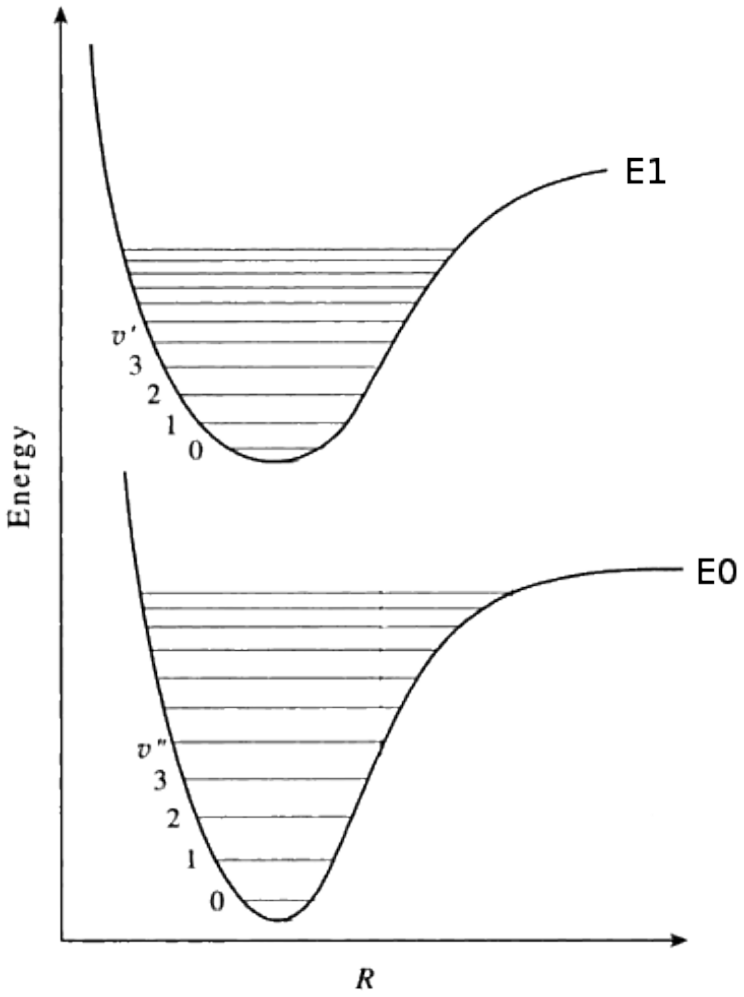


Figure 1.1: An illustration of vibrational state energy levels within electronic states. The lines labeled E0 and E1 represent the potential energy surfaces of two electronic states (the x-axis can be thought of some measure of nuclear configuration in coordinate space). The lines within each electronic potential well represent vibrational energy levels within each electronic energy level.

1.1 Derivation of the Born-Oppenheimer approximation

Let us start from the *exact* time-independent molecular Schrödinger equation:

$$\hat{H}\Psi(\mathbf{r},\mathbf{R}) = [\hat{T}_n + \hat{H}_e]\Psi(\mathbf{r},\mathbf{R}) = E\Psi(\mathbf{r},\mathbf{R}). \tag{1.5}$$

Suppose we have a set of electronic eigenfunctions $\{\psi_i(\mathbf{r};\mathbf{R})\}$; that is, we have solved

$$\hat{H}_e\psi_i(\mathbf{r};\mathbf{R}) = V_i(\mathbf{R})\psi_i(\mathbf{r};\mathbf{R}) \quad \text{for} \quad i = 1, 2, \ldots \tag{1.6}$$

The electronic eigenfunctions $\psi_i(\mathbf{r};\mathbf{R})$ are taken to be real, which is possible when there are no magnetic or spin interactions. The parametric dependence of the functions $\psi_i(\mathbf{r};\mathbf{R})$ on the nuclear coordinates is indicated by the symbol after the semicolon. This means that, although $\psi_i(\mathbf{r};\mathbf{R})$ is a real-valued function of $\mathbf{r}$, its functional form depends on $\mathbf{R}$.

Since the electronic eigenfunctions $\psi_i(\mathbf{r};\mathbf{R})$ form a complete basis in $L^2(\mathbf{r})$ at the nuclear configuration $\mathbf{R}$, the full molecular wave function $\Psi(\mathbf{r},\mathbf{R})$ can be expanded in terms of ψ_i:

$$\Psi(\mathbf{r},\mathbf{R}) = \sum_i \psi_i(\mathbf{r};\mathbf{R})\Omega^{(i)}(\mathbf{R}). \tag{1.7}$$

By substituting this functional form into the exact Schrödinger equation (equation 1.5), left multiplying by the real function $\psi_{i'}(\mathbf{r};\mathbf{R})$, and integrating over the electronic coordinates $\mathbf{r}$, the total Schrödinger equation (equation 1.5) becomes a set of coupled eigenvalue equations depending on nuclear coordinates only:

$$[\mathbb{H}_n(\mathbf{R}) + \mathbb{H}_e(\mathbf{R})]\,\boldsymbol{\Omega}(\mathbf{R}) = E\boldsymbol{\Omega}(\mathbf{R}). \tag{1.8}$$

The column vector $\mathbf{\Omega}(\mathbf{R})$ has elements $\Omega^{(i)}(\mathbf{R})$. The matrix $\mathbb{H}_e(\mathbf{R})$ has entries

$$(\mathbb{H}_e(\mathbf{R}))_{i'i} = \langle \psi_{i'}(\mathbf{r};\mathbf{R})| \, \hat{H}_e \, |\psi_i(\mathbf{r};\mathbf{R})\rangle \tag{1.9}$$

and is diagonal, since

$$\langle \psi_{i'}(\mathbf{r};\mathbf{R})| \, \hat{H}_e \, |\psi_i(\mathbf{r};\mathbf{R})\rangle = V_i(\mathbf{R}) \, \langle \psi_{i'}(\mathbf{r};\mathbf{R})|\psi_i(\mathbf{r};\mathbf{R})\rangle \tag{1.10}$$

$$= V_i(\mathbf{R})\delta_{i'i}, \tag{1.11}$$

due to the orthogonality of the eigenfunctions of $\hat{H}_e$. The matrix, $\mathbb{H}_n(\mathbf{R})$, on the other hand, is non-diagonal with entries

$$(\mathbb{H}_n(\mathbf{R}))_{i'i} = \langle \psi_{i'}(\mathbf{r};\mathbf{R})| \, \hat{T}_n \, |\psi_i(\mathbf{r};\mathbf{R})\rangle \, . \tag{1.12}$$

Each of these entries can be expanded as

$$\langle \psi_i(\mathbf{r};\mathbf{R})| \, \hat{T}_n \, |\psi_{i'}(\mathbf{r};\mathbf{R})\rangle = \sum_\alpha \frac{-\hbar\omega_\alpha}{2} \, \langle \psi_i(\mathbf{r};\mathbf{R})| \, \nabla_\alpha^2 \, |\psi_{i'}(\mathbf{r};\mathbf{R})\rangle \tag{1.13}$$

$$= \sum_\alpha \frac{-\hbar\omega_\alpha}{2} \, \langle \psi_i(\mathbf{r};\mathbf{R})|\psi_{i'}(\mathbf{r};\mathbf{R})\rangle \, \nabla_\alpha^2 \tag{1.14}$$

$$+ \sum_\alpha -\hbar\omega_\alpha \, \langle \psi_i(\mathbf{r};\mathbf{R})|\nabla_\alpha\psi_{i'}(\mathbf{r};\mathbf{R})\rangle \, \nabla_\alpha \tag{1.15}$$

$$+ \sum_\alpha \frac{-\hbar\omega_\alpha}{2} \, \langle \psi_i(\mathbf{r};\mathbf{R})|\nabla_\alpha^2\psi_{i'}(\mathbf{r};\mathbf{R})\rangle \, , \tag{1.16}$$

where α indexes the nuclear coordinates. For off-diagonal entries of $\mathbb{H}_n(\mathbf{R})$ ($i \neq i'$), the first term (1.14) vanishes due to orthogonality of $\psi_i(\mathbf{r};\mathbf{R})$. The term $\langle \psi_i(\mathbf{r};\mathbf{R})|\nabla_\alpha\psi_{i'}(\mathbf{r};\mathbf{R})\rangle$

in the above expression can be expanded as

$$\langle\psi_i(\mathbf{r};\mathbf{R})|\nabla_\alpha\psi_{i'}(\mathbf{r};\mathbf{R})\rangle = \frac{V_{i'}(\mathbf{R})-V_i(\mathbf{R})}{V_{i'}(\mathbf{R})-V_i(\mathbf{R})}\langle\psi_i(\mathbf{r};\mathbf{R})|\nabla_\alpha\psi_{i'}(\mathbf{r};\mathbf{R})\rangle \tag{1.17}$$

$$= \frac{\langle\psi_i(\mathbf{r};\mathbf{R})|\,\nabla_\alpha\hat{H}_e\,|\psi_{i'}(\mathbf{r};\mathbf{R})\rangle - \langle\psi_i(\mathbf{r};\mathbf{R})|\,\hat{H}_e\nabla_\alpha\,|\psi_{i'}(\mathbf{r};\mathbf{R})\rangle}{V_{i'}(\mathbf{R})-V_i(\mathbf{R})} \tag{1.18}$$

$$= \frac{\langle\psi_i(\mathbf{r};\mathbf{R})|\,[\nabla_\alpha,\hat{H}_e]\,|\psi_{i'}(\mathbf{r};\mathbf{R})\rangle}{V_{i'}(\mathbf{R})-V_i(\mathbf{R})}. \tag{1.19}$$

If all surfaces $V_i(\mathbf{R})$ are energetically well-separated, (if $|V_{i'}(\mathbf{R})-V_i(\mathbf{R})| \gg 0$ for all $i \neq i'$) the off-diagonal terms (1.19) can be neglected. The third term (1.16) in the expansion can be approximately written as the square of term (1.15) and can thus also be neglected. The nuclear eigenvalue equations then become an uncoupled set of equations

$$\left[\hat{T}_n + V_i(\mathbf{R})\right]\Omega^{(i)}(\mathbf{R}) = E\Omega^{(i)}(\mathbf{R}) \quad \text{for} \quad i = 1,2,\ldots, \tag{1.20}$$

the solution of which is the second step of the Born-Oppenheimer approach described at the beginning of this section.

It is important to note again that the Born-Oppenheimer approach **is only accurate when the electronic states of interest are energetically well-separated**. For many problems of interest, this is indeed the case and the Born-Oppenheimer approximation is an extremely useful tool for simplifying the solution of the Schrödinger equation. However, there exist systems that we wish to study that do not satisfy this condition (two or more electronic states are close in energy). When this is the case, the Born-Oppenheimer approximation breaks down and alternative methods must be used. This chapter focuses on the case in which the Born-Oppenheimer approximation is accurate. The alternative case is considered in Chapter 2.

1.2 Using the Born-Oppenheimer approximation

The Born-Oppenheimer approach to solving the Schrödinger equation is a two-step process. In the first step, one must solve for the adiabatic potential energy surface $V_i(\mathbf{R})$ of interest via the equation

$$\hat{H}_e \psi_i(\mathbf{r}; \mathbf{R}) = V_i(\mathbf{R}) \psi_i(\mathbf{r}; \mathbf{R}). \tag{1.21}$$

There are *many* available techniques for solving the above equation for a fixed $\mathbf{R}$ (Hartree-Fock, Möller-Plesset perturbation theory, Configuration Interaction, Coupled cluster, just to name a few), with varying degrees of accuracy. The potential energy $V_i(\mathbf{R})$ is solved for at several different nuclear configurations $\mathbf{R}$, and the resulting energies are "sewn" together to form the function $V_i(\mathbf{R})$. The second step of the Born-Oppenheimer approach is then to solve the nuclear Schrödinger equation

$$[\hat{T}_n + V_i(\mathbf{R})]\Omega_j^{(i)}(\mathbf{R}) = E_{ij}\Omega_j^{(i)}(\mathbf{R}), \tag{1.22}$$

given some potential energy surface $V_i(\mathbf{R})$.

1.2.1 Solving the nuclear Schrödinger equation

The kinetic energy operator $\hat{T}_n$ is defined as

$$\hat{T}_n = \sum_{\alpha=1}^{N} -\frac{\hbar^2}{2m_\alpha}\left[\frac{\partial^2}{\partial x_\alpha^2} + \frac{\partial^2}{\partial y_\alpha^2} + \frac{\partial^2}{\partial z_\alpha^2}\right] \tag{1.23}$$

and any potential function V can be expanded as a Taylor series around its minimum energy geometry, $\mathbf{R}_0$ (where $\mathbf{R}_0$ contains equilibrium Cartesian coordinates

$(x_0^{(\alpha)}, y_0^{(\alpha)}, z_0^{(\alpha)})$ for each atom α):

$$\begin{aligned} V &= V_0 \\ &+ \sum_{i=1}^{N} \left[\left.\frac{\partial V}{\partial x_i}\right|_{x_0^{(i)}} (x_i - x_0^{(i)}) + \left.\frac{\partial V}{\partial y_i}\right|_{y_0^{(i)}} (y_i - y_0^{(i)}) + \left.\frac{\partial V}{\partial z_i}\right|_{z_0^{(i)}} (z_i - z_0^{(i)}) \right] \\ &+ \text{higher-order terms.} \end{aligned} \tag{1.24}$$

We can greatly simplify the above expression by changing our coordinate system to use *mass-weighted Cartesian displacement coordinates*:

$$\tilde{x}_1 = \sqrt{m_1}\Delta x_1 \quad \tilde{x}_2 = \sqrt{m_1}\Delta y_1 \quad \tilde{x}_3 = \sqrt{m_1}\Delta z_1 \quad \tilde{x}_4 = \sqrt{m_2}\Delta x_2, \tag{1.25}$$

where $\Delta x_i = (x_i - x_0^{(i)})$ (with similar definitions for y and z coordinates). In this new coordinate system, the kinetic energy operator becomes

$$\hat{T}_n = -\frac{\hbar^2}{2} \sum_{i=1}^{3N} \frac{\partial^2}{\partial \tilde{x}_i^2} \tag{1.26}$$

$$= -\frac{\hbar^2}{2} \frac{\boldsymbol{\partial}}{\boldsymbol{\partial}\tilde{\mathbf{x}}}^T \frac{\boldsymbol{\partial}}{\boldsymbol{\partial}\tilde{\mathbf{x}}} \tag{1.27}$$

where

$$\frac{\boldsymbol{\partial}}{\boldsymbol{\partial}\tilde{\mathbf{x}}} = \left[\frac{\partial}{\partial \tilde{x}_1}, \frac{\partial}{\partial \tilde{x}_2}, \ldots, \frac{\partial}{\partial \tilde{x}_{3N}} \right]^T. \tag{1.28}$$

The potential energy takes the form

$$V = V_0 + \underbrace{\sum_{i=1}^{3N} \left.\frac{\partial V}{\partial \tilde{x}_i}\right|_0 \tilde{x}_i}_{0} + \frac{1}{2} \sum_{i,j=1}^{3N} \left.\frac{\partial^2 V}{\partial \tilde{x}_i \tilde{x}_j}\right|_0 \tilde{x}_i \tilde{x}_j + \text{higher-order terms,} \tag{1.29}$$

or

$$V = V_0 + \underbrace{\sum_{i=1}^{3N} f_i \tilde{x}_i}_{0} + \frac{1}{2} \sum_{i,j=1}^{3N} f_{ij} \tilde{x}_i \tilde{x}_j + \text{higher-order terms.} \tag{1.30}$$

Note that the first-order terms are zero since the expansion point was chosen to be the minimum of the potential energy surface. For relatively small displacements around the equilibrium point, the higher-order terms (cubic, quartic, etc.) can be neglected. For spectroscopic study, the geometry at which we are interested in solving the nuclear Schrödinger equation is the absorbing state equilibrium geometry. As long as the excited state equilibrium geometry, $\mathbf{R}_0$, is not significantly displaced from the absorbing state equilibrium geometry (true for many problems), neglection of the higher-order terms is a reasonable approximation. This truncation of the potential Taylor series to quadratic terms is known as the *harmonic approximation* and is a widely-used approach for simplifying the solution of the nuclear Schrödinger equation. The result is a potential function of the form

$$V = \frac{1}{2}\left(\begin{bmatrix} \tilde{x}_1 & \tilde{x}_2 & \dots & \tilde{x}_{3N} \end{bmatrix} \begin{bmatrix} f_{1,1} & f_{1,2} & \dots & f_{1,3N} \\ f_{2,1} & f_{2,2} & \dots & f_{2,3N} \\ \vdots & \vdots & \ddots & \vdots \\ f_{3N,1} & f_{3N,2} & \dots & f_{3N,3N} \end{bmatrix} \begin{bmatrix} \tilde{x}_1 \\ \tilde{x}_2 \\ \vdots \\ \tilde{x}_{3N} \end{bmatrix} \right) \tag{1.31}$$

$$= \frac{1}{2}(\tilde{\mathbf{x}}^T \mathbf{F} \tilde{\mathbf{x}}). \tag{1.32}$$

Together with the kinetic energy operator, the harmonic potential in terms of mass-weighted Cartesian displacement coordinates create the Hamiltonian operator of a $(3N)$-dimensional *coupled* harmonic oscillator system. To uncouple the system, we will adopt another change of coordinate system to what are known as *normal coordinates*, Q_i, each of which corresponds to a *normal mode of vibration*.

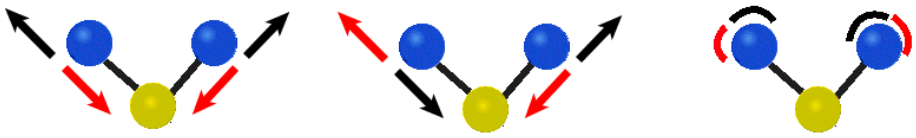


Figure 1.2: Graphical representation of the three normal modes of water. The left most picture is a symmetric stretch (both hydrogen atoms either stretching or compressing their bonds with the oxygen atom). The center is an asymmetric stretch (when one bond compresses, the other stretches). The right most picture is a bending motion (the angle between the two bonds either grows or shrinks).

1.2.1.1 Normal modes of vibration

There are $3N$ degrees of freedom that describe nuclear motion within a molecule, each of which can be categorized into one of three groups: translation, rotation, and vibration. Translation refers to the movement of a molecule with fixed orientation and internuclear distances through space. Since translational movement can occur in x, y, and z directions, there are three translational degrees of freedom. Translational energy is not relevant to spectroscopy (as it is very small and unquantized), so the three translational degrees of freedom are ignored in this work.

For non-linear molecules, rotational motion can be described in terms of rotations around three axes $\{x, y\}$, $\{x, z\}$, and $\{y, z\}$, so there are three rotational degrees of freedom. For linear molecules, rotational along its own axis leaves the molecule unchanged, so there are only two rotational degrees of freedom. Rotational energy *is* quantized, but contributes much less to the overall energy of the molecule than vibrational energy, so I also ignore rotational degrees of freedom in

this work.

What remains are $3N - 6$ (or $3N - 5$ for linear molecules) vibrational degrees of freedom. Each vibrational degree of freedom has associated with it a *normal mode* which describes the collective motion of all atoms along some fixed path, determined by the potential energy surface on which the molecule resides. Some properties of normal modes are:

1. A normal mode is a molecular vibration where some or all atoms vibrate together with the same frequency in a defined manner.

2. All vibrational movement of a molecule can be defined as linear combinations of normal modes (the normal modes are a basis for vibrational motion).

3. Normal modes are orthogonal.

4. Each mode has a definite frequency of vibration.

An example of the three normal modes of water (H_2O) is shown in Figure 1.2. Because all vibrational movement can be described as a superposition of the normal modes, it is possible to completely describe the location of the nuclei in a molecule by specifying how far the nuclei have traveled away from some equilibrium position, along each normal mode. Let our equilibrium position be labeled $\mathbf{Q} = 0$. The distance traveled away from the equilibrium position along each normal mode is encapsulated in a *normal coordinate*, Q_i, which can take on negative and positive values, depending on which way the nuclei travel away from the equilibrium position.

1.2.1.2 Solving for the normal coordinates of given electronic state

The normal coordinates determined by a particular potential energy surface are related to the mass-weighted Cartesian displacement coordinates by the orthonormal transformation

$$\tilde{\mathbf{x}} = \mathbf{A}^T\mathbf{Q} \tag{1.33}$$

or

$$\begin{bmatrix} \tilde{x}_1 \\ \tilde{x}_2 \\ \vdots \\ \tilde{x}_{3N} \end{bmatrix} = \left[\begin{array}{c} \vec{a}_1^T \\ \hline \vec{a}_2^T \\ \hline \vdots \\ \hline \vec{a}_{3N}^T \end{array}\right] \begin{bmatrix} Q_1 \\ Q_2 \\ \vdots \\ Q_{3N} \end{bmatrix}, \tag{1.34}$$

where $\vec{a}_i$ are eigenvectors of the Hessian matrix, $\mathbf{F}$, such that

$$\mathbf{AFA}^T = \mathbf{D} = \text{diag}(\omega_1^2, \omega_2^2, \ldots, \omega_{3N}^2). \tag{1.35}$$

The terms ω_i are the frequencies corresponding to each normal mode. Revisiting equation 1.27, in normal coordinates, the kinetic energy operator becomes

$$T = -\frac{\hbar^2}{2}\left(\mathbf{A}^T\frac{\boldsymbol{\partial}}{\boldsymbol{\partial}\mathbf{Q}}\right)^T\mathbf{A}^T\frac{\boldsymbol{\partial}}{\boldsymbol{\partial}\mathbf{Q}} \tag{1.36}$$

$$= -\frac{\hbar^2}{2}\frac{\boldsymbol{\partial}}{\boldsymbol{\partial}\mathbf{Q}}^T\mathbf{A}\mathbf{A}^T\frac{\boldsymbol{\partial}}{\boldsymbol{\partial}\mathbf{Q}} \tag{1.37}$$

$$= -\frac{\hbar^2}{2}\frac{\boldsymbol{\partial}}{\boldsymbol{\partial}\mathbf{Q}}^T\frac{\boldsymbol{\partial}}{\boldsymbol{\partial}\mathbf{Q}} \tag{1.38}$$

$$= -\frac{\hbar^2}{2}\sum_{i=1}^{3N}\frac{\partial}{\partial Q_i^2} \tag{1.39}$$

since the matrix of eigenvectors $\mathbf{A}$ is orthogonal, and $\dfrac{\boldsymbol{\partial}}{\boldsymbol{\partial}\tilde{\mathbf{x}}} = \mathbf{A}^T\dfrac{\boldsymbol{\partial}}{\boldsymbol{\partial}\mathbf{Q}}$ since $\tilde{\mathbf{x}}$ and $\mathbf{Q}$ are related by a linear operator and differentiation is a linear operator. Revisiting

equation 1.32, we can write V using the matrix form

$$V = \frac{1}{2}\left(\begin{bmatrix} \tilde{x}_1 & \tilde{x}_2 & \dots & \tilde{x}_{3N} \end{bmatrix} \begin{bmatrix} f_{1,1} & f_{1,2} & \dots & f_{1,3N} \\ f_{2,1} & f_{2,2} & \dots & f_{2,3N} \\ \vdots & \vdots & \ddots & \vdots \\ f_{3N,1} & f_{3N,2} & \dots & f_{3N,3N} \end{bmatrix} \begin{bmatrix} \tilde{x}_1 \\ \tilde{x}_2 \\ \vdots \\ \tilde{x}_{3N} \end{bmatrix} \right) \tag{1.40}$$

$$= \frac{1}{2}(\tilde{\mathbf{x}}^T \mathbf{F} \tilde{\mathbf{x}}). \tag{1.41}$$

Changing the coordinate system to normal coordinates gives the potential

$$V = \frac{1}{2}(\mathbf{Q}^T \mathbf{A}\mathbf{F}\mathbf{A}^T \mathbf{Q}) \tag{1.42}$$

$$= \frac{1}{2}(\mathbf{Q}^T \mathbf{D} \mathbf{Q}) \tag{1.43}$$

where $\mathbf{D}$ = diag$(\omega_1^2, \omega_2^2, \dots, \omega_{3N}^2)$

$$= \frac{1}{2}\sum_{i=1}^{3N} \omega_i^2 Q_i^2. \tag{1.44}$$

In terms of the normal coordinates Q_i, then, the kinetic and potential energies of vibration are given by equations 1.39 and 1.44, respectively. The nuclear Schödinger equation will then have the form

$$\hat{H}\Omega(\mathbf{Q}) = \left[-\frac{\hbar^2}{2}\sum_{i=1}^{3N} \frac{\partial^2}{\partial Q_i^2} + \frac{1}{2}\sum_{i=1}^{3N} \omega_i^2 Q_i^2 \right] \Omega(\mathbf{Q}) = E\Omega(\mathbf{Q}) \tag{1.45}$$

where the Hamiltonian operator $\hat{H}$ is that of an *uncoupled* $3N$-dimensional harmonic oscillator. After removing the modes associated with translational and rotational movement, equation 1.45 becomes

$$\hat{H}\Omega(\mathbf{Q}) = \left[-\frac{\hbar^2}{2}\sum_{i=1}^{m} \frac{\partial^2}{\partial Q_i^2} + \frac{1}{2}\sum_{i=1}^{m} \omega_i^2 Q_i^2 \right] \Omega(\mathbf{Q}) \tag{1.46}$$

$$= \left[\sum_{i=1}^{m} \left(-\frac{\hbar^2}{2}\frac{\partial^2}{\partial Q_i^2} + \frac{1}{2}\omega_i^2 Q_i^2 \right) \right] \Omega(\mathbf{Q}) = E\Omega(\mathbf{Q}), \tag{1.47}$$

where m is the number of vibrational modes in the system, and describes an m-dimensional uncoupled harmonic oscillator system. The nuclear wave functions of 1.47 are easily determined analytically, as we will see in the next section.

1.2.2 Quantum harmonic oscillators

Quantum harmonic oscillators are the quantum-mechanical analogs of the classical harmonic oscillator. Because an arbitrary potential can usually be well-approximated as a harmonic potential in the vicinity of a stable equilibrium point, the harmonic oscillator is one of the most important model systems in quantum mechanics and is widely used in theoretical vibronic spectroscopy.

A quantum harmonic oscillator is a system that has the following Hamiltonian operator:

$$\hat{H} = -\frac{\hbar^2}{2m_\alpha}\frac{\partial^2}{\partial x^2} + \frac{1}{2}m_\alpha\omega^2 x^2, \tag{1.48}$$

where m_α is the particle's mass and x is the position of the particle in terms of its displacement from some equilibrium point ($x = 0$).

The harmonic oscillator is one of the few quantum-mechanical systems for which an exact, analytical solution is known. With the Hamiltonian operator in equation 1.48, the time-independent Schrödinger equation has solutions

$$\chi_v(x) = \frac{1}{\sqrt{2^v v!}} \cdot \left(\frac{m_\alpha\omega}{\pi\hbar}\right)^{1/4} \cdot \exp\left(-\frac{m_\alpha\omega x^2}{2\hbar}\right) \cdot H_v\left(\sqrt{\frac{m_\alpha\omega}{\hbar}}x\right) \tag{1.49}$$

for $v \in \{0, 1, 2, \dots\}$, where the functions $H_v(z)$ are the physicists' Hermite polynomials

$$H_v(z) = (-1)^v e^{z^2}\frac{d^v}{dz^v}\left(e^{-z^2}\right). \tag{1.50}$$

The corresponding energy levels are

$$E_v = \hbar\omega\left(v + \frac{1}{2}\right). \tag{1.51}$$

The harmonic oscillator in terms of mass-weighted coordinates ($\tilde{x} = \sqrt{m_\alpha} \cdot x$) is described by a Hamiltonian of the form

$$\hat{H} = -\frac{\hbar^2}{2}\frac{\partial^2}{\partial\tilde{x}^2} + \frac{1}{2}\omega^2\tilde{x}^2, \tag{1.52}$$

which is the same functional form as each term of the summation in equation 1.47. The resulting wave functions are

$$\chi_v(\tilde{x}) = \frac{1}{\sqrt{2^v v!}} \cdot \left(\frac{\omega}{\pi\hbar}\right)^{1/4} \cdot \exp\left(-\frac{\omega\tilde{x}^2}{2\hbar}\right) \cdot H_v\left(\sqrt{\frac{\omega}{\hbar}}\tilde{x}\right) \tag{1.53}$$

with the same energies as in equation 1.51. So, the wave functions for each 1-D Hamiltonian operator in the summation in equation 1.47 are then

$$\chi_v(Q_i) = \frac{1}{\sqrt{2^v v!}} \cdot \left(\frac{\omega_i}{\pi\hbar}\right)^{1/4} \cdot \exp\left(-\frac{\omega_i Q_i^2}{2\hbar}\right) \cdot H_v\left(\sqrt{\frac{\omega_i}{\hbar}}Q_i\right) \tag{1.54}$$

1.2.2.1 The m-dimensional harmonic oscillator

The m-dimensional harmonic oscillator Hamiltonian can be written as the sum of m one-dimensional harmonic oscillator Hamiltonians:

$$\hat{H} = \sum_{i=1}^{m} \hat{H}_i \tag{1.55}$$

where

$$\hat{H}_i = -\frac{\hbar^2}{2}\frac{\partial^2}{\partial Q_i^2} + \frac{1}{2}\omega^2 Q_i^2. \tag{1.56}$$

Let $\chi_{v_i}(Q_i)$ be a solution to the one-dimensional time-independent Schrödiger equation,

$$\hat{H}_i \chi_{v_i}(Q_i) = E_{v_i} \chi_{v_i}(Q_i). \tag{1.57}$$

The set of eigenfunctions of the m-dimensional harmonic oscillator Hamiltonian (equation 1.55) is the direct product of the eigenfunctions of each one-dimensional harmonic oscillator Hamiltonian (equation 1.56):

$$\chi_{\vec{v}}(\mathbf{q}) = \chi_{v_1}(Q_1) \cdot \chi_{v_2}(Q_2) \cdots \chi_{v_m}(Q_m), \tag{1.58}$$

where $\vec{v}$ is an m-dimensional vector of integers ($v_i \in \{0, 1, 2, \dots\}$) that specify the quantum number associated with each 1-D harmonic oscillator wave function, $\chi_{v_i}(q_i)$. To see this, let the m-dimensional harmonic oscillator Hamiltonian operate on the function in equation 1.58:

$$\begin{aligned}
\hat{H}\chi_{\vec{v}}(\mathbf{Q}) &= \hat{H}\chi_{v_1}(Q_1) \cdots \chi_{v_m}(Q_m) && (1.59)\\
&= \left[\sum_{i=1}^{m} \hat{H}_i\right] \chi_{v_1}(Q_1) \cdots \chi_{v_m}(Q_m) && (1.60)\\
&= \sum_{i=1}^{m} \hat{H}_i \chi_{v_1}(Q_1) \cdots \chi_{v_m}(Q_m) && (1.61)\\
&= \sum_{i=1}^{m} E_{v_i} \chi_{v_1}(Q_1) \cdots \chi_{v_m}(Q_m) && (1.62)
\end{aligned}$$

since $\hat{H}_i \chi_{v_i}(Q_i) = E_{v_i} \chi_{v_i}(Q_i)$,

$$= \left(\sum_{i=1}^{m} E_{v_i}\right) \chi_{\vec{v}}(\mathbf{Q}). \tag{1.63}$$

So, the functions described in equation 1.58 (products of 1-D harmonic oscillator functions) are in fact eigenfunctions of the m-dimensional harmonic oscillator

Hamiltonian, with energies

$$E_{\vec{v}} = E_{v_1} + E_{v_2} + \cdots + E_{v_m} \tag{1.64}$$

1.2.3 Franck-Condon principle

Now that we've solved for the vibronic energies and wave functions using the Born-Oppenheimer approximation under the harmonic approximation, we need to compute the transition intensities associated with each possible vibronic transition

$$I = P^2 = [\langle \Psi' | \, \boldsymbol{\mu} \, | \Psi'' \rangle]^2 = \left[\int \Psi'^* \boldsymbol{\mu} \Psi'' \, \mathrm{d}\tau \right]^2 . \tag{1.65}$$

From the Born-Oppenheimer approximation, the wave functions $\Psi'(\mathbf{r}, \mathbf{Q}')$ and $\Psi''(\mathbf{r}, \mathbf{Q}'')$ can be written as a product of electronic and nuclear wave functions:

$$\Psi'(\mathbf{r}, \mathbf{Q}') = \psi'(\mathbf{r}; \mathbf{Q}') \Omega'_k(\mathbf{Q}') \tag{1.66}$$

and

$$\Psi''(\mathbf{r}, \mathbf{Q}'') = \psi''(\mathbf{r}; \mathbf{Q}'') \Omega''_l(\mathbf{Q}''). \tag{1.67}$$

Substituting these in to the equation for P, we get

$$\begin{aligned}
P &= \langle \psi' \Omega'_k | \, \boldsymbol{\mu} \, | \psi'' \Omega''_l \rangle && (1.68)\\
&= \int \psi'^* \Omega'^*_k (\boldsymbol{\mu}_e + \boldsymbol{\mu}_n) \psi'' \Omega''_l \, \mathrm{d}\tau && (1.69)\\
&= \int \psi'^* \Omega'^*_k \boldsymbol{\mu}_e \psi'' \Omega''_l \, \mathrm{d}\tau + \int \psi'^* \Omega'^*_k \boldsymbol{\mu}_n \psi'' \Omega''_l \, \mathrm{d}\tau && (1.70)\\
&= \int \psi'^* \Omega'^*_k \boldsymbol{\mu}_e \psi'' \Omega''_l \, \mathrm{d}\tau + \underbrace{\int \psi'^* \psi'' \, \mathrm{d}\tau_e}_{0} \cdot \int \Omega'^*_k \boldsymbol{\mu}_n \Omega''_l \, \mathrm{d}\tau_n && (1.71)
\end{aligned}$$

due to the orthogonality of the electronic wave functions,

$$= \int \psi'^{*}\Omega_k'^{*}\boldsymbol{\mu}_e\psi''\Omega_l'' \,\mathrm{d}\tau. \tag{1.72}$$

To simplify this equation even further, we make an *approximate* separation of integrals to produce

$$P = \underbrace{\int \psi'^{*}\boldsymbol{\mu}_e\psi'' \,\mathrm{d}\tau_e}_{\text{transition moment}} \cdot \underbrace{\int \Omega_k'^{*}\Omega_l'' \,\mathrm{d}\tau_n}_{\text{Franck-Condon overlap}} . \tag{1.73}$$

This factorization would be exact if the integral $\int \psi'^{*}\boldsymbol{\mu}_e\psi'' \,\mathrm{d}\tau_e$ over the spatial coordinates of the electrons (called a transition dipole surface) did not depend on the nuclear coordinates. However, in the Born-Oppenheimer approximation, $\psi'(\mathbf{r};\mathbf{Q}')$ and $\psi''(\mathbf{r};\mathbf{Q}'')$ *do* depend (parametrically) on the nuclear coordinates. Since the dependence is usually rather smooth, though, it is neglected (this is called the *Condon approximation*), and the transition dipole surface is factored out of the nuclear integral.

When investigating a single electronic transition, the transition moment is not important to calculate (it is simply a scaling factor). The Franck-Condon overlaps are the essential calculations and provide the transition intensities for the possible vibronic transitions between the electronic states $\psi'(\mathbf{r};\mathbf{Q}')$ and $\psi''(\mathbf{r};\mathbf{Q}'')$.

1.3 Calculating Franck-Condon factors

So far, we've established that Franck-Condon (FC) overlaps are integrals over the nuclear coordinates whose integrands are products of two vibrational wave functions with different coordinate origins; one from an initial electronic state and

one from an excited electronic state, and that the square of an FC overlap (a Franck-Condon factor, or FCF) represents the intensity of the corresponding vibronic transition, under the Born-Oppenheimer approximation. The calculation of FC overlaps is then crucial to understanding the electronic spectroscopy of molecular systems.

Many approaches have been suggested for the efficient computation of FC overlaps under the assumption of harmonicity [11, 33, 59, 28, 92]. Gruner and Brumer [39] discussed the computational aspects of some of these methods and proposed the use of recurrence equations developed by Doktorov, Malkin, and Manko [28] for the computation of FC overlaps. Today, the Doktorov recurrence equations are widely considered to be the most computationally efficient approach for computing FC overlaps and have been implemented in many popular programs for FC calculations [22, 73, 6].

Even within the subset of Franck-Condon algorithms that utilize the Doktorov recurrence equations, though, there is still debate over the preferred computer implementation of the equations. Various algorithms have been proposed [39, 86, 42], each with its own advantages and disadvantages. Research towards the improvement in efficiency of Franck-Condon factor calculations persists because the molecules that we are interested in studying are growing larger and more complex than ever before. In 2005, Dierksen and Grimme [23] performed a Franck-Condon calculation for the first photoelectron transition of a large PAH derivative with 468 normal modes of vibration. Their calculation required 22 hours to complete. In 2007, Jankowiak, Stuber, and Berger [48] demonstrated their improvements to existing FC algorithms by performing an FC calculation for the same molecule in

roughly 70 minutes, making the study of the PAH derivative much easier than before. This example illustrates that Frank-Condon calculations for large molecules are not computationally cheap, and that research towards to the improvement of FC calculation algorithms is important to facilitate the study of larger molecules than is currently possible.

The purpose of the remainder of this chapter is to present a new algorithm for the computation of FC overlaps using the Doktorov recurrence equations, facilitated by a novel data structure for storing FC overlaps; the method outlined here achieves significant computational savings with respect to other algorithms that are based on the recurrence equation framework [80].

The structure of the remainder of this chapter is as follows: In Section 1.3.1, the recursion equations developed by Doktorov *et al.* for the computation of FC overlaps will be reviewed. In Section 1.3.2, a new algorithm for using the Doktorov equations to compute FC overlaps will be described in detail. In Section 1.4, timing results of my implementation of the proposed algorithm for several different systems will be shown.

1.3.1 Recurrence equations for Franck-Condon calculations

Consider a molecule with m modes of vibration. Under the harmonic approximation, the vibrational eigenstates of a particular electronic state take the form

$$|\vec{v}\rangle = \prod_{i=1}^{m} \frac{1}{\sqrt{2^{v_i} v_i!}} \cdot \left(\frac{\omega_i}{\pi\hbar}\right)^{\frac{1}{4}} \cdot \exp\left(-\frac{\omega_i Q_i^2}{2\hbar}\right) \cdot H_{v_i}\left(\sqrt{\frac{\omega_i}{\hbar}} Q_i\right), \tag{1.74}$$

where $\vec{v} = (v_1, v_2, \ldots, v_m)$ is a set of quantum numbers that defines the eigenstate, q_i are *dimensionless normal coordinates* defined by the potential energy surface of the electronic state, and $H_{v_i}(x)$ are the physicists' Hermite polynomials. Following standard convention, variables referring to the initial electronic state of a transition will be doubly primed (e.g. Q_i''), and those referring to the final electronic state will be singly primed (e.g. Q_i').

The Franck-Condon overlaps associated with a given electronic transition then take the form of integrals

$$\langle \vec{v}\,' | \vec{v}\,'' \rangle = \int \left[\prod_{i=1}^{m} \frac{1}{\sqrt{2^{v_i'} v_i'!}} \cdot \left(\frac{\omega_i'}{\pi\hbar} \right)^{\frac{1}{4}} \cdot \exp\left(-\frac{\omega_i' Q_i'^2}{2\hbar} \right) \cdot H_{v_i'} \left(\sqrt{\frac{\omega_i'}{\hbar}} Q_i' \right) \right] \cdot \tag{1.75}$$

$$\left[\prod_{i=1}^{m} \frac{1}{\sqrt{2^{v_i''} v_i''!}} \cdot \left(\frac{\omega_i''}{\pi\hbar} \right)^{\frac{1}{4}} \cdot \exp\left(-\frac{\omega_i'' Q_i''^2}{2\hbar} \right) \cdot H_{v_i''} \left(\sqrt{\frac{\omega_i''}{\hbar}} Q_i'' \right) \right] \mathrm{d}\tau_n \tag{1.76}$$

for all combinations of $\vec{v}\,'$ and $\vec{v}\,''$. Since the two functions inside the integral are functions of two different coordinate systems, it is important to understand the relationship between the two sets of coordinates.

The normal coordinates $\mathbf{Q}$ of the two electronic states are related to each other by the linear transformation [31]

$$\mathbf{Q}' = \mathbf{S}\mathbf{Q}'' + \mathbf{d}, \tag{1.77}$$

where $\mathbf{Q}''$ and $\mathbf{Q}'$ are vectors containing normal coordinates in the initial and final electronic states, respectively, $\mathbf{S}$ is a Duschinsky matrix that relates the two sets of normal coordinates, and $\mathbf{d}$ is the equilibrium displacement vector in the space of the

final state normal coordinates. The Duschinsky matrix **S** is computed by

$$\mathbf{S} = (\mathbf{L}')^T \mathbf{L}'' \tag{1.78}$$

where **L** is a $3N \times m$ matrix whose columns are the m columns of **A** (equation 1.35) corresponding to the vibrational normal modes. The displacement vector **d** is computed by

$$\mathbf{d} = (\mathbf{L}')^T \mathbf{r}_\delta \tag{1.79}$$

where $\mathbf{r}_\delta = \mathbf{r}'' - \mathbf{r}'$ is the difference (in mass-weighted Cartesian coordinates) between the equilibrium geometries of the two electronic states.

In their paper [28], Doktorov *et al.* developed recurrence equations to compute Franck-Condon overlaps (equation 1.76) given the Duschinsky matrix **S**, displacement vector **d**, and normal mode frequencies ω_i'' and ω_i' for both the initial and final electronic states, respectively. Their recurrence equations are widely considered to be the most computationally efficient method for computing a large number of FC overlaps (as is often needed for understanding vibronic spectra). Let us define the following matrices $(\boldsymbol{\lambda}'', \boldsymbol{\lambda}', \mathbf{J}, \mathbf{Q}, \mathbf{P}, \mathbf{R})$ and vector (δ) in accordance with the notation introduced by Doktorov *et al.* (note that here **Q** is a matrix and not a vector of normal coordinates):

$$\boldsymbol{\lambda}'' = \text{diag}(\omega_1''^{\frac{1}{2}}, \omega_2''^{\frac{1}{2}}, \ldots, \omega_m''^{\frac{1}{2}}), \qquad \boldsymbol{\lambda}' = \text{diag}(\omega_1'^{\frac{1}{2}}, \omega_2'^{\frac{1}{2}}, \ldots, \omega_m'^{\frac{1}{2}}) \tag{1.80}$$

$$\mathbf{J} = \boldsymbol{\lambda}'' \mathbf{S} \boldsymbol{\lambda}'^{-1}, \qquad \mathbf{Q} = (\mathbf{I} + \mathbf{J}^T \mathbf{J})^{-1}, \qquad \mathbf{P} = \mathbf{J}\mathbf{Q}\mathbf{J}^T, \tag{1.81}$$

$$\mathbf{R} = \mathbf{Q}\mathbf{J}^T, \quad \text{and} \quad \delta = \hbar^{1/2} \boldsymbol{\lambda}' \mathbf{d}. \tag{1.82}$$

Using these definitions, Doktorov *et al.* show that the overlap $\langle \vec{0}\,' | \vec{0}\,'' \rangle$ (where $\vec{0} = (0, 0, \ldots, 0)$ is the ground vibrational wave function) can be computed by

$$\langle \vec{0}\,' | \vec{0}\,'' \rangle = 2^{m/2} \left(\prod_{j=1}^{m} (\omega'_j / \omega''_j)^{1/4} \right) (\det \mathbf{Q})^{1/2} \exp[-\frac{1}{2} \delta^T (\mathbf{I} - \mathbf{P}) \delta] \tag{1.83}$$

Overlaps involving excited vibrational levels, of either the ground or upper electronic states, can then be computed using the recurrence relations

$$\begin{aligned} &\langle \vec{v}\,' | v''_1, \ldots, v''_i + 1, \ldots, v''_m \rangle \\ &\quad = 2 \sum_{k=1}^{m} (\mathbf{R})_{ik} \left(\frac{v'_k}{v''_i + 1} \right)^{1/2} \langle v'_1, \ldots, v'_k - 1, \ldots, v'_m | v''_1, \ldots, v''_i, \ldots, v''_m \rangle \\ &\quad + \sum_{j=1}^{m} (2\mathbf{Q} - \mathbf{I})_{ij} \left(\frac{v''_j}{v''_i + 1} \right)^{1/2} \langle \vec{v}\,' | v''_1, \ldots, v''_j - 1, \ldots, v''_m \rangle \\ &\quad - (\mathbf{R}\delta)_i \left(\frac{2}{v''_i + 1} \right)^{1/2} \langle \vec{v}\,' | v''_1, \ldots, v''_i, \ldots, v''_m \rangle \end{aligned} \tag{1.84}$$

and

$$\begin{aligned} &\langle v'_1, \ldots, v'_k + 1, \ldots, v'_m | \vec{v}\,'' \rangle \\ &\quad = 2 \sum_{i=1}^{m} (\mathbf{R})_{ki} \left(\frac{v''_i}{v'_k + 1} \right)^{1/2} \langle v'_1, \ldots, v'_k, \ldots, v'_m | v''_1, \ldots, v''_i - 1, \ldots, v''_m \rangle \\ &\quad + \sum_{j=1}^{m} (2\mathbf{P} - \mathbf{I})_{kj} \left(\frac{v'_j}{v'_k + 1} \right)^{1/2} \langle v'_1, \ldots, v'_j - 1, \ldots, v'_m | \vec{v}\,'' \rangle \\ &\quad - ((\mathbf{I} - \mathbf{P})\delta)_k \left(\frac{2}{v'_k + 1} \right)^{1/2} \langle v'_1, \ldots, v'_k, \ldots, v'_m | \vec{v}\,'' \rangle \,. \end{aligned} \tag{1.85}$$

If we assume that the vibrational wave function in the initial electronic state is $|\vec{0}\,''\rangle$ (as is true at 0 K temperatures), then the Franck-Condon overlaps with excited vibrational states in the final electronic state can be computed with the simpler

recurrence

$$\begin{aligned}
&\langle v'_1, \ldots, v'_k + 1, \ldots, v'_m | \vec{0}\,'' \rangle \\
&\quad = \sum_{j=1}^{m} (2\mathbf{P} - \mathbf{I})_{kj} \left(\frac{v'_j}{v'_k + 1} \right)^{1/2} \langle v'_1, \ldots, v'_j - 1, \ldots, v'_m | \vec{0}\,'' \rangle \\
&\quad - ((\mathbf{I} - \mathbf{P})\delta)_k \left(\frac{2}{v'_k + 1} \right)^{1/2} \langle v'_1, \ldots, v'_k, \ldots, v'_m | \vec{0}\,'' \rangle \,. \qquad (1.86)
\end{aligned}$$

The majority of Section 1.3.2 will focus on the case where $|\vec{v}\,''\rangle$ is fixed at $|\vec{0}\,''\rangle$ and, as a result, equation 1.86 will be the primary referenced equation. However, in Section 1.3.6.3, the case of an excited initial vibrational state will be introduced and equations 1.84 and 1.85 will become relevant.

1.3.2 Implementation of the Doktorov recurrence equations

The computer implementation of the Doktorov recurrence equations has been the subject of much research over the past thirty years. For any implementation of the recurrence equations, an important consideration is whether or not FC overlaps should be stored in memory after they are computed. Note that, using equations 1.84-1.86, computing any FC overlap requires the values of other FC overlaps. When computing many FC overlaps at once, in some prescribed order, it is likely that computing an FC overlap will require the values of other overlaps that have already been computed. If previously computed overlaps are stored in memory, they can be recovered; otherwise, they must be recomputed. This consideration led Berger *et al.* [6] to classify the possible recursive FC algorithms into three groups:

1. Algorithms that store all computed FC overlaps in memory (conventional methods).

2. Algorithms that store some of the computed FC overlaps in memory (semi-direct methods).

3. Algorithms that avoid the storage of FC overlaps and recompute all required overlaps using the recurrence equations (direct methods).

Conventional methods are often the fastest, but require the most memory. Alternatively, semi-direct and direct methods save on memory usage, but perform redundant calculations that increase the total run time of an algorithm.

For conventional methods, Gruner and Brumer [39] found that the major computational problems involved are indexing and searching for overlap integrals in memory. These problems also affect semi-direct methods, as these methods also store at least some of the computed overlaps in memory. Consequently, the data structure used to store FC overlaps can play an important role in the computational efficiency of the overall algorithm. It is important that FC overlaps are stored in such a way that they can be recovered quickly and efficiently. To this end, several data structures / algorithms have been proposed:

1. Gruner and Brumer [39] proposed the use of a binary tree data structure to store Franck-Condon overlaps. Each node of a binary tree has two children nodes (called left and right, here). An illustration of their proposed data structure can be found in Figure 1.3. The primary benefit of using a binary tree

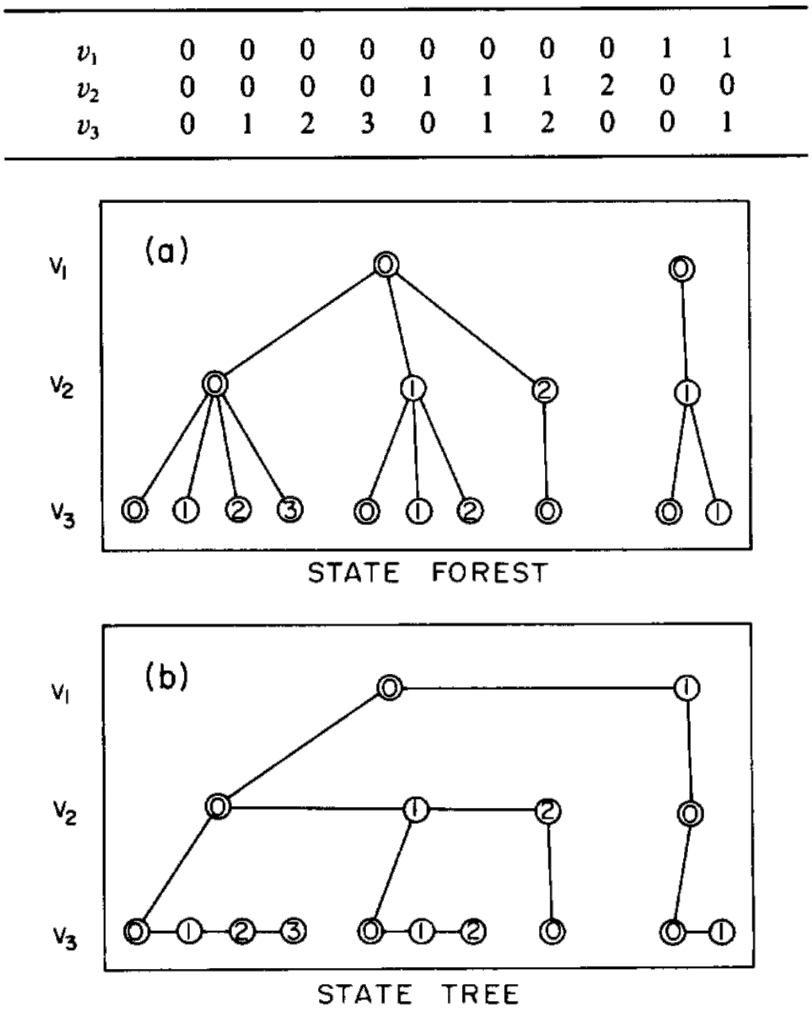


Figure 1.3: Image and caption taken from Gruner *et al.* [39]: (a) A forest representation of the states in table at the top of the figure. Here each vertically connected set of nodes forms a triplet of quanta. (b) A binary tree representation of the same states as in (a). The path through the tree is described in the text. Note that each node has at most two descendant nodes, considerably simplifying passage along the tree.

Algorithm 1 Binary tree overlap retrieval

```
right v_1 times
for i = 2:m do
    left once
    right v_i times
end for
end
```

structure is that it provides a simple, efficient algorithm for retrieving overlaps that have been stored in the tree: To retrieve an overlap associated with a particular final vibrational state, $\langle v'_1 v'_2 \dots v'_m|$, begin at the root of the tree and follow Algorithm 1. The binary tree structure (or slight modifications thereof) has been utilized in several proposed algorithms [42, 23, 86] and is implemented in the program FCfast [22, 23].

2. Ruhoff and Ratner [86] discussed the idea of storing FC overlaps in a two-dimensional array, where one dimension is indexed by vibrational states in the initial electronic state, and the other is indexed by vibrational states in the final electronic state. An illustration of their indexing scheme for a single dimension is shown in Figure 1.4.

   A mapping function (called POSITION by Ruhoff *et al.*) is used to calculate an array index p, for each dimension, using the quantum numbers ($\vec{v}$) associated with each vibrational state in the overlap. The POSITION function provided in [86] is shown in Algorithm 2.

   When cycling over the vibrational states for which FC overlaps are to be computed, Ruhoff *et al.* suggest iterating over the array indices sequentially

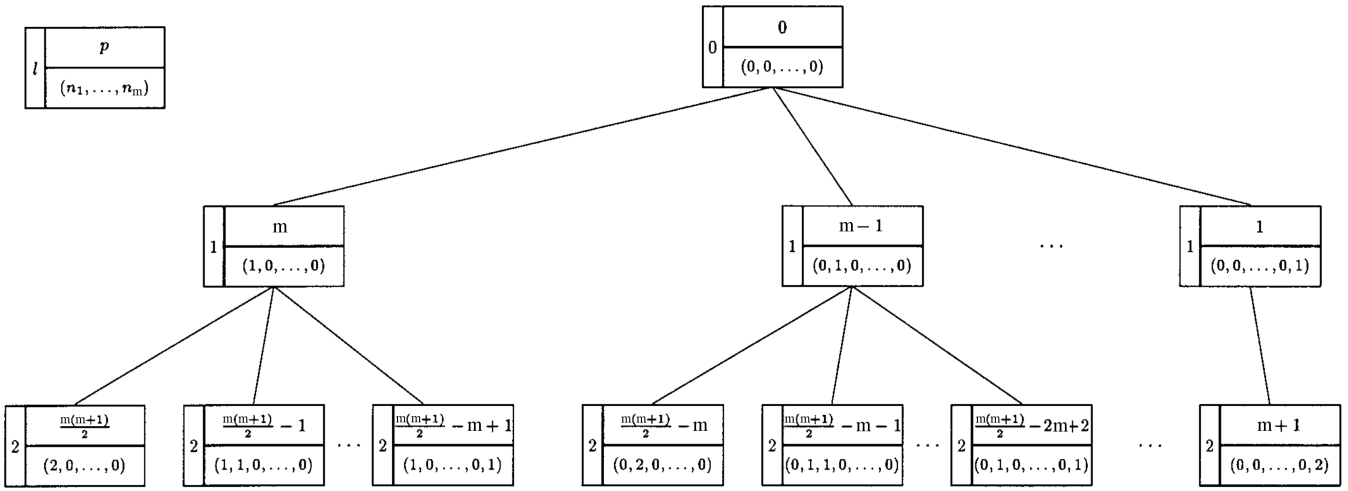


Figure 1.4: Image and caption taken from Ruhoff *et al.* [86]: Graphical representation of the mapping POSITION for the first three level $l = 0, 1, 2$. If we want to compute POSITION$(\vec{v})$, we search for in the tree and read the corresponding value of p. Likewise, if we will compute POSITION$^{-1}(p)$, we search for p and read . The upper-left box shows the labeling.

Algorithm 2 One-dimensional array overlap retrieval

```
1:  procedure POSITION(int m, int l, int v[])
2:      i = 1, p = N(m, l) − 1;
3:      while l > 0 do
4:          if v[i] > 0 then
5:              l = l − v[i];
6:          end if
7:          if l > 0 then
8:              p = p − S(m − i + 1, l − 1);
9:          end if
10:         i = i + 1
11:     end while
12:     return p
13: end procedure
```

where

$$N(m,l) = \sum_{i=0}^{l} S(m,i) \tag{1.87}$$

and

$$S(m,i) = \binom{m+i-1}{i}. \tag{1.88}$$

and using an algorithm they call NEXTSTATE to determine the quanta associated with each upcoming array index. The details of NEXTSTATE are not important for the purposes of this discussion, but it is important (for the understanding of Figure 1.6) to note that it is associated with the indexing of Franck-Condon overlaps.

A variant of the two-dimensional array structure and the algorithmic suggestions of Ruhoff *et al.* [86] have been implemented in the program ezSpectrum [73].

No matter what data structure is used, many existing algorithms for computing FC overlaps (such as the ones used in FCfast and ezSpectrum) tend to follow the same general structure : Cycle over the vibrational states of the desired FC overlaps and, for each desired overlap,

1. Determine the identity of the other overlaps that are needed to compute it (via equations 1.84-1.86).

2. Use a general routine (determined by the data structure) to search for each of these overlaps in memory.

3. Use the recovered overlaps to compute the new FC overlap.

Appropriately, algorithms that follow the above framework will be referred to as "searching algorithms" for the remainder of this section, since, for each overlap to be computed, they must search through a given data structure for overlap dependencies. The problem with algorithms of this type is that they spend the majority of their run time searching for overlaps stored in memory. Figures 1.5 and 1.6 show GNU gprof [38] profiling data for FC overlap calculations performed by FCfast and ezSpectrum, respectively. Note that the functions related to searching data structures for overlaps take up at least 75% of the total run time of each of the FC overlap calculation algorithms.

The algorithm proposed in this section, facilitated by a data structure proposed here, avoids the inefficiency of searching for overlaps by tracking, throughout the algorithm, the locations in memory of overlaps needed to calculate new overlaps

% time	self seconds	calls	name	
70.13	267.09	1512271258	mp_get_integral_	(Algorithm 1)
16.58	63.13	181031721	mp_calc_integral_	
5.45	20.75	181031720	mp_insert_integral_	(Algorithm 1)

Figure 1.5: GNU gprof profiling data for an FC overlap calculation for the molecule oxirane using FCfast. Both of the functions get_integral and insert_integral use the general algorithm outlined in Algorithm 1.

% time	self seconds	calls	name	
58.90	220.33	2455444693	convVibrState2Index()	(POSITION)
17.97	67.23	314457502	enumerateVibrStates()	(NEXTSTATE)
13.86	51.85	314457494	Dushinsky::evalSingleFCF()	

Figure 1.6: GNU gprof profiling data for an FC overlaps calculations for the molecule adenine using ezSpectrum. The function convVibrState2Index() is derived from Algorithm 2 and enumerateVibrStates() is derived from the NEXTSTATE function described in [86].

via equations 1.84-1.86. Since the proposed algorithm improves upon the recovering of overlaps from memory, it is only appropriate to discuss in the context of approaches that store overlaps in memory (conventional and semi-direct approaches); direct methods, which do not store any overlaps, will not be discussed. The rest of this section will be devoted to describing the proposed data structure and algorithm for FC overlap computation.

1.3.3 Proposed general tree structure

The FC overlap storage scheme proposed in this paper can be considered an extension of the binary tree advocated by Gruner and Brumer. Instead of a *binary* tree, though, a *general* tree structure is used, whose nodes can have more than two children. The general tree structure, like the binary tree, organizes vibrational states within a single electronic state, so the following description of the structure assumes that the vibrational state in the initial electronic state is fixed (i.e. the proposed tree can store the overlaps $\langle \vec{v}' | \vec{v}'' \rangle$ for all $\vec{v}'$, given a fixed $\vec{v}''$). An extension of the structure to account for multiple initial states (as is needed for finite temperature calculations) has been implemented as well and exists in the form of a nested tree structure that will be introduced in Section 1.3.6.3.

In the proposed general tree structure, each final vibrational state $\langle \vec{v}' |$ has a node associated with it. Each node contains two pieces of information: the Franck-Condon overlap $\langle \vec{v}' | \vec{v}'' \rangle$ and a memory pointer to an array of other nodes (called the $\mathrm{children}$ of a node). The $\mathrm{children}$ of a given node are determined by increasing a single quantum number v'_k of the node's vibrational state $\langle \vec{v}' |$ by one. Each node,

however, will *not* have m children (where m is the number of vibrational modes). If this were the case, there would exist two distinct nodes that correspond to the same vibrational state. Instead, each node has a maximum number of children determined by how its set of quanta differs from that of its parent. Consider a node that was created by increasing the i^{th} quantum number (v'_i) of its parent node by one; in my proposed tree structure, I allow this node to have a maximum of i children, created by increasing its j^{th} quantum number (v'_j) by one, where j is less than or equal to i.

Figure 1.7 shows the proposed tree structure (how children are assigned to any given node). Note that node $\langle 010|$ is created by increasing the 2^{nd} quantum number of $\langle 000|$ by one. As a result, node $\langle 010|$ can have a maximum of 2 children, one created by increasing its 1^{st} quantum number by one ($\langle 110|$) and one created by increasing its 2^{nd} quantum number by one ($\langle 020|$). Similarly, node $\langle 100|$ is created by increasing the 1^{st} quantum number of $\langle 000|$ by one, so it can have a maximum of 1 child, created by increasing *its* 1^{st} quantum number by one ($\langle 200|$).

1.3.3.1 Implementing the general tree structure

Implementing the proposed tree structure in a way that avoids unnecessary memory usage is a non-trivial task. In my implementation, each node of the tree (which corresponds to a target vibrational state $\langle \vec{v}\,'|$) is a structure called a Tree that contains a double-precision floating point variable (containing the FC overlap $\langle \vec{v}\,'|\vec{0}\,''\rangle$), and a memory address pointing to an array of Tree objects:

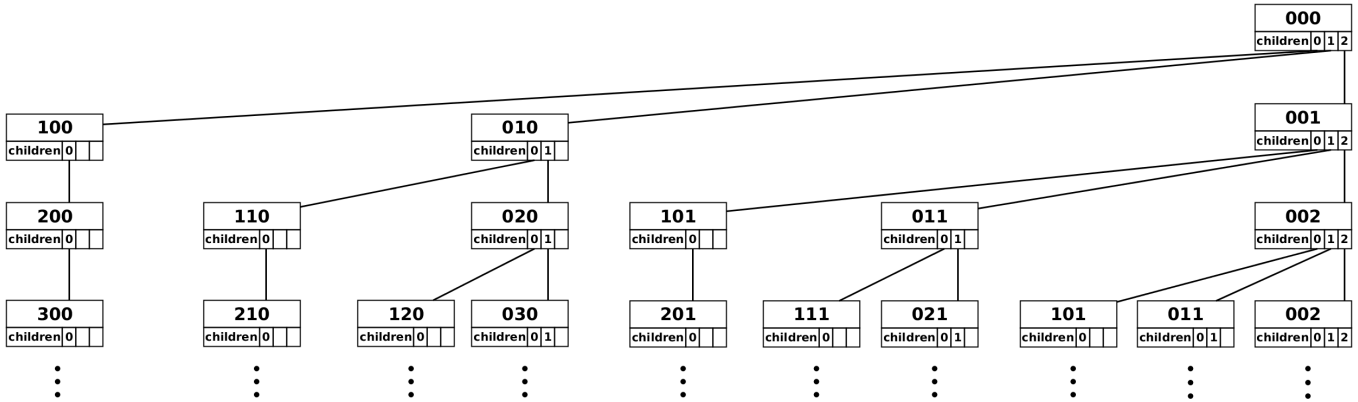


Figure 1.7: Representation of the proposed tree structure for storing FC overlaps for a system with three vibrational modes. Each node has a set of quanta associated with it (that is not actually stored in memory), as well as a pointer to an array of other tree nodes (its children).

```
struct Tree {
  double overlap;
  Tree* children;
}
```

Before the computation of FC overlaps occurs, an array of tree nodes of size equal to a maximum tree size specified by the user is created (assume for now that this array is large enough to store all required FC overlaps; the alternative case is discussed in Section 1.3.6.2). The general tree structure, shown in Figure 1.7, is then created using a "self-referencing" approach. The term "self-referencing" is used because the children pointer of each node in the array of nodes points to some other location in the same array of nodes. By creating the tree in this way, only 16 bytes of memory are required for each FC overlap that will be calculated (one double-precision floating-point number and one memory address). It is important to note that the storage requirements of the general tree structure are independent of the number of modes in a system; they only depend on the number of desired FC overlaps. A description of the array of nodes that describes the tree in Figure 1.7 can be found in Figure 1.8.

1.3.4 Computing Franck-Condon overlaps using the general tree structure

The proposed general tree data structure for storing FC overlaps facilitates a new algorithm for computing FC overlaps that eliminates the need to search the data structure for each use of the recurrence equations. The algorithm described in this section assumes that the initial state is fixed at $|\vec{0}''\rangle$, so equation 1.86 will be used for the recurrence calculations. Some terminology that will be used in the

State	000	100	010	001	200	300	110	020	210
Array Index	0	1	2	3	4	5	6	7	8
Children	1	4	6	11	5	-	8	9	-
State	120	030	101	011	002	201	111	021	211
Array Index	9	10	11	12	13	14	15	16	17
Children	-	-	14	15	-	-	17	-	-

Figure 1.8: An explanation of the implementation of the general tree structure found in Figure 1.7. For each node (State), the table shows the location of the node in the linear array of nodes (Array Index) and the location in the linear array to which the children pointer of the node points (Children).

description of the proposed algorithm will now be introduced.

When computing a given FC overlap using the recurrence relation in equation 1.86, the tree node associated with the bra state (final state) in the last overlap on the right hand side of the equation,

$$\langle v'_1, \ldots, v'_k, \ldots, v'_m| \,, \tag{1.89}$$

will be called the parent node. The node associated with the bra state on the left hand side of the equation,

$$\langle v'_1, \ldots, v'_k + 1, \ldots, v'_m| \,, \tag{1.90}$$

will be called a child of the parent node, since the associated vibrational state is obtained by increasing a single quantum number (v'_k) of the parent node by one. Finally, nodes associated with the bra states in the summation term in the equation,

$$\langle v'_1, \ldots, v'_j - 1, \ldots, v'_m| \;\forall\; j, \tag{1.91}$$

will be called the grandnodes of the child node (since they exist two levels above the child, similar to the distance between a child and its grandparents' generation in a family tree). Note that any child can have up to m grandnodes (i.e. not just four, as "grand" here refers to a generation without regard to lineage) associated with it.

The following algorithm computes all FC overlaps corresponding to vibrational states whose energies

$$E_{\vec{v}'} = \sum_{j=1}^{m} v'_j \cdot \omega'_j, \tag{1.92}$$

are less than some specified energy threshold, E_{max}. Extensions of this approach to account for other methods of truncating the total number of computed FC overlaps are possible (and will be discussed in Section 1.3.6.1), and details regarding the implementation of these extensions can be found in supporting information. Before the algorithm begins, we allocate an array of Tree objects, called node_array, of size equal to a maximum tree size specified by the user (assume for now that this array is large enough to store all required FC overlaps). Next, we compute the FC overlap $\langle \vec{0}' | \vec{0}'' \rangle$ (using equation 1.83) and store the overlap in node_array[0]. Algorithm 3 is then followed.

Each call of Compute_FC_Factors() begins by setting the children array of the current parent node to a location in the node_array array determined by global_children_index. The value of global_children_index is then updated for the upcoming parent node (see Figure 1.8 for an example of how the children array of each node begins at a specific index of node_array). After updating global_children_array, the algorithm loops over all possible children of the cur-

Algorithm 3 The proposed recursive FC algorithm

```
1: call Compute_FC_Factors(node_array[0], {0,0,...,0}, m − 1,
                                        0.0, 1, NULL, 0)
2: procedure COMPUTE_FC_FACTORS(Tree parent, int quanta[],
                                int incremented_mode, double current_energy,
                                int global_children_index,
                                Tree* grandnodes[], int num_grandnodes)
3:     (parent → children) = (node_array + global_children_index)
4:     Count the number of children parent will have based on the maximum
       energy criteria and increment global_children_index by the number
       of children
5:     child_index = 0
6:     start = 0
7:     for i ∈ {start, start+1, ..., incremented_mode} do
8:         double child_energy = current_energy +
                                        target_state_frequency[i]
9:         if child_energy > max_energy then
10:            return
11:        end if
12:        Compute FC overlap for children[child_index] of
           parent using equation 1.86
13:        Update grandnodes and num_grandnodes for current child,
           creating new_grandnodes (see Algorithm 4)
14:        quanta[i] ← quanta[i] + 1
15:        call Compute_FC_Factors(parent → children[child_index],
                                        quanta, i, child_energy,
                                        global_children_index,
                                        new_grandnodes, num_grandnodes)
16:        child_index ← child_index + 1
17:    end for
18:    quanta[incremented_mode] ← quanta[incremented_mode] − 1
19:    return
20: end procedure
```

rent parent node (the variable incremented_mode contains the mode whose quantum number was incremented to create the current parent node from *its* parent, which is the maximum number of children the current parent node can have). If the energy of the vibrational state corresponding to the current child is larger than the maximum allowed energy, then the function returns. Otherwise, the FC overlap for the current child is computed using equation 1.86 (which requires the overlaps of the parent node and the grandnodes).

The cornerstone of the proposed algorithm is the procedure that locates the grandnodes for each use of equation 1.86. Throughout the algorithm, the locations in memory of the grandnodes of the current child are tracked using an array of memory pointers (called grandnodes[] in the above algorithm). The tracking of the grandnodes is the main improvement that the proposed algorithm provides over existing searching algorithms and accounts for the majority of the difference in run time between the two algorithms (discussed in Section 1.4).

After the FC overlap for the current child has been computed, the array of pointers that point to the grandnodes of the current child is updated to point to grandnodes of the current child's children (this is done using a simple procedure described in the next section). The set of quantum numbers of the current parent is then updated to those of the current child, and the recursive function is called, setting the current child as the next parent.

Before the algorithm begins, it is important that the vibrational modes are ordered such that the normal mode frequencies in the final electronic state ω'_j are in ascending order. When this is the case, the algorithm just described uses the

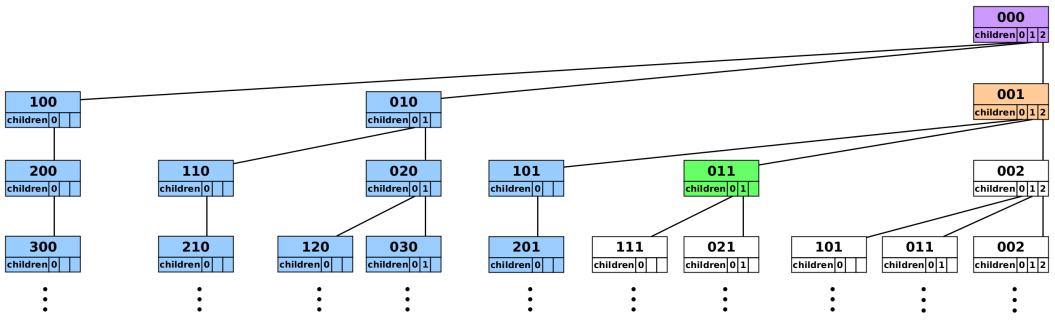
FC overlap calculation ordering suggested by Gruner and Brumer (first varying the quantum number corresponding to the lowest-frequency mode, then the second lowest, and so on, rejecting states that exceed the energy threshold). When iterating over vibrational states in this order, computing an FC overlap using equation 1.86 will never require FC overlaps that have not already been calculated and stored (when the algorithm is used conventionally). This property helps to simplify the algorithm by allowing us to ignore the possibility of attempting to retrieve from memory an FC overlap that has yet to be computed.

1.3.5 Tracking Franck-Condon overlap dependencies

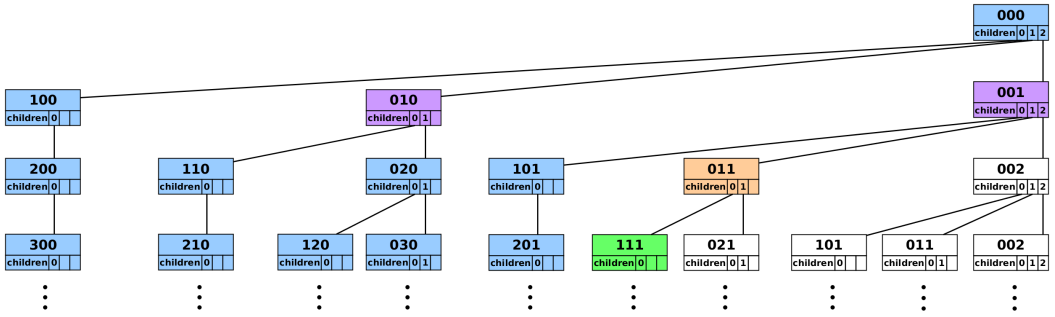
To compute the FC overlaps for each child of a given parent node, it is necessary to recover the FC overlaps associated with each grandnode. Finding where each of these FC overlaps are stored in memory can be very time-consuming and hinders the performance of the calculation. In the proposed algorithm, a method for tracking grandnodes is used that makes searching the tree for FC overlaps unnecessary and greatly improves the performance of calculating FC overlaps. In the following description, the phrase "child at index i" will be used to specify a certain child node of a given parent. The term "index" refers to the array index of the children array that each parent node contains (with indexing starting at 0).

Figure 1.9 illustrates how grandnodes spawn and update throughout the algorithm. In Step 1 of the figure, the current node (parent node) is at $\langle 001|$ (shown in orange). The current node is computing the FC overlap for its child at index 1 (state $\langle 011|$, shown in green). The grandnode required to compute $\langle 011|\vec{0}''\rangle$ is then

Step 1:



Step 2:



Step 3:

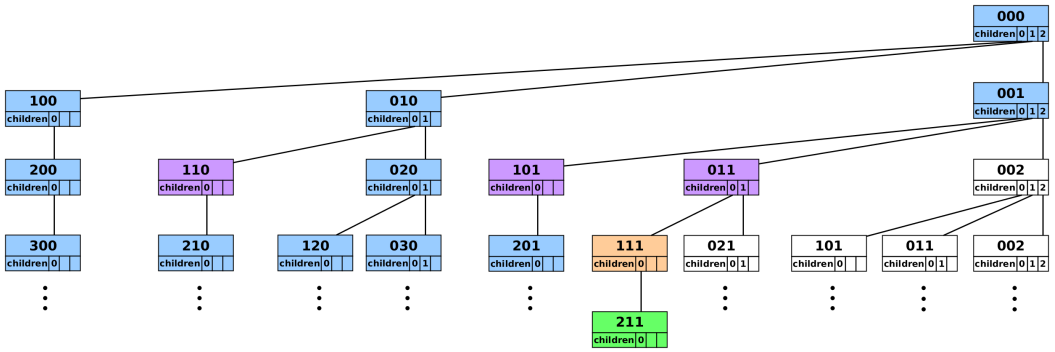


Figure 1.9: See Figure 1.10 for second part of Figure 1.9

$$\langle v'_1, \ldots, v'_k + 1, \ldots, v'_m | \vec{0}'' \rangle$$
$$= \sum_{j=1}^{m} (2\mathbf{P} - \mathbf{I})_{kj} \left(\frac{v'_j}{v'_k + 1} \right)^{1/2} \langle v'_1, \ldots, v'_j - 1, \ldots, v'_m | \vec{0}'' \rangle$$
$$- ((\mathbf{I} - \mathbf{P})\delta)_k \left(\frac{2}{v'_k + 1} \right)^{1/2} \langle v'_1, \ldots, v'_k, \ldots, v'_m | \vec{0}'' \rangle \quad \text{(1.86 revisited)}$$

Figure 1.10: Figure 1.9 is a step-by-step illustration of the FC overlap calculation algorithm. In each step, the overlaps that have already been computed are in blue, purple, and orange. The current child node being calculated is in green, its parent is in orange, and its grandnodes are in purple.

$\langle 000|$ (shown in purple). In Step 2, the current node has moved to $\langle 011|$ and the new child node is $\langle 111|$, with grandnodes $\langle 010|$ and $\langle 001|$. In Step 3, the current node has moved to $\langle 111|$ and the new child node is $\langle 211|$, with grandnodes $\langle 110|$, $\langle 101|$, and $\langle 001|$.

As is shown in Algorithm 3, throughout the algorithm, an array of memory pointers that point to the locations of the grandnodes is maintained. The pointer array is updated at each step of the algorithm using a simple procedure:

1. **When moving the current node to one of its children at index i, update all existing grandnodes by moving them to their children at index i.**

   From Step 1 to Step 2, the existing grandnode moves to its child at index 1 (since the current node $\langle 001|$ moves to its child of index 1). From Step 2 to Step 3, all grandnodes existing at Step 2 are moved to their children at index 0 (since the current node $\langle 011|$ moves to its child of index 0).

Algorithm 4 Procedure for updating grandnodes (replaces line 13 in Algorithm 3)

1: Allocate pointer array new_grandnodes of size num_grandnodes+1 to store grandnodes for the next level of recursion
2: **for** j $\in$ {0, 1, . . . , num_grandnodes-1} **do**
3: new_grandnodes[j] = (grandnodes[j]$\rightarrow$ children[child_index])
4: **end for**
5: **if** quanta[i] == 0 **then**
6: new_grandnodes[num_grandnodes] = parent
7: num_grandnodes $\leftarrow$ num_grandnodes + 1
8: **end if**

2. **When moving the current node from a vibrational state with zero quanta in a given mode, to a child with one quantum in the same mode, spawn a new grandnode at the current node.**

   From Step 1 to Step 2, the current node moves from $\langle 001|$ (which has zero quanta in mode 2), to $\langle 011|$ (which has one quantum in mode 2); thus, a grandnode is spawned at $\langle 001|$. Similarly, from Step 2 to Step 3, the current node moves from $\langle 011|$ (which has zero quanta in mode 1) to $\langle 111|$ (which has one quantum in mode 1); thus, a grandnode is spawned at $\langle 011|$.

With each increasing level of recursion in Algorithm 3 (corresponding to increasing depth of the tree structure), an array of memory pointers is allocated to store the grandnodes updated from the previous level of recursion. Right before the algorithm returns to the previous level of recursion, the pointer array can be freed, as it is no longer needed. Algorithm 4 summarizes the grandnode updating procedure just described. In the context of Algorithm 3, Algorithm 4 replaces line 13. With the above grandnode tracking procedure, searching trees and arrays for FC over-

laps becomes unnecessary and computing a large number of FC overlaps becomes vastly more efficient, as is demonstrated in Section 1.4.

1.3.6 Extensions of the basic algorithm

The above description of the proposed algorithm provides a general overview of the data structure used and the procedure for tracking grandnodes. As currently described, the approach is a conventional method that computes Franck-Condon overlaps at 0 K temperature and truncates the total number of FC overlaps using a maximum energy cutoff. To demonstrate a broader applicability of the algorithm, the following extensions will be described throughout this subsection: using methods other than a maximum energy to truncate the number of FC overlaps, using the algorithm in conjunction with a semi-direct approach, and using the algorithm in finite temperature calculations.

1.3.6.1 Other methods of truncating the total number of overlaps

Much research has been done in an attempt to reduce the total number of Franck-Condon factor calculations required to essentially capture the vibronic spectroscopy of large systems. One such approach involves using prescreening procedures, which truncate the number of FC overlap calculations by defining a maximum allowed quantum number for each individual vibrational mode (i.e. setting variables n_j such that

$$v_j \in \{0, 1, ..., n_j\} \tag{1.93}$$

for all j) and a maximum number of simultaneously excited modes (i.e. setting a variable s such that

$$\left[\sum_{j=1}^{m} \mathbf{1}_{\mathbb{Z}^+}(v_j)\right] < s, \tag{1.94}$$

where $\mathbf{1}_{\mathbb{Z}^+}(x)$ is an indicator function defined on the set of positive integers). These restrictions are put in place, in addition to restricting the energy of the target vibrational states, in order to avoid wasting computation time on FC overlaps that are of negligible magnitude. As shown in literature by Santoro *et al.* [87] and Jankowiak *et al.* [48], significant computational savings can be achieved by using these truncation methods in addition to a standard energy truncation. It is thus important that the computational approach proposed herein supports the use of these prescreening methods.

Algorithm 3 uses only a maximum energy to truncate the total number of FC overlaps, but the approach can be extended to support individual maximum quantum number restrictions and restrictions on the number of simultaneously excited modes. Support for each method of truncation is possible by selectively limiting the number of children that a given node can have in the general tree structure. In Algorithm 3, the set of quantum numbers associated with each parent node is tracked throughout, so it is possible to avoid creating certain children of a node if one of its quantum numbers is larger than a specified threshold, or if its number of simultaneously excited modes is larger than a specified threshold. For example, the pseudocode

```
1: if quanta[i] == maximum_allowed_quanta[i] then
2:     continue
```

3: **end if**

can be placed between lines 7 and 8 of Algorithm 3 to account for individual maximum quantum number restrictions. Additionally, line 4 of Algorithm 3 must also be modified to consider individual maximum quantum number restrictions when counting the number of children parent will have.

Adding support for restrictions on the maximum number of simultaneously excited modes is slightly more complicated. Notice that if the current parent node in Algorithm 3 has the maximum number of simultaneously excited modes, then only one of its children is allowed to exist (the child created by incrementing its quantum number in the mode indexed by incremented_mode). So, the pseudocode

1: **if** num_grandnodes == max_number_of_simultaneously_excited_modes **then**
2: start = incremented_mode
3: **end if**

can be placed between lines 6 and 7 of Algorithm 3 to account for maximum number of simultaneously excited modes restrictions. Note that the variable num_grandnodes, in addition to storing the number of grandnodes that the children of the current parent have, stores the number of simultaneously excited modes in the current parent node.

When adding support for maximum number of simultaneously excited mode restrictions, the grandnode updating procedure must also be modified, but this modification will not be discussed here. Interested readers are referred to Ap-

pendix A for a detailed description of the proposed algorithm with support for all truncation methods mentioned in this paper.

Support for the truncation procedures described in this section has be included in my implementation of the proposed algorithm. A demonstration of the proposed algorithm using prescreening truncation methods is shown in Section 1.4.3.

1.3.6.2 Creating a semi-direct variant

The grandnode tracking procedure described in Section 1.3.5 helps produce significant speedups in computation time over existing algorithms (as seen in Section 1.4), but it requires that all FC overlaps needed for a given FC overlap computation be stored in memory (in the general tree structure). This requirement, at first glance, appears to imply that the proposed algorithm is only of use when used conventionally (when *all* computed FC overlaps are stored in memory for the duration of the algorithm). Conventional methods, though, are often only of use with relatively small molecules. While such molecules comprise the majority of quantum chemical and spectroscopic studies in the Stanton research group, I recognize the need for a *generally useful* Franck-Condon program that is also applicable to larger systems. When working with large molecules, the space required to store all desired FC overlaps can quickly exceed the amount of memory available on a typical machine. For example, for a recent calculation of the FC profile for a polycyclic aromatic hydrocarbon derivative, Jankowiak *et al.* [48] computed almost 3 billion FC overlaps. Using the proposed tree storage structure, storing 3 billion overlaps would require 48 GB of memory (3 billion * 16 bytes), which, while not

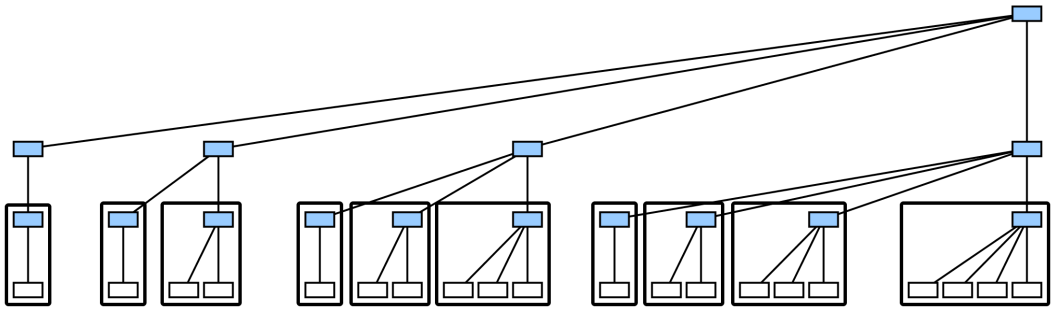


Figure 1.11: An example tree partitioning used in semi-direct variant of the proposed algorithm. In this example, there are four vibrational modes, and the tree is partitioned at the second generation.

impossibly large, exceeds the memory limits of typically available workstations. It is therefore important that the grandnode tracking procedure proposed in this paper be compatible with semi-direct approaches, in which the tree data structure is flushed when it reaches a user-specified maximum size and refilled with relevant integrals throughout the algorithm.

A semi-direct variant of the proposed algorithm can be created by partitioning the general tree data structure into individual branches. In the following description, the term "generation" will be used to describe a subset of tree nodes in which all nodes have the same sum of their quantum numbers. For example, referencing Figure 1.7, node 000 belongs to generation 0, nodes 100, 010, and 001 belong to generation 1, and so on.

The semi-direct algorithm begins by partitioning the tree into individual branches at some user-specified generation (or, by default, generation 1). An example of branch partitioning can be found in Figure 1.11. In Figure 1.11, the tree

is partitioned into branches at generation 2, creating 10 individual branches. All of the overlaps up to, and including, the partition generation are stored in memory indefinitely (these values are never flushed). The semi-direct algorithm then follows the algorithm described in Section 1.3.4, adding new nodes to the tree as they are computed. When the tree reaches a user-defined maximum size, the tree will flush all nodes belonging to generations greater than the partition generation and the algorithm will restart at the branch in which the algorithm existed before the flushing occurred.

When starting at a branch other than the first branch of the tree, the algorithm will have to recompute some of the previously computed overlaps to be used as grandnodes in future computations. This recomputation is done on a branch-by-branch basis. Throughout the algorithm, the computation of each branch begins by computing (if necessary) previously-computed (possibly flushed) branches that contain nodes that will be grandnodes of nodes in the current branch.

We can avoid unnecessary recomputation by noting that, when recomputing branches for grandnodes, often only a subset of the entire branch needs to be recomputed. This is because the branch usually contains a number of nodes that can *not* be grandnodes in future computations. For example, consider an FC calculation for a system with three vibrational modes that uses a maximum generation criteria of 4 to truncate the total number of FC overlaps to compute. Let the resulting FC overlap tree, shown in Figure 1.12, be partitioned at generation 1, creating three branches. If we assume that a flushing of the tree has just occurred, and the algorithm is restarting at the third branch, then recomputation of the first and sec-

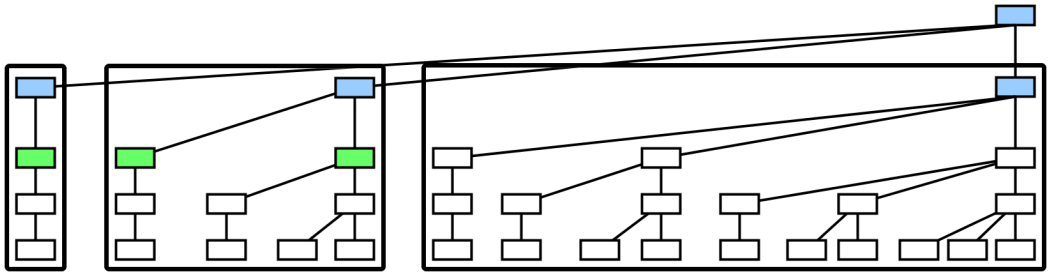


Figure 1.12: An example FC overlap tree for a three-mode system that restricts the number FC overlaps by setting a maximum generation (in this case, 4). The tree has been partitioned at generation 1 to create three branches. The nodes in green represent nodes that must be recomputed in the first and second branches when starting the algorithm in the third branch after a flush.

ond branches to compute grandnodes of nodes in the third branch can use a more restrictive maximum generation criteria of 2 (2 less than the original value of 4). Any overlaps in the first or second branches of generation higher than 2 will not be used as grandnodes while computing the third branch and do not have to be recomputed. Other methods of overlap truncation (e.g. maximum energy, maximum number of simultaneously excited modes) are also usually more restrictive when performing recomputation of overlaps as opposed to their original computation as well, and this property should be utilized to avoid unnecessary recomputation in a semi-direct approach.

Since all possible grandnodes of the current branch are computed before computing the current branch, the grandnode tracking procedure described in Section 1.3.5 (and the computational improvements it provides) can still be utilized in the semi-direct variant of the proposed algorithm. Timing results for the semi-

direct approach described here for various amounts of memory usage can be found in Section 1.4.2.

1.3.6.3 Finite temperature calculations

The algorithmic approach for calculating FC overlaps at 0 K described previously can be extended to support finite temperature calculations as well, for which hot bands must be computed. Hot bands appear in vibronic spectroscopy at finite temperatures when the electronic transition originates from an excited vibrational state.

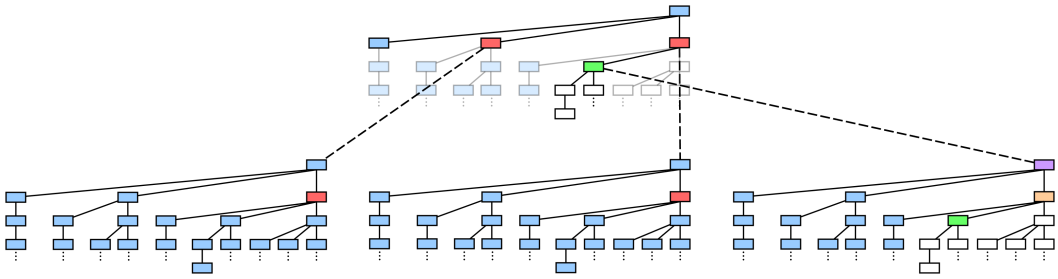
To extend the proposed algorithm to support hot band calculations, a nested tree structure is used for storing FC overlaps. The top-level tree (called an *outer tree*), has nodes that represent different vibrational states within the initial electronic state. Each node of the top-level tree has a pointer to the head of a bottom-level tree (called an *inner tree*). Each node of an inner tree represents a different vibrational state within the final electronic state. Both inner and outer trees have the same organizational structure, shown in Figure 1.7. The inner and outer tree structures are recursively defined (in the C programming language) as

```
struct Inner_Tree {
  double overlap;
  Inner_Tree* children;
}
```

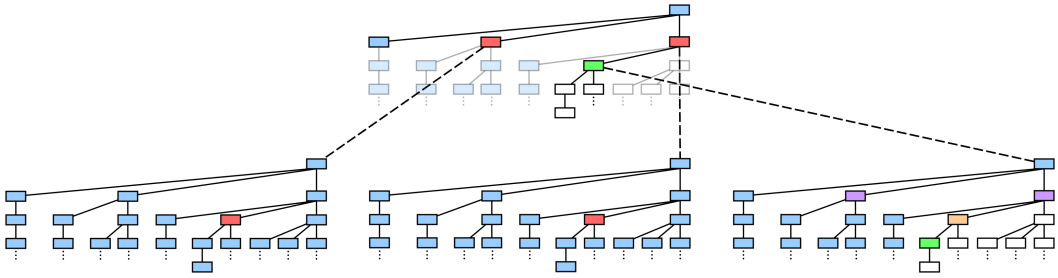
```
struct Outer_Tree {
  Inner_Tree* tree_head;
  Outer_Tree* children;
}
```

Each inner tree node stores an FC overlap and a memory pointer to an array of other inner tree nodes (its children). Each outer tree node stores a memory pointer to the head node of an inner tree and a memory pointer to an array of other outer tree

Step 1:



Step 2:



Step 3:

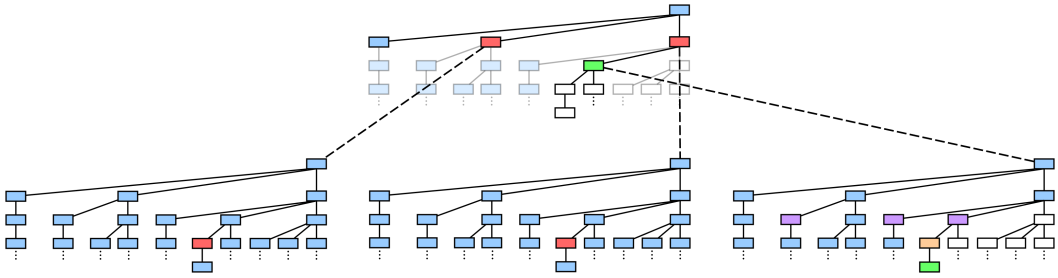


Figure 1.13: See Figure 1.14 for second part of Figure 1.13.

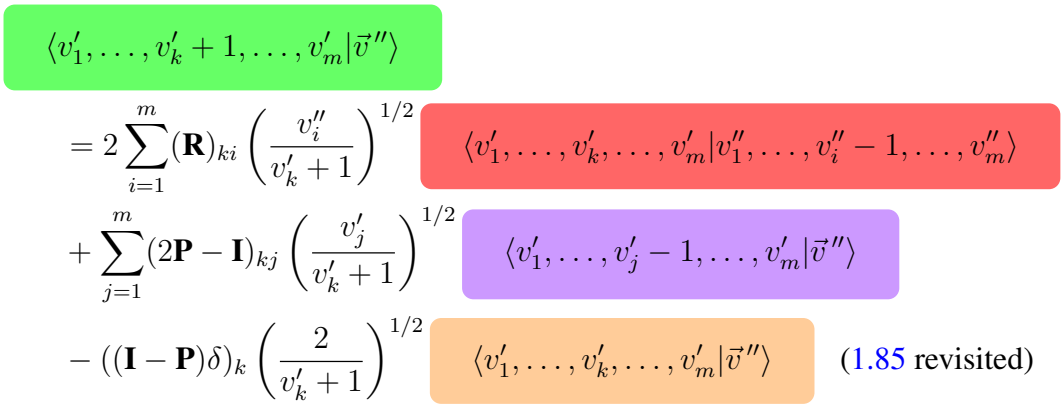


Figure 1.14: Figure 1.13 is a step-by-step illustration of the algorithm for computing FC overlaps between all desired vibrational states in the final electronic state and a fixed *excited* vibrational state in the initial electronic state (hot bands). Note that hot band calculations require tracking of cousin nodes, shown in red. The progression shown in bottom-right tree is the same as in Figure 1.9.

nodes (its children). The algorithm for computing FC overlaps while accounting for hot bands takes the form of a two-level recurrence algorithm:

Top level:

The top-level recurrence is in charge of computing the FC overlap for the inner tree head associated with each outer tree node. Since the head of each inner tree corresponds to a ground vibrational state in the final electronic state, $\langle \vec{0}\,' |$, the top-level recurrence is in charge of computing $\langle \vec{0}\,' | v''_1, \ldots, v''_i + 1, \ldots, v''_m \rangle$ using equation 1.84. These calculations are done using the same dependency tracking algorithm described in Sections 1.3.4 and 1.3.5.

Bottom level:

After an FC overlap $\langle \vec{0}\,' | \vec{v}\,'' \rangle$ is computed by the top-level recurrence algorithm, the FC overlaps $\langle \vec{v}\,' | \vec{v}\,'' \rangle$ for a fixed vibrational state $|\vec{v}\,''\rangle$ in the initial electronic state must be computed using the bottom-level recurrence algorithm. The bottom-level recurrence algorithm is very similar to the algorithm described in Sections 1.3.4 and 1.3.5, with an additional consideration. Now, not only must grandnodes be tracked, but what I call cousin nodes as well. The cousin nodes correspond to FC overlaps found in the first summation of equation 1.85.

Figure 1.13 illustrates how both grandnode and cousin tracking is done in the bottom-level recurrence algorithm. At each step in the figure, the top tree is an outer tree whose nodes correspond to vibrational states within the initial electronic state, while the trees at the bottom of each picture are inner trees whose nodes correspond to vibrational states within the final electronic state. The nodes shown in red are cousin nodes. The outer tree cousin nodes are tracked during the top-level recurrence using a procedure similar to the one used to track grandnodes. The inner tree cousin nodes corresponding to each outer tree cousin node are tracked during the bottom-level recurrence by updating the inner tree cousin nodes at the same time and in the same way as the inner tree parent node at each step.

At the end of the algorithm, we will have an ensemble of outer tree nodes, each corresponding to a different vibrational state in the initial electronic state $|\vec{v}\,''\rangle$, whose inner trees contain FC overlaps that are weighted by a Boltzmann factor determined by the energy of $|\vec{v}\,''\rangle$ and the temperature.

It is worth discussing the additional computational efforts required for finite temperature calculations compared to 0 K calculations. In a finite temperature calculation, each possible vibrational state in the initial electronic state, $|\vec{v}''\rangle$, will have associated with it an inner tree structure containing FC overlaps, $\langle\vec{v}'|\vec{v}''\rangle$. Each of these inner tree structures will have the same number of nodes, so the memory consumption for a finite temperature calculation will be equal to the memory consumption of a 0 K calculation multiplied by the number of vibrational states considered in the initial electronic state. Some additional memory is needed for the storage of $\mathrm{Outer\_Tree}$ objects, but this extra cost is almost always negligible.

The amount of computation increases by slightly more than a factor equal to the number of vibrational states in the initial electronic state when moving from a 0 K calculation to a finite temperature calculation. In a 0 K calculation, equation 1.86 can be used to compute all FC overlaps, but in a finite temperature calculation, equation 1.85 is used to compute FC overlaps with excited initial vibrational states. Equation 1.85 has more terms than equation 1.86, so computing each FC overlap for a finite temperature calculation will require more floating-point operations than for a 0 K calculation.

1.4 Performance results

In this section, timing results from computations using the proposed algorithm (and the semi-direct variant) are reported. All computations were performed using a single core on a compute node of the Stampede supercomputer at the Texas Advanced Computing Center (single core of a Xeon E5-2680 2.7GHz processor).

1.4.1 Timing comparisons with searching algorithms

To demonstrate the computational efficiency of the proposed algorithm, FC overlap calculations were run for several systems with varying numbers of modes. The run times for a given computation using the proposed algorithm were compared against run times for the same computations using the programs FCfast (searching algorithm implementation using a binary tree data structure) and ezSpectrum (searching algorithm implementation using a one-dimensional array data structure). These two programs were chosen as they are freely available codes that are representative of implementations of the searching algorithm described in the introduction of Section 1.3.2.

For comparisons with FCfast, an energy cutoff was used to truncate the number of FC overlaps that needed to be computed. For comparisons with ezSpectrum, a "maximum sum of quanta" cutoff was used to limit the number of FC overlaps considered in the calculation.[1] For each comparison, both algorithms calculate the same number of FC overlaps using the same recurrence equations.

In the calculations below, the maximum amount of memory that the program was allowed to use was set to be large enough to store all computed FC overlaps, so each calculation completed without having to recompute any calculated FC overlaps. Timing results for calculations that restricted memory usage such that all FC overlaps were not able to be stored simultaneously can be found in Section 1.4.2.

[1] An energy cutoff would be used in a real spectral simulation, but the "maximum sum of quanta" cutoff is the only way to fairly compare the proposed algorithm with the searching algorithm implementation used in the program ezSpectrum

The first system used for comparison with the binary-tree searching algorithm was oxirane (C_2H_4O), which has 15 vibrational modes. Both the searching algorithm and the proposed algorithm were used to compute FC overlaps for all vibrational states in the final electronic state with energy less than $24,000$ cm^{-1}. In total, about 181 million FC overlaps were computed. The searching algorithm performed the FC overlap computation in 350 seconds, while the proposed algorithm took roughly 10 seconds to perform the same calculation, resulting in an overall speedup of around $35\times$. The second system used for comparison with the binary-tree searching algorithm was cyclopentadienone (C_5H_4O), which has 24 vibrational modes. An energy cutoff of $9,000$ cm^{-1} was used and 87 million FC overlaps were computed. For this computation, the searching algorithm ran for 157 seconds, while the proposed algorithm took only 5 seconds (a speedup of $31\times$). A summary of these calculations and timing results can be found in Figure 1.15. The frequencies listed for each system are for the final electronic state.

For comparison with the two-dimensional array searching algorithm, the same system, adenine ($C_5H_5N_5$), was used twice. For the first run, the "maximum sum of quanta" cutoff was set at 7, and 53 million FC overlaps were computed. For the second run, the "maximum sum of quanta" cutoff was increased to 8, and 314 million FC overlaps were computed. The proposed algorithm performed 33 and 32 times faster than the two-dimensional array searching algorithm for the first and second calculations, respectively. A summary of the calculations and timing results can be found in Figure 1.16.

System 1: Oxirane

Number of modes:	15
Maximum energy:	24,000 cm^{-1}
Number of FC overlaps:	181,031,776

	Computation time (seconds)
Searching algorithm	347.66
Proposed algorithm	9.77
Speedup	**35**$\times$

Frequencies (cm^{-1}):	855.820	878.193	968.188	1044.416	1163.690	1307.469
	1318.029	1327.971	1436.615	1610.336	1674.187	3565.469
	3590.512	3726.800	3729.562			

System 2: Cyclopentadienone

Number of modes:	24
Maximum energy:	9000 cm^{-1}
Number of FC overlaps:	87,551,551

	Computation time (seconds)
Searching algorithm	157.01
Proposed algorithm	5.01
Speedup	**31**$\times$

Frequencies (cm^{-1}):	236.401	447.513	457.548	489.526	662.765	681.503
	701.521	790.656	796.675	895.196	900.191	951.277
	1041.944	1113.900	1187.124	1293.426	1349.620	1358.754
	1501.384	1615.119	3234.647	3243.767	3250.560	3261.427

Figure 1.15: Timing comparisons between a binary tree searching algorithm and the proposed tracking algorithm.

System 1: Adenine

Number of modes:	39
Maximum sum of quanta:	7
Number of FC overlaps:	53,524,680

	Computation time (seconds)
Searching algorithm	74.34
Proposed algorithm	2.26
Speedup	**33**×

System 2: Adenine

Number of modes:	39
Maximum sum of quanta:	8
Number of FC overlaps:	314,457,495

	Computation time (seconds)
Searching algorithm	464.02
Proposed algorithm	14.71
Speedup	**32**×

Figure 1.16: Timing comparisons between a two-dimensional array searching algorithm and the proposed tracking algorithm.

1.4.2 Semi-direct algorithm timings

To demonstrate the effectiveness of the semi-direct variant of the proposed algorithm described in Section 1.3.6.2, I re-ran calculations for oxirane and cyclopentadienone with a fixed maximum tree size. For oxirane, the original conventional algorithm finished in 10.07 seconds and required 2.7 GB of memory. When limiting the memory usage to 305 MB, the run time increased to 16.58 seconds, and limiting the memory usage even further to 61 MB led to a run time of 30.99 seconds. In summary, for oxirane, I was able to reduce the memory usage by a factor of 44 and only incur a factor of 3 increase in run time. For cyclopentadienone, the original computation took 5.11 seconds using 1.3 GB of memory. The semi-direct runs took 7.38 seconds using 152 MB of memory and 15.17 seconds using 15.2 MB of memory. For cyclopentadienone, I was able to reduce the memory consumption by a factor of 86 while only incurring a run time increase of a factor of 3. A summary of these results can be found in Figure 1.17.

1.4.3 Franck-Condon calculations for large molecules

In their prescreening paper [48], Jankowiak *et al.* display results of a benchmark calculation of the FC profile for a polycyclic aromatic hydrocarbon derivative ($C_{102}H_{51}N_3$) to demonstrate the effectiveness of their prescreening technique. To demonstrate the proposed algorithm's ability to handle large molecules with many vibrational modes, and its ability to utilize overlap truncation procedures determined by prescreening methods, I will now display timings for a similar calculation here. Specifically, I aim to simulate their largest calculations which use a maximum

System 1: Oxirane

	Computation time (seconds)	Memory required (MB)
Original algorithm	10.07	2698
Semi-direct variant	16.58	305
	30.99	61

System 2: Cyclopentadienone

	Computation time (seconds)	Memory required (MB)
Original algorithm	5.11	1305
Semi-direct variant	7.38	152
	15.17	15.2

Figure 1.17: Timing comparisons between various amounts of memory usage with the semi-direct approach described in Section 1.3.6.2.

number of simultaneously excited modes of 4 and maximum energies of 5000 cm^{-1} and 13880 cm^{-1}.

In their supporting information, Jankowiak *et al.* reported the final state frequencies used for each vibrational mode in the system, as well as the maximum allowed quantum level associated with each mode (determined by prescreening). This data is the only information required for my reproduction of the FC overlap calculation, since I am interested in only the computational performance of the proposed algorithm, not the resulting FC overlaps; as a result, random values for the Duschinsky matrix **S** and displacement vector **d** were used in my calculation. This should not be a concern, though, as the timings of my calculations are independent of the values of **S** and **d**; the same number of FC overlaps are computed regardless of the random values used.

My reproduction of the calculation required the computation of 1.24 billion overlaps for the 5000 cm^{-1} maximum energy calculation and 2.65 billion overlaps for the 13880 cm^{-1} maximum energy calculation. These FC overlap counts are slightly different than the ones reported in the Jankowiak paper (1.48 billion and 2.98 billion, respectively). The reason for this is not immediately clear, although this small difference is not regarded as crucial, as the sizes are similar and the current example is intended only to show qualitative behavior and capabilities of the algorithm.

For the 5000 cm^{-1} calculation, the semi-direct variant of the proposed algorithm was able to compute the desired 1.24 billion overlaps in 44.08 seconds using 14 GB of memory, 45.42 seconds using 1491 MB of memory, and 74.23 seconds using only 152 MB of memory. For the 13880 cm^{-1} calculation, the semi-direct variant of the proposed algorithm was able to compute the 2.65 billion FC overlaps in 87.95 seconds using 14 GB of memory, 94.35 seconds using 2635 MB of memory, and 138.77 seconds using only 350 MB of memory. A summary of the results can be found in Figure 1.18.

It is difficult to directly compare our timings (44 seconds and 88 seconds) with those of Jankowiak (70 minutes and 140 minutes), as different processors were used for each set of calculations. Based on the results in Section , though, I think it is reasonable to assume a 30 to 35 $\times$ speedup after accounting for differences in processor speeds.

System 3: Polycyclic aromatic hydrocarbon derivative

Number of modes:	462
Maximum energy:	5000 cm^{-1}
Number of FC overlaps:	1,248,403,212

	Computation time (seconds)	Memory required (MB)
Original algorithm		18603
Semi-direct variant	44.08	14000
	45.42	1491
	74.23	152

Maximum energy:	13880 cm^{-1}
Number of FC overlaps:	2,651,587,371

	Computation time (seconds)	Memory required (MB)
Original algorithm		39512
Semi-direct variant	87.95	14000
	94.35	2635
	138.77	350

Figure 1.18: Timing and memory usage results for conventional and semi-direct variant of the proposed algorithm for FC calculations for a PAH derivative.

1.4.4 PAPI performance data

In an effort to precisely explain the vast improvement in computational performance that the proposed algorithm achieves over existing implementations (specifically over FCfast; see Figure 1.15), I have used a tool called Performance Application Programming Interface (PAPI) to produce detailed information about the computation that occurs in the simulations discussed in Section 1.4.3. PAPI is able to provide detailed information regarding memory operations (Level 1/2/3 data/instruction cache misses/accesses/hits, total number of load/store instructions, data/instruction translation lookaside buffer (TLB) misses) and floating-point operations (number of floating-point operations executed) that can help reveal the reasons behind the proposed algorithm's improved performance. I used PAPI to compare the oxirane calculation (described in Figure 1.15) performed using my algorithm (fc_squared) and the same calculation performed using FCfast. The measurements are reported in Figure 1.19.

In Section 1.3.2, I motivate the use of my proposed algorithm by remarking that existing methods spend an unnecessarily large amount of time searching data structures for stored overlaps. For a binary-tree implementation, searching the data structure involves performing many memory operations (reaching a tree node, picking a child of the node, moving to the memory address of the child, repeat). We can predict, then, that an implementation of the proposed algorithm (which does not have to search data structures) should perform less memory operations (data loads and stores) than an implementation of the binary-tree searching algorithm. The PAPI measurements in Figure 1.19 confirm this prediction, specifically

the PAPI_LD_INS and PAPI_SR_INS keywords. These PAPI counters count the number of load and store instructions executed, and it can be seen that my proposed algorithm executes $8\times$ and $7\times$ fewer load and store instructions than FCfast, respectively.

While the difference in the number of memory operations is significant, it does not explain the entire $\sim 30\times$ speedup. One can argue, though, that the remainder of the speedup can be explained by the significant improvement in cache utilization that my algorithm exhibits. Three PAPI keywords in particular are important to note: PAPI_L1_DCM (level 1 data cache misses), PAPI_L2_DCM (level 2 data cache misses), and PAPI_TLB_DM (data translation lookaside buffer misses). The proposed algorithm incurs $30\times$ fewer level 1 cache misses, $138\times$ fewer level 2 cache misses, and $93\times$ fewer TLB misses than the binary tree algorithm.

Because the two algorithms perform vastly different numbers of memory operations, it is perhaps more appropriate to look at the cache misses in terms of cache misses per access (an access being a load or a store). Let us define new variables L1_DCMPA and L2_DCMPA (L1 and L2 data cache miss per access, respectively), whose values can be found in Figure 1.20. We can see that fc_squared misses the L1 cache about $4\times$ less often than FCfast, and misses the L2 cache about $17\times$ less often than FCfast. Another interesting finding is that FCfast appears to have less than $2\times$ fewer L2 misses than L1 misses, suggesting that when FCfast misses the L1 cache, it most likely also misses the L2 cache. This result implies that the binary tree searching algorithm performs memory accesses very "far" away from each other in the available memory.

PAPI_L1_DCM
Level 1 data cache misses

FCfast	9430451569
fc_squared	317673321
Ratio	30

PAPI_L2_DCM
Level 2 data cache misses

FCfast	5290045144
fc_squared	38393153
Ratio	138

PAPI_LD_INS
Load instructions executed

FCfast	202447772459
fc_squared	24387890099
Ratio	8.3

PAPI_SR_INS
Store instructions executed

FCfast	48476752303
fc_squared	7247439989
Ratio	6.7

PAPI_TLB_DM
Data translation lookaside buffer (TLB) misses

FCfast	700872569
fc_squared	7513486
Ratio	93

PAPI_FP_OPS
Floating-point operations executed

FCfast	8518791128
fc_squared	4808713823
Ratio	1.77

PAPI_TOT_CYC
Total cycles executed

FCfast	1087734178651
fc_squared	28122384388
Ratio	39

Figure 1.19: Performance Application Programming Interface (PAPI) measurements for the oxirane calculation using FCfast and fc_squared.

L1_DCMPA
Level 1 data cache misses per access

FCfast	0.0375828
fc_squared	0.0100417
Ratio	3.7

L2_DCMPA
Level 2 data cache misses per access

FCfast	0.0210822
fc_squared	0.0012136
Ratio	17.4

Figure 1.20: L1 and L2 data cache miss per access rate for FCfast and fc_squared calculations.

It is difficult to say why exactly the proposed algorithm achieves better cache reuse, but some possibile explanations are:

1. The proposed algorithm constructs the overlap tree in such a way that the $\mathrm{children}$ of any given node are stored contiguously in memory. Accessing children of a node consecutively, then, uses contiguous memory reads.

2. Many of the proposed algorithm's memory accesses are reading pointers from a contiguous $\mathrm{grandnodes}$ array.

In any case, the proposed algorithm, in addition to performing fewer memory operations, achieves better cache reuse than the binary tree searching algorithm, which provides reasoning for the $30\times$ reduction in run time that we observe.

1.5 Applications: Linear carbon chain HC_7H

The fc_squared implementation of the proposed algorithm has been used for the assignment of transitions for the spectra of two molecules, as well as for preliminary calculations for several vibronic coupling simulations. The first molecule

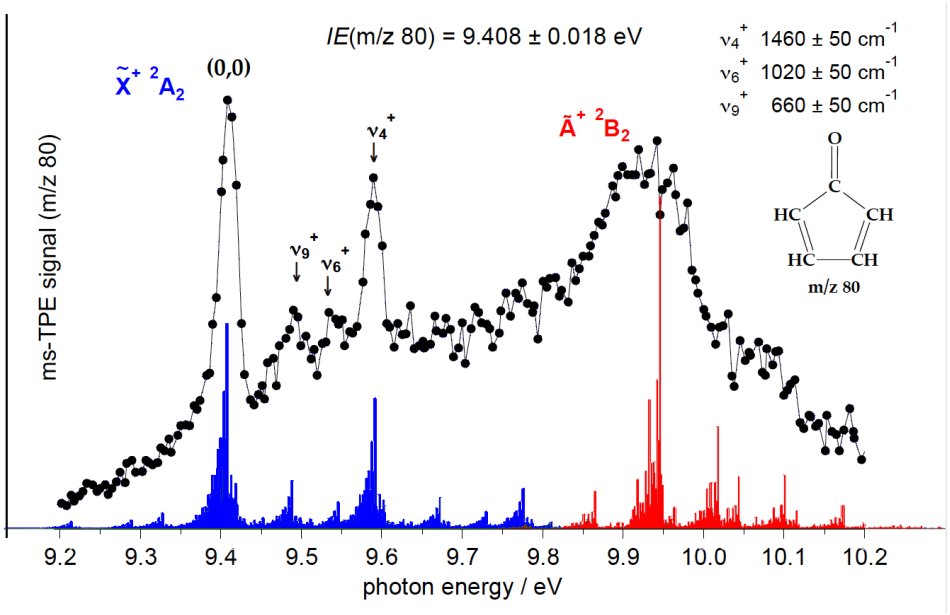


Figure 1.21: Finite temperature FC spectrum for the ionisation of cyclopentadienone to its two lowest-lying cation states $\tilde{\mathrm{X}}^{+\,2}\mathrm{A}_2$ and $\tilde{\mathrm{A}}^{+\,2}\mathrm{B}_2$.

assigned was cyclopentadienone (C_5H_4O), for which finite temperature calculations were performed to study the ionisation from the neutral ground state $\tilde{\mathrm{X}}^{1}\mathrm{A}_1$ to the two lowest-lying cation states $\tilde{\mathrm{X}}^{+\,2}\mathrm{A}_2$ and $\tilde{\mathrm{A}}^{+\,2}\mathrm{B}_2$. The resulting spectrum is shown in Figure 1.21. The second molecule assigned was the linear carbon chain HC_7H, for which finite temperature calculations were performed to study its $A^3\Sigma_u^- \leftarrow X^3\Sigma_g^-$ transition. The work on HC_7H will be the primary focus of this section.

The spectrum corresponding to the $A^3\Sigma_u^- \leftarrow X^3\Sigma_g^-$ transition of the linear carbon chain HC_7H has been measured in several studies [35, 24] and, in each, attempts have been made to assign transitions to the spectrum's peaks. My work on

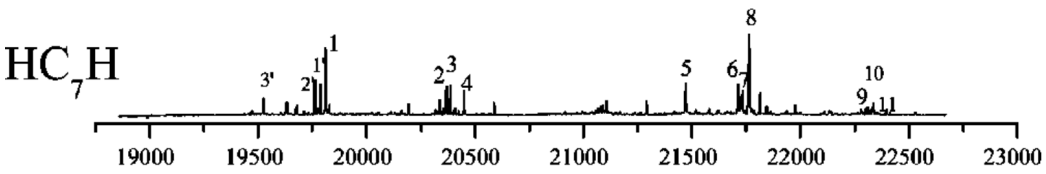


Figure 1.22: REMPI spectrum measured by Ding *et al.*

HC_7H is twofold:

1. **Assign transitions to peaks of the REMPI spectrum**: Around 2003, the Schmidt group at the University of New South Wales (Ding *et al.*) performed a spectroscopic measurement of HC_7H using a resonance-enhanced multiphoton ionization (REMPI) technique. Their spectrum is shown in Figure 1.22. There are several sticks in their spectrum whose assignments are unknown, most notably are the sticks lower in energy than the assigned origin stick (which corresponds to the $0 \rightarrow 0$ transition). In [24], Ding assigns these as sequence bands (hot bands such that the vibrational states in the ground and upper electronic states are the same, but the vibrational states are not ground vibrational states, $|\vec{0}\rangle$), but specific assignments were not made. In this work, I aim to assign these peaks (and others unassigned by Ding *et al.*) by calculating the spectrum using a finite-temperature Franck-Condon calculation,[2] given a Duschinsky matrix (**S**), displacement vector (**d**), and upper/lower-state frequencies (ω_i', ω_i'') determined using *ab initio* calculations.

[2] A finite-temperate calculation is required to compute sequence bands, since sequence bands originate from transitions that being at an excited vibrational state, which would not be populated at 0 K

2. **Fit the FC spectrum to the experimental spectrum to obtain accurate ground- and upper-state frequencies**: More recently, the photoelectron spectrum of HC_7H was measured by the University of Wisconsin Biotechnology Center (UWBC) using mass spectroscopy (shown in Figure 1.23). This is the most highly-resolved (i.e. detailed) spectrum to date. Because of its high resolution, it can be used to help determine accurate ground and upper state frequencies by "fitting" the calculated FC spectrum to the measured spectrum. The second part of my work involves tweaking the *ab initio* frequencies to fit the FC spectrum to the observed data, resulting in (ideally) accurate frequencies for the ground and upper states.

The molecule HC_7H is a linear molecule with 9 atoms and 22 vibrational modes. To determine the normal mode frequencies in the ground and upper states (ω_i'' and ω_i', respectively), as well as the Duschinsky matrix **S** and displacement vector **d**, *ab initio* calculations were performed using EOM-CCSD and the ANO2 basis set. The calculated frequencies for each vibrational mode can be found in Table 1.1. Each Franck-Condon overlap calculation performed was a finite-temperature calculation at temperature $T = 300K$. Each overlap calculation was quite large, requiring the storage of roughly 783 million overlaps (11.67 GB of memory) and running for about one minute using fc_squared.

The majority of my work fitting upper/lower-state frequencies to the UWBC spectrum was focused around the origin energy region, as an accurate fitting of this region is crucial to the understanding of how the frequencies change between the lower and upper states. The origin energy region is outlined with a dotted line in

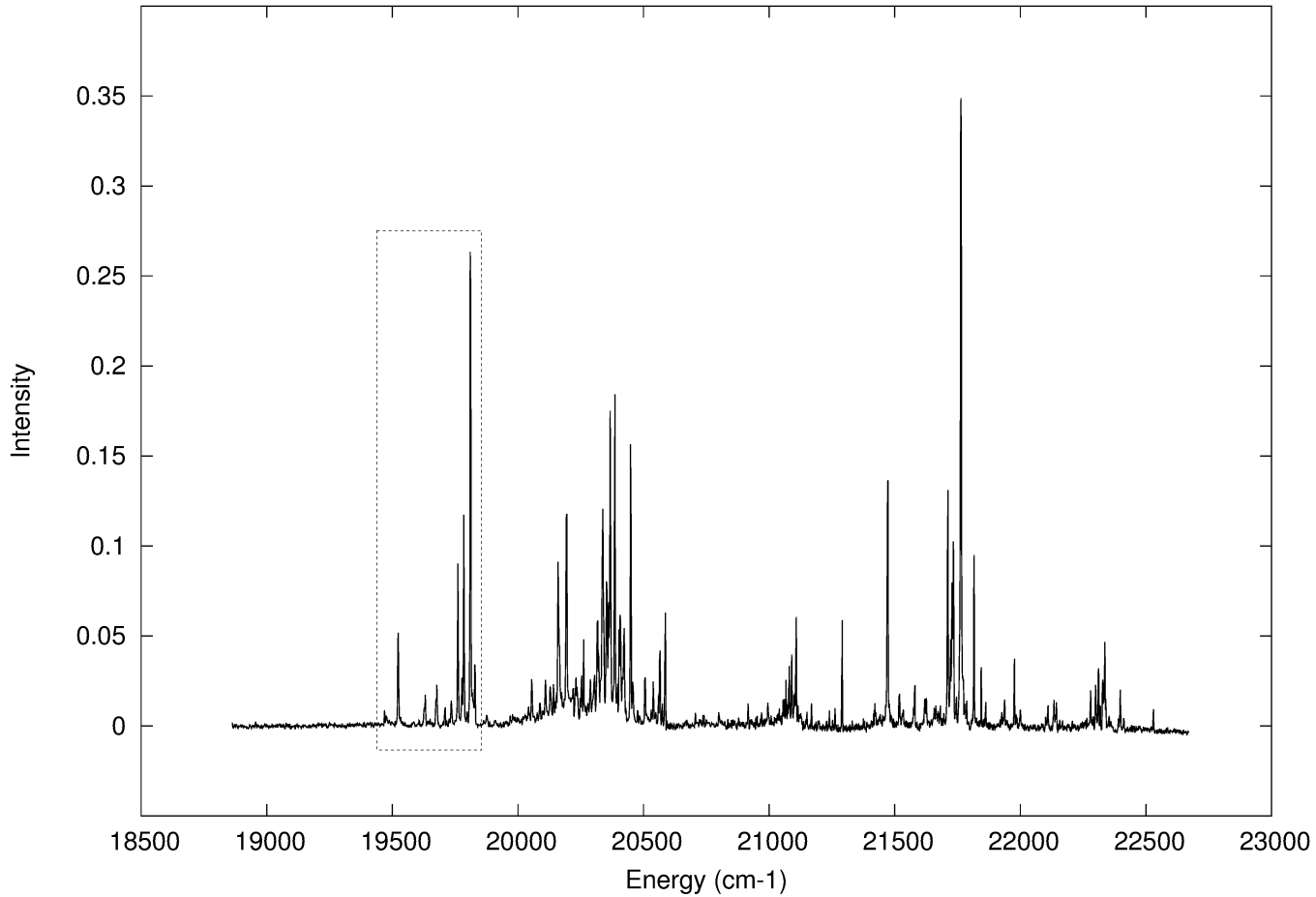


Figure 1.23: An experimentally-measured vibronic spectrum of HC_7H in the 18500 to 23000 cm^{-1} range. The dotted box represents the part of the spectrum that is believed to contain the 0-0 transition band and corresponding combination hot bands. My initial study focused on the assignment of the dotted region of the spectrum.

		ν_1	ν_2	ν_3	ν_4	ν_5	ν_6	ν_7	ν_8
Ab initio	Ground	3478	2075	1683	552	3476	2063	1423	1048
	Upper	3469	2056	1567	556	3459	1642	3742	1108
Fitted	Ground	3478	2075	1683	551	3476	2063	1423	1048
	Upper	3469	1950	1567	554	3459	1642	3742	1113

		ν_9	ν_{10}	ν_{11}	ν_{12}	ν_{13}	ν_{14}	ν_{15}
Ab initio	Ground	596	419	164†	596	419	375	70†
	Upper	293	411	178†	289	406	337	72†
Fitted	Ground	575	419	184	567	414	380	70
	Upper	288	393	184	278	389	330	70

† Transitions involving these modes produced a very busy spectrum at 300 K because of their low frequencies. For clarity, I set the ground and upper state frequencies for these modes to be equal, effectively removing them from the spectrum.

Table 1.1: *Ab initio* and fitted frequencies (in cm^{-1}) for each mode ν_i in both the ground and upper electronic states.

Figure 1.23 and shown in more detail in Figure 1.24. The origin energy region of the Franck-Condon spectrum calculated using the *ab initio* frequencies is shown in Figure 1.25. It is clear that the agreement is unsatisfactory.

To fit the ground- and upper-state frequencies to the observed data, hundreds of Franck-Condon calculations with performed using varying frequencies in an attempt to match the sticks in Figure 1.25 with the observed peaks. Each modification of the frequencies was manually chosen. In total, the fitting process required upwards of 200 Franck-Condon calculations, each taking roughly one minute using fc_squared. The resulting fitted frequencies are shown in Table 1.1 and the fitted spectrum is shown in Figure 1.26.

With the fitted frequencies available, I compared the full calculated spectrum with the REMPI spectrum calculated by Ding *et al.* and assigned transitions to their numbered sticks. The comparison is shown in Figure 1.27 and the transition assignments can be found in Table 1.2. Overall, the fitted stick spectrum appears to be in a good agreement with the REMPI spectrum. The transition assignments made by Ding *et al.* are mostly confirmed by my calculations (barring the 4_0^1 assignment of stick 4). In our fitted spectrum, the stick associated with the 4_0^1 transition was roughly 80 cm^{-1} lower than stick 4 in the REMPI spectrum. The sticks below the origin peak are, in fact, sequence bands, as predicted by Ding *et al.*, with sticks 3' and 1' assigned to two different sequence bands in close proximity.

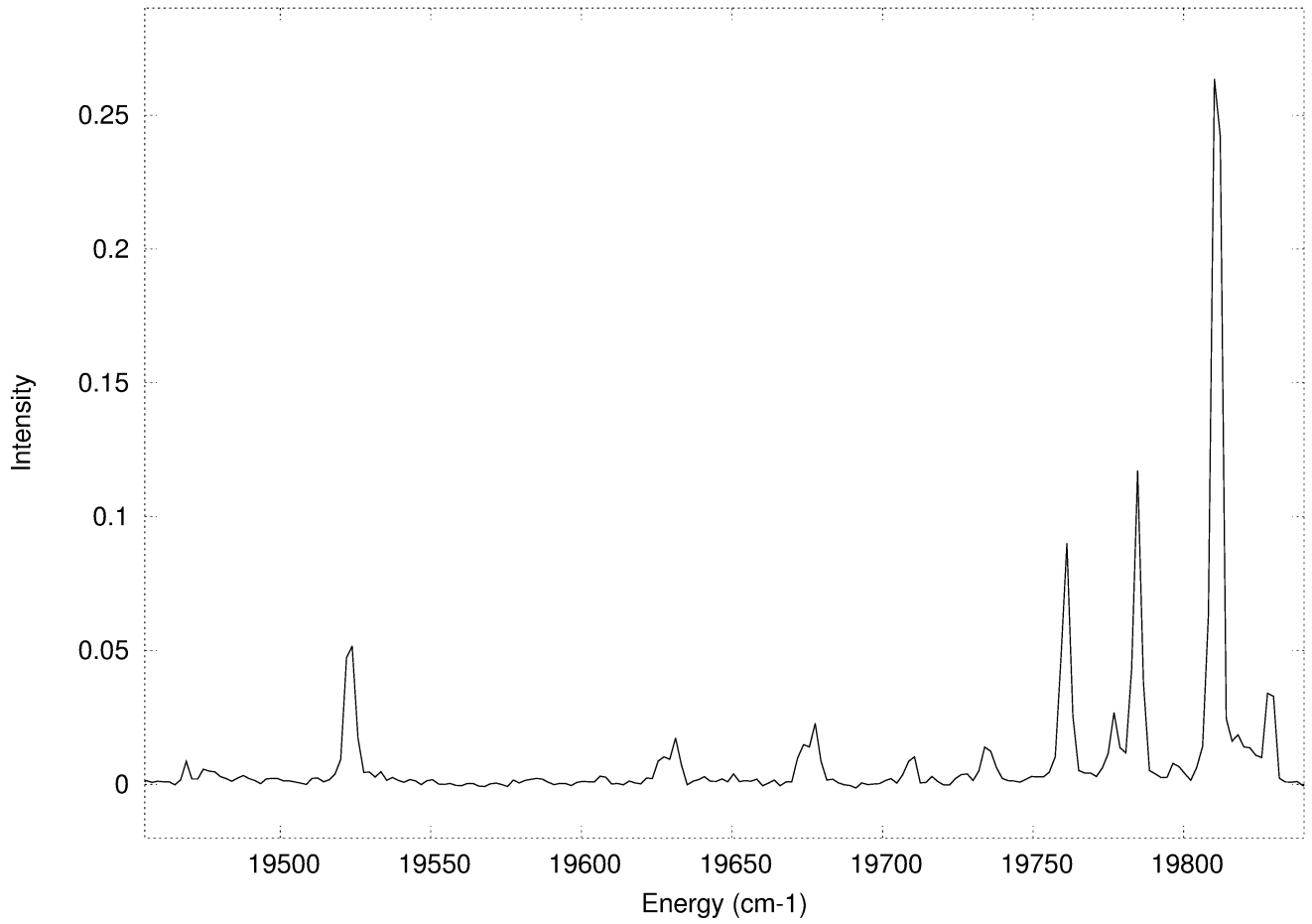


Figure 1.24: The dotted portion of the spectrum in Figure 1.23. It is believed that the large peak around 19830 cm^{-1} corresponds to the 0-0 transition; peaks lower than 19830 cm^{-1} in energy correspond to hot bands (transitions that begin at an excited vibrational state).

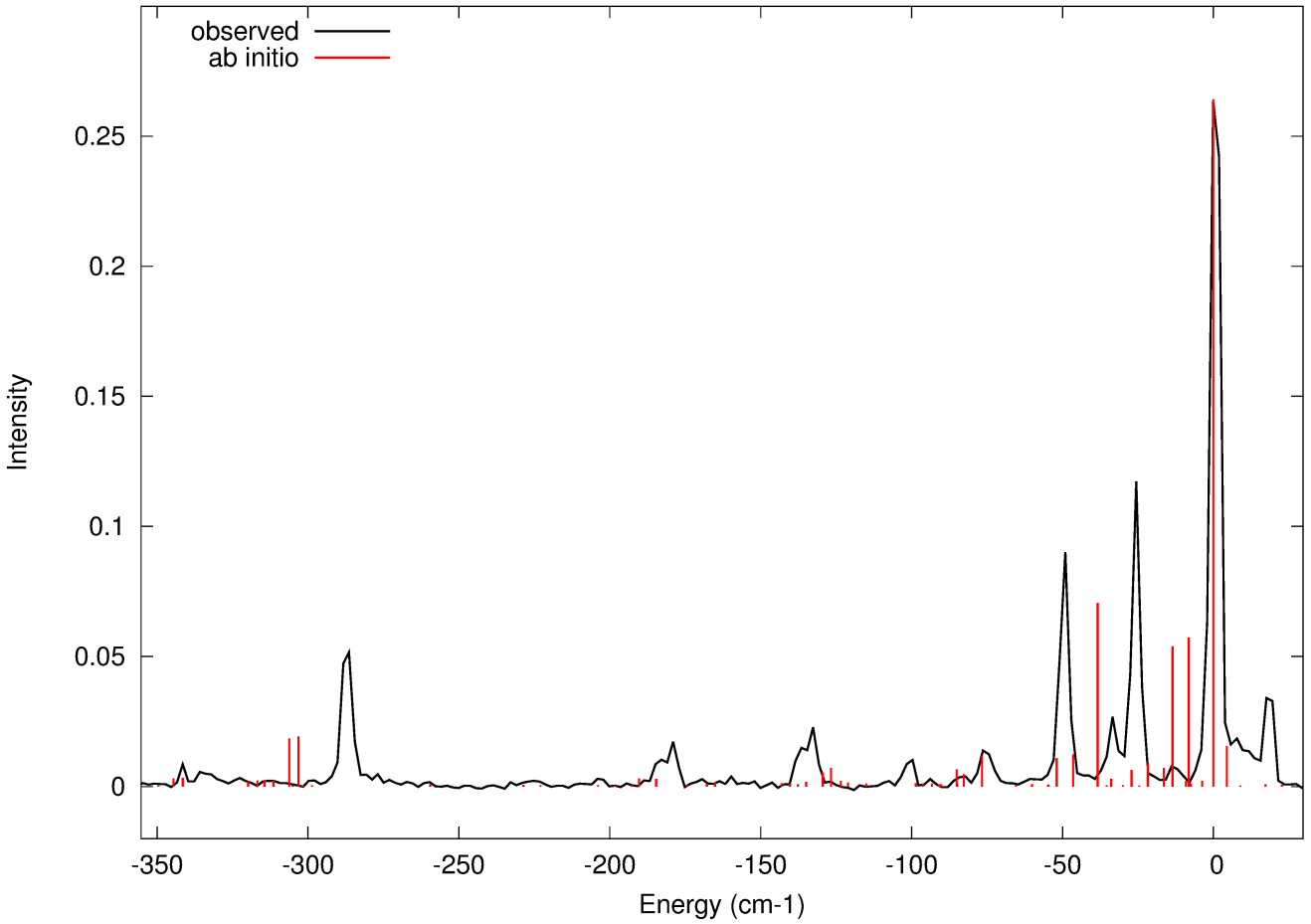


Figure 1.25: The stick spectrum produced by a Franck-Condon calculation using the *ab initio* calculated frequencies.

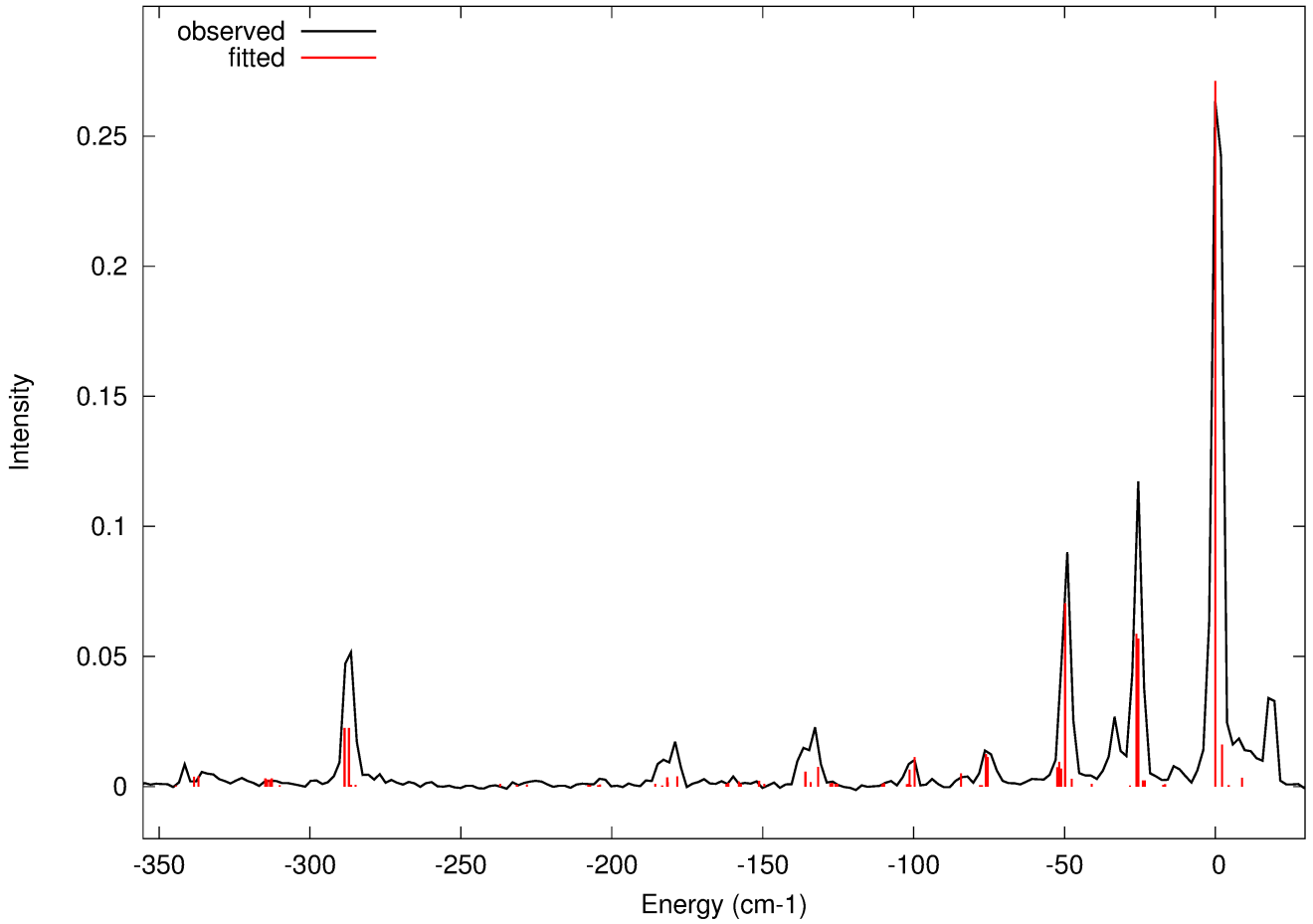


Figure 1.26: The stick spectrum produced by a Franck-Condon calculation using frequencies manually fitted to the observed spectrum. The transition assignments for the numbered sticks can be found in Table 1.2.

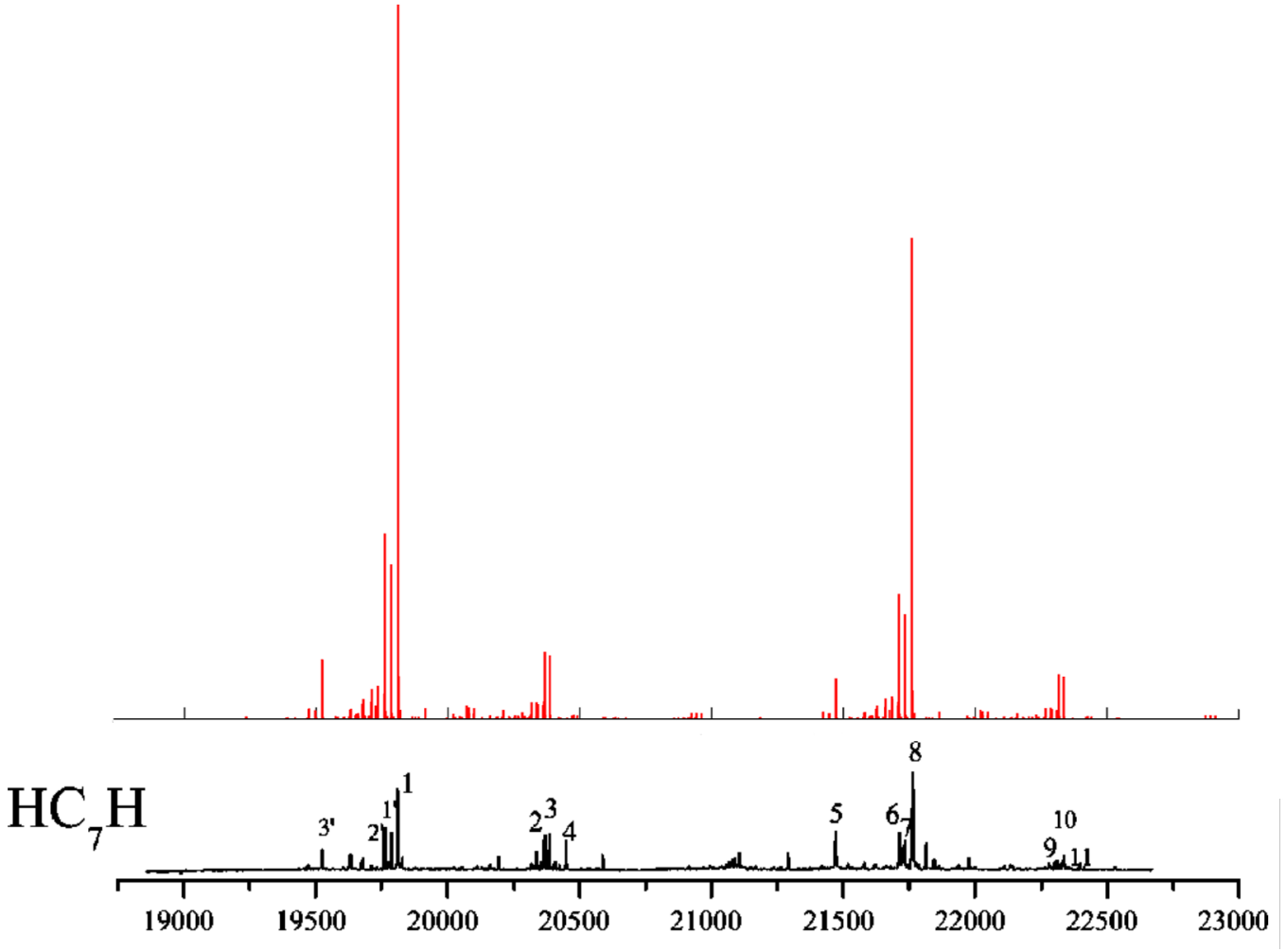


Figure 1.27: Bottom: The REMPI spectrum measured by Ding *et al.*. The numbers on certain sticks correspond to assignments they made (shown in Table 1.2). Top: The stick spectrum calculated using fc_squared at $T = 300\ K$.

Stick Number	Assignments	
	Ding *et al.*	Current
3'		9_1^1 12_1^1
2'		14_1^1
1'		10_1^1 13_1^1
1	0	0
2	14_0^2	14_0^2
3	10_0^2	10_0^2
4	4_0^1	
5		$2_0^1 + 9_1^1$ $2_0^1 + 12_1^1$
6		$2_0^1 + 14_1^1$
7		$2_0^1 + 10_1^1$ $2_0^1 + 13_1^1$
8	2_0^1	2_0^1
9	$2_0^1 + 14_0^2$	$2_0^1 + 14_0^2$
10	$2_0^1 + 10_0^2$	$2_0^1 + 10_0^2$
11	$2_0^1 + 4_0^1$	

Table 1.2: Transition assignments for the numbered sticks in Figure 1.27. The notation a_b^c refers to a transition from $|0 \ldots v_a'' \ldots 0\rangle$, where $v_a'' = b$, to $\langle 0 \ldots v_a' \ldots 0|$, where $v_a' = c$. Similarly, $a_b^c + d_e^f$ refers to a transition from $|0 \ldots v_a'' \ldots v_d'' \ldots 0\rangle$, where $v_a'' = b$ and $v_d'' = e$, to $\langle 0 \ldots v_a' \ldots v_d' \ldots 0|$, where $v_a' = c$ and $v_d' = f$.

1.6 Conclusion and future work

In this chapter, I described how vibronic spectra are calculated using the Born-Oppenheimer approximation under the assumption of harmonicity and investigated the computational bottlenecks involved with these calculations. The computation of Franck-Condon overlaps (which correspond to transition intensities of the vibronic states in the spectrum) is the most computationally intensive part of these vibronic spectral calculations and is a major computational problem when attempting to study molecules of increasing size.

A new algorithm for computing Franck-Condon overlaps, facilitated by the use of a novel data structure, has been described in this chapter. The newly-developed algorithm uses a dependency tracking method that alleviates the computational stresses that arise from searching data structures for overlaps stored in memory. As a result, significant run-time speedups (on the order of 30 times) are achieved over the current state-of-the-art algorithms.

The algorithm, originally designed for small molecules where all FC overlaps can be stored in memory ("conventional" calculations), also can and has been extended to a semi-direct implementation and is compatible with prescreening techniques. The proposed algorithm has been implemented in a new computer program [81], which will soon be added to the CFOUR program suite for quantum chemical and spectroscopic calculations.

1.6.1 Future work

There is no doubt that the $30\times$ single-core speedup achieved by the algorithm proposed in this chapter successfully extends the reach of Franck-Condon calculations. With all of the multi-core architectures available today, though, it is shameful to not take advantage of the extra processing power that parallelization provides. Future work for this project should then focus on the parallelization of the above algorithm.

Unfortunately, because the proposed algorithm was not developed with parallelization in mind, a scheme for its parallelization is not immediately obvious. The depth-first traversal of the tree structure performed throughout the algorithm is seemingly a necessarily sequential process, and the dependency structure of the recurrence equations makes a clean distribution of work difficult. Despite this, I have done a preliminary investigation of the parallelization of the algorithm, which I will outline here.

Shared-memory parallelization of the algorithm appears possible if the tree structure is filled one "level" at a time (where level describes nodes of the same depth within the tree). When the algorithm proceeds in this way, the computation of each level only depends on nodes that already exist in the tree (e.g. all nodes at level 7 have *parents* at level 6 and *grandnodes* at level 5). The computation of each level is then a collection of calculations that do not depend on each other in any way, which can be easily distributed among multiple processes.

The downside of this approach, though, is that filling a tree one level at a

time increases the total amount of time spent traversing the tree structure. For the computation of each level, the algorithm must *repeat* the tree traversal performed for the computation of the previous level. For example, computing all nodes on level 10 requires *at least* the same amount of traversal as computing all nodes on level 9, since, to get to each node on level 10, the algorithm must necessarily travel to a node on level 9. This extra traversal is inefficient and will negate some of the computational improvement that the $\mathrm{grandnode}$ tracking procedure provides.

Future work on this topic will focus on the development of new algorithmic variants that can compute one level at a time and utilize the $\mathrm{grandnode}$ tracking procedure, without incurring the cost of the extra traversal described above.

Chapter 2

Beyond the Born-Oppenheimer approximation

So far, I have described how to calculate a vibronic spectrum with the help of the Born-Oppenheimer approximation under the assumption of harmonicity. An important assumption made in the formation of the Born-Oppenheimer approximation is that the electronic energy levels of the system of study are well-separated (i.e. $|V_i(\mathbf{R}) - V_{i'}(\mathbf{R})|$ is large in the vicinity of the absorbing state equilibrium geometry

---

Parts of this chapter are based on the following two publications:

Rabidoux, S.M.; Eijkhout, V.; Stanton, J.F. Parallelization Strategy for Large-scale Vibronic Coupling Calculations. *Journal of Physical Chemistry A*, 118(51), **2014**, 12059-12068.

Lee, K; Rabidoux, S.M.; Stanton, J.F. Cation States of Ethane: HEAT Calculations and Vibronic Simulations of the Photoelectron Spectrum of Ethane. *J. Phys. Chem. A*, 120(38), **2016**, 7548-7553.

For the first publication, the co-authors Stanton, J.F. and Eijkhout, V. supervised the project. For the second publication, my contribution was performing the actual vibronic coupling calculations with my parallel Lanczos implementation.

$\mathbf{R}_0$). Looking back at equation 1.19, the kinetic matrix entries take the form

$$\begin{aligned}
\langle\psi_i(\mathbf{r};\mathbf{R})|\,\hat{T}_n\,|\psi_{i'}(\mathbf{r};\mathbf{R})\rangle &= \sum_\alpha \frac{-\hbar\omega_\alpha}{2}\,\langle\psi_i(\mathbf{r};\mathbf{R})|\,\nabla_\alpha^2\,|\psi_{i'}(\mathbf{r};\mathbf{R})\rangle \\
&= \sum_\alpha \frac{-\hbar\omega_\alpha}{2}\,\langle\psi_i(\mathbf{r};\mathbf{R})|\psi_{i'}(\mathbf{r};\mathbf{R})\rangle\,\nabla_\alpha^2 \qquad \text{(1.14 revisited)} \\
&+ \sum_\alpha -\hbar\omega_\alpha\,\langle\psi_i(\mathbf{r};\mathbf{R})|\nabla_\alpha\psi_{i'}(\mathbf{r};\mathbf{R})\rangle\,\nabla_\alpha \qquad \text{(1.15 rev.)} \\
&+ \sum_\alpha \frac{-\hbar\omega_\alpha}{2}\,\langle\psi_i(\mathbf{r};\mathbf{R})|\nabla_\alpha^2\psi_{i'}(\mathbf{r};\mathbf{R})\rangle\,, \qquad \text{(1.16 rev.)}
\end{aligned}$$

where

$$\begin{aligned}
\langle\psi_i(\mathbf{r};\mathbf{R})|\nabla_\alpha\psi_{i'}(\mathbf{r};\mathbf{R})\rangle &= \frac{V_{i'}(\mathbf{R})-V_i(\mathbf{R})}{V_{i'}(\mathbf{R})-V_i(\mathbf{R})}\,\langle\psi_i(\mathbf{r};\mathbf{R})|\nabla_\alpha\psi_{i'}(\mathbf{r};\mathbf{R})\rangle \\
&= \frac{\langle\psi_i(\mathbf{r};\mathbf{R})|\,\nabla_\alpha\hat{H}_e\,|\psi_{i'}(\mathbf{r};\mathbf{R})\rangle - \langle\psi_i(\mathbf{r};\mathbf{R})|\,\hat{H}_e\nabla_\alpha\,|\psi_{i'}(\mathbf{r};\mathbf{R})\rangle}{V_{i'}(\mathbf{R})-V_i(\mathbf{R})} \\
&= \frac{\langle\psi_i(\mathbf{r};\mathbf{R})|\,[\nabla_\alpha,\hat{H}_e]\,|\psi_{i'}(\mathbf{r};\mathbf{R})\rangle}{V_{i'}(\mathbf{R})-V_i(\mathbf{R})}. \qquad \text{(1.19 revisited)}
\end{aligned}$$

If the potential energy surfaces $V_i(\mathbf{R})$ of a molecule are well-separated, the off-diagonal kinetic energy terms ($\langle\psi_i(\mathbf{r};\mathbf{R})|\,\hat{T}_n\,|\psi_{i'}(\mathbf{r};\mathbf{R})\rangle$ for $i \neq i'$) are small enough to be neglected, and the Born-Oppenheimer approximation is valid.

While it is often safe to assume well-separated electronic states for the majority of systems, there many systems of interest for which it is not appropriate. When two electronic states i and i' become energetically close, the kinetic matrix term in equation 1.19 blows up and can no longer be neglected. When this is the case, we say that the two states are *vibronically coupled* and can no longer be treated separately in the calculation of their vibronic spectra. Fortunately, it is not necessary to completely abandon the Born-Oppenheimer framework in the case of

vibronic coupling, but rather it is possible to *extend* the approximation to consider *groups* of electronic states, rather than individual ones.

2.1 The KDC vibronic coupling model

In their paper [54], Köppel, Domcke, and Cederbaum describe the following extension to the Born-Oppenheimer approximation: Let α be a subset of vibronically coupled electronic states that contains the electronic state of interest. Let us assume that the subset of states α is vibronically decoupled from the remainder of the states, denoted by β. We can then neglect the off-diagonal kinetic matrix elements $\langle \psi_i(\mathbf{r};\mathbf{R}) | \, \hat{T}_n \, | \psi_{i'}(\mathbf{r};\mathbf{R}) \rangle$ for which $i \in \alpha$ and $i' \in \beta$ (or vice versa), creating a block-diagonal kinetic matrix instead of the diagonal kinetic matrix that the original Born-Oppenheimer approximation produces.

Let us consider the simplest case of accounting for vibronic coupling between two electronic states, a and b. We start by making the assumption that states a and b, while strongly coupled with each other, are effectively decoupled from the rest of the electronic states. The molecular wave functions (around the energy region corresponding to electronic states a and b) are then taken to be linear combinations of the coupled electronic wave functions,

$$\Psi_A(\mathbf{r},\mathbf{Q}) = \psi_a(\mathbf{r};\mathbf{Q})\Omega_a^{(A)}(\mathbf{Q}) + \psi_b(\mathbf{r};\mathbf{Q})\Omega_b^{(A)}(\mathbf{Q}), \tag{2.1}$$

where $\Omega_i^{(A)}(\mathbf{Q})$ is the part of the nuclear wave function of vibronic state A corresponding to electronic state i. The electronic wave functions are still taken to be eigenfunctions of the electronic Hamiltonian (just as in the Born-Oppenheimer ap-

proximation), but the nuclear wave functions are now solutions to a set of coupled equations

$$\left(\begin{bmatrix} \langle \psi_a | \hat{T}_n | \psi_a \rangle & \langle \psi_a | \hat{T}_n | \psi_b \rangle \\ \langle \psi_b | \hat{T}_n | \psi_a \rangle & \langle \psi_b | \hat{T}_n | \psi_b \rangle \end{bmatrix} + \begin{bmatrix} V_a(\mathbf{Q}) & 0 \\ 0 & V_b(\mathbf{Q}) \end{bmatrix} \right) \begin{bmatrix} \Omega_a(\mathbf{Q}) \\ \Omega_b(\mathbf{Q}) \end{bmatrix} = E \begin{bmatrix} \Omega_a(\mathbf{Q}) \\ \Omega_b(\mathbf{Q}) \end{bmatrix}. \tag{2.2}$$

Remember that, as the two adiabatic potential energy surfaces $V_a(\mathbf{Q})$ and $V_b(\mathbf{Q})$, become energetically close, the coupling term $\langle \psi_a(\mathbf{r};\mathbf{Q}) | \nabla_\alpha \psi_b(\mathbf{r};\mathbf{Q}) \rangle$ (and thus $\langle \psi_a | \hat{T}_n | \psi_b \rangle$) grows to infinity. Due to their divergent nature (especially at an avoided crossing or conical intersection, i.e. nuclear configuration $\mathbf{Q}$ such that $V_i(\mathbf{Q}) = V_{i'}(\mathbf{Q})$), the nuclear kinetic energy coupling terms can be very difficult to estimate numerically, making it difficult to solve the coupled equations in equation 2.2. To remove this computational inconvenience that off-diagonal kinetic matrix elements create, it is common to replace the adiabatic electronic wave functions ψ_i with a new set of functions ψ_i^{QD} called *quasidiabatic electronic wave functions*, for which off-diagonal $\langle \psi_i^{\mathrm{QD}} | \hat{T}_n | \psi_{i'}^{\mathrm{QD}} \rangle$ terms are negligible in magnitude.

2.1.1 Quasidiabatic electronic wave functions

Quasidiabatic electronic wave functions are defined by a unitary rotation of adiabatic electronic wave functions. Because a unitary transformation is used, quasidiabatic electronic wave functions span the same space as the adiabatic electronic wave functions from which they are derived. As a result, the energy levels computed from the Schrödinger equation are the same no matter which electronic basis is used. In the earlier example, where only two electronic states are considered to

be vibronically coupled, the quasidiabatic electronic wave functions are computed by

$$\begin{bmatrix} \psi_a^{\mathrm{QD}}(\mathbf{r};\mathbf{Q}) \\ \psi_b^{\mathrm{QD}}(\mathbf{r};\mathbf{Q}) \end{bmatrix} = \begin{bmatrix} M_{aa}(\mathbf{Q}) & M_{ab}(\mathbf{Q}) \\ M_{ba}(\mathbf{Q}) & M_{bb}(\mathbf{Q}) \end{bmatrix} \begin{bmatrix} \psi_a(\mathbf{r};\mathbf{Q}) \\ \psi_b(\mathbf{r};\mathbf{Q}) \end{bmatrix} \tag{2.3}$$

where

$$[\mathbf{M}(\mathbf{Q})]^\dagger[\mathbf{M}(\mathbf{Q})] = [\mathbf{M}(\mathbf{Q})][\mathbf{M}(\mathbf{Q})]^\dagger = \mathbb{I}. \tag{2.4}$$

The unitary matrix $\mathbf{M}$ is chosen so that the values $\langle \psi_i^{\mathrm{QD}} | \hat{T}_n | \psi_{i'}^{\mathrm{QD}} \rangle$ are small enough to be safely neglected. Ideally, one would want to choose $\mathbf{M}$ such that $\langle \psi_i^{\mathrm{QD}} | \hat{T}_n | \psi_{i'}^{\mathrm{QD}} \rangle$ are exactly zero. Electronic wave functions that satisfy this condition are called *diabatic electronic wave functions*. It has been shown [71], though, that a true diabatic transformation generally does not exist for polyatomic molecules, so often the best one can do is to try to make the nuclear kinetic coupling terms as small as possible and neglect them, hence the term *quasi*diabatic.

When quasidiabatic electronic wave functions are used to expand the full wave function, the potential matrix in equation 2.2 is no longer diagonal, since quasidiabatic electronic wave functions are not eigenfunctions of the electronic Hamiltonian. Instead, the potential matrix consists of matrix elements $\langle \psi_i^{\mathrm{QD}} | \hat{H}_e | \psi_{i'}^{\mathrm{QD}} \rangle$. Essentially, by transforming the adiabatic electronic wave functions to quasidiabatic electronic wave functions, we are shifting the off-diagonal matrix elements from the nuclear kinetic energy matrix to the potential matrix. The change is beneficial, though, because the electronic Hamiltonian coupling terms that appear in the quasidiabatic framework are often easier to work with than the divergent nuclear kinetic energy coupling terms that appear in the adiabatic framework. The resulting

nuclear Hamiltonian operator is known as the KDC model Hamiltonian, as its use was popularized by Köppel, Domcke, and Cederbaum [54] in 1984.

Before moving forward, it is important that I introduce a new coordinate system that will be used for the remainder of this chapter. Instead of using the normal coordinates $\mathbf{Q}$ introduced in the previous chapter to describe our nuclear configurations in the KDC model, we will use what are known as *dimensionless* normal coordinates, $\mathbf{q}$, defined as

$$q_i = \sqrt{\frac{\omega_i}{\hbar}} Q_i, \tag{2.5}$$

where ω_i is the frequency associated with normal mode i. The use of dimensionless normal coordinates greatly simplifies the use of the KDC model, as we will see in Section 2.2.1.

2.1.2 KDC model Hamiltonian

The KDC model has enjoyed great success in describing vibronic coupling effects over the last 30 years [32, 95, 76, 88]. If we consider again the simplest case of two strongly coupled electronic states, labeled a and b, the full molecular wave functions are taken to be linear combinations of the coupled quasidiabatic electronic wave functions:

$$\Psi_A(\mathbf{r}, \mathbf{q}) = \psi_a^{\mathrm{QD}}(\mathbf{r}; \mathbf{q}) \Omega_a^{(A)}(\mathbf{q}) + \psi_b^{\mathrm{QD}}(\mathbf{r}; \mathbf{q}) \Omega_b^{(A)}(\mathbf{q}). \tag{2.6}$$

The nuclear Schrödinger equation (the solutions of which are the nuclear wave functions $\Omega_i^{(A)}(\mathbf{q})$ and energy levels E) is then written as

$$\begin{aligned}
&\begin{bmatrix} \hat{H}_{aa} & \hat{H}_{ab} \\ \hat{H}_{ba} & \hat{H}_{bb} \end{bmatrix} \begin{bmatrix} \Omega_a(\mathbf{q}) \\ \Omega_b(\mathbf{q}) \end{bmatrix} \\
&\quad = \left(\begin{bmatrix} \hat{T}_n & 0 \\ 0 & \hat{T}_n \end{bmatrix} + \begin{bmatrix} \langle \psi_a^{\mathrm{QD}} | \, \hat{H}_e \, | \psi_a^{\mathrm{QD}} \rangle & \langle \psi_a^{\mathrm{QD}} | \, \hat{H}_e \, | \psi_b^{\mathrm{QD}} \rangle \\ \langle \psi_b^{\mathrm{QD}} | \, \hat{H}_e \, | \psi_a^{\mathrm{QD}} \rangle & \langle \psi_b^{\mathrm{QD}} | \, \hat{H}_e \, | \psi_b^{\mathrm{QD}} \rangle \end{bmatrix} \right) \begin{bmatrix} \Omega_a(\mathbf{q}) \\ \Omega_b(\mathbf{q}) \end{bmatrix} \\
&\quad = E \begin{bmatrix} \Omega_a(\mathbf{q}) \\ \Omega_b(\mathbf{q}) \end{bmatrix}.
\end{aligned} \tag{2.7}$$

Each of the quasidiabatic potential matrix elements, $\langle \psi_i^{\mathrm{QD}} | \, \hat{H}_e \, | \psi_{i'}^{\mathrm{QD}} \rangle$, can be expanded using a Taylor series expansion around the absorbing state equilibrium geometry (which we define as $\mathbf{q} = 0$):

$$\begin{aligned}
\langle \psi_i^{\mathrm{QD}} | \, \hat{H}_e \, | \psi_{i'}^{\mathrm{QD}} \rangle = \delta_{ii'} E(0)_i &+ \sum_j \left(\frac{\partial}{\partial q_j} \langle \psi_i^{\mathrm{QD}} | \, \hat{H}_e \, | \psi_{i'}^{\mathrm{QD}} \rangle \right)_{\mathbf{q}=0} q_j \\
&+ \frac{1}{2} \sum_{jk} \left(\frac{\partial^2}{\partial q_j \partial q_k} \langle \psi_i^{\mathrm{QD}} | \, \hat{H}_e \, | \psi_{i'}^{\mathrm{QD}} \rangle \right)_{\mathbf{q}=0} q_j q_k \\
&+ \frac{1}{6} \sum_{jkl} \left(\frac{\partial^3}{\partial q_j \partial q_k \partial q_l} \langle \psi_i^{\mathrm{QD}} | \, \hat{H}_e \, | \psi_{i'}^{\mathrm{QD}} \rangle \right)_{\mathbf{q}=0} q_j q_k q_l \\
&+ \frac{1}{24} \sum_{jklm} \left(\frac{\partial^4}{\partial q_j \partial q_k \partial q_l \partial q_m} \langle \psi_i^{\mathrm{QD}} | \, \hat{H}_e \, | \psi_{i'}^{\mathrm{QD}} \rangle \right)_{\mathbf{q}=0} q_j q_k q_l q_m + \cdots,
\end{aligned} \tag{2.8}$$

where $E(0)_a$ is the energy difference between the potential energy surface for electronic state a and the absorbing electronic state potential energy surface, at the geometry $\mathbf{q} = 0$.

In practice, the Taylor series expansion in equation 2.8 must be truncated. Different levels of truncation result in different levels of accuracy of the model.

Models that use only first-order terms in the Taylor series expansions (and second-order terms for diagonal entries) are called linear vibronic coupling (LVC) models. Models that use up to second-order terms are called quadratic vibronic coupling (QVC) models. While LVC and QVC models are often all that are needed to accurately compute vibronic energy levels, some systems require up to fourth-order terms [51] and, in extreme cases, up to sixth-order terms to produce an accurate model.

2.2 Solving the Schrödinger equation with the KDC model Hamiltonian

To solve for the stick spectrum (energy levels and transition intensities) of a system described by the KDC Hamiltonian, the nuclear Schrödinger equation in equation 2.7 must be solved, where the nuclear kinetic energy operator takes the form

$$\hat{T}_n = \frac{1}{2}\sum_j \omega_j p_j^2, \tag{2.9}$$

and the quasidiabatic potential matrix elements are represented by the Taylor series polynomial

$$\begin{aligned}\langle \psi_i^{\mathrm{QD}} | \, \hat{H}_e \, | \psi_{i'}^{\mathrm{QD}} \rangle = \delta_{ii'} E(0)_i + \sum_j F_j^{ii'} q_j + \frac{1}{2}\sum_{jk} F_{jk}^{ii'} q_j q_k \\ + \frac{1}{6}\sum_{jkl} F_{jkl}^{ii'} q_j q_k q_l \qquad (2.10) \\ + \frac{1}{24}\sum_{jklm} F_{jklm}^{ii'} q_j q_k q_l q_m + \cdots . \qquad (2.11)\end{aligned}$$

The variables $F^{ii'}_{jk\ldots}$ are used as a shorthand notation for the quasidiabatic potential matrix derivatives.

The nuclear Schrödinger equation (equation 2.7) can be solved approximately using a variational method called the Rayleigh-Ritz method [84]. The Rayleigh-Ritz method begins by assuming that the nuclear wave functions, $\Omega_i(\mathbf{q})$, can be represented as a linear combination of a finite number of known, orthogonal, basis functions $\{\phi_j\}$:

$$\Omega_i(\mathbf{q}) = \sum_{j=1}^{N} c_j \phi_j(\mathbf{q}) \tag{2.12}$$

where

$$\langle \phi_j | \phi_k \rangle = \delta_{jk}. \tag{2.13}$$

In theory, the basis used to represent the nuclear wave function for each electronic state can differ, but in this work (and in most other related works), the same basis is used for each nuclear wave function.

Each block of the Hamiltonian, $\hat{H}_{ii'}$, is then projected onto the known basis, forming submatrices $\mathbf{H}_{ii'}$ with entries

$$\mathbf{H}_{ii',jk} = \langle \phi_j | \, \hat{H}_{ii'} \, | \phi_k \rangle \tag{2.14}$$

which, together, create an $nstate \times nstate$ block matrix, $\mathbf{H}$, where $nstate$ is the number of coupled electronic states considered in the KDC model Hamiltonian.

The eigenvalues of the resulting matrix are approximations to the eigenvalues of the model Hamiltonian. The eigenvectors of the Hamiltonian matrix are expansion coefficients c_j used to define nuclear wave functions in terms of the known basis functions (equation 2.12).

For the solution of the nuclear Schrödinger equation in 2.7, the most common choice of basis $\{\phi_j\}$ to represent the nuclear wave functions Ω_i is one comprised of the vibrational wave functions of the absorbing electronic state under the harmonic approximation. The basis $\{\phi_j\}$, then, is the basis of an m-dimensional harmonic oscillator (where m is the number of normal modes in the system) defined by the absorbing state potential energy surface and centered at the absorbing state equilibrium geometry. Our nuclear wave functions then have the form

$$\Omega_i(\mathbf{q}) = \sum_{\vec{v}} c^i_{\vec{v}} \chi_{\vec{v}}(\mathbf{q}). \tag{2.15}$$

As seen in Section 1.2.2.1, each m-dimensional harmonic oscillator function $\chi_{\vec{v}}(\mathbf{q})$ can be written as the product of m one-dimensional harmonic oscillator functions,

$$\chi_{\vec{v}}(\mathbf{q}) = \chi_{v_1}(q_1)\chi_{v_2}(q_2)\dots\chi_{v_m}(q_m). \tag{2.16}$$

For ease of notation, I will write each m-dimensional harmonic oscillator function as the collection of its quantum numbers:

$$|\vec{v}\rangle = |v_1 v_2 \dots v_m\rangle = |\chi_{v_1}(q_1)\chi_{v_2}(q_2)\dots\chi_{v_m}(q_m)\rangle\,. \tag{2.17}$$

The vibrational basis can then be written as

$$|\vec{v}\rangle = |v_1 v_2 \dots v_m\rangle \quad \text{where} \quad v_j \in \{0, 1, \dots\}. \tag{2.18}$$

In theory, the vibrational basis is of infinite size. In practice, however, to produce a matrix $\mathbf{H}$ of finite size, the basis must be truncated. I use variables n_j to set the maximum number of possible quanta for each one-dimensional harmonic oscillator function in the m-dimensional harmonic oscillator function:

$$|\vec{v}\rangle = |v_1 v_2 \dots v_m\rangle \quad \text{where} \quad v_j \in \{0, 1, \dots, n_j - 1\}. \tag{2.19}$$

2.2.1 The Hamiltonian matrix

To perform a variational calculation, each sub-operator, $\hat{H}_{ii'}$, of the model Hamiltonian is projected onto the basis of an m-dimensional harmonic oscillator (equation 2.16). Projected in this basis, the Hamiltonian operator becomes a blocked matrix $\mathbf{H}$, where each block $\mathbf{H}_{ii'}$ has entries

$$\begin{aligned}
\mathbf{H}_{ii',uv} &= \langle\vec{u}|\,\delta_{ii'}(\hat{T}_n) + \langle\psi_i^{\text{QD}}|\,\hat{H}_e\,|\psi_{i'}^{\text{QD}}\rangle\,|\vec{v}\rangle \\
&= \langle\vec{u}|\,\delta_{ii'}(\frac{1}{2}\sum_j \omega_j p_j^2 + E(0)_i) + \sum_j F_j^{ii'} q_j \\
&+ \frac{1}{2}\sum_{jk} F_{jk}^{ii'} q_j q_k + \frac{1}{6}\sum_{jkl} F_{jkl}^{ii'} q_j q_k q_l + \ldots\,|\vec{v}\rangle \\
&= \delta_{ii'}(\frac{1}{2}\sum_j \omega_j\,\langle\vec{u}|\,p_j^2\,|\vec{v}\rangle + \delta_{uv}(E(0)_j)) + \sum_j F_j^{ii'}\,\langle\vec{u}|\,q_j\,|\vec{v}\rangle \\
&+ \frac{1}{2}\sum_{jk} F_{jk}^{ii'}\,\langle\vec{u}|\,q_j q_k\,|\vec{v}\rangle + \frac{1}{6}\sum_{jkl} F_{jkl}^{ii'}\,\langle\vec{u}|\,q_j q_k q_l\,|\vec{v}\rangle + \cdots . \qquad (2.20)
\end{aligned}$$

The matrix entries of $\mathbf{H}_{ii'}$ can be computed using the following equations [105]: Let $w = \max(u_j, v_j)$. Then,

$$\langle\vec{u}|\,q_j\,|\vec{v}\rangle = \begin{cases} \left(\frac{w}{2}\right)^{\frac{1}{2}} & \text{if } u_j = v_j \pm 1 \\ & \text{and } u_k = v_k\ \forall k \neq j \\ 0 & \text{else} \end{cases}, \qquad (2.21)$$

$$\langle\vec{u}|\,q_j^2\,|\vec{v}\rangle = \begin{cases} \frac{1}{2}\left[(w)(w-1)\right]^{\frac{1}{2}} & \text{if } u_j = v_j \pm 2 \\ & \text{and } u_k = v_k\ \forall k \neq j \\ w + \frac{1}{2} & \text{if } u_j = v_j \pm 0 \\ & \text{and } u_k = v_k\ \forall k \neq j \\ 0 & \text{else} \end{cases}, \qquad (2.22)$$

$$\langle \vec{u}| \, q_j^3 \, |\vec{v}\rangle = \begin{cases} \left[\frac{w(w-1)(w-2)}{8}\right]^{\frac{1}{2}} & \text{if } u_j = v_j \pm 3 \\ & \text{and } u_k = v_k \, \forall k \neq j \\ 3\left[\frac{w^3}{8}\right]^{\frac{1}{2}} & \text{if } u_j = v_j \pm 1 \\ & \text{and } u_k = v_k \, \forall k \neq j \\ 0 & \text{else} \end{cases}, \tag{2.23}$$

$$\langle \vec{u}| \, q_j^4 \, |\vec{v}\rangle = \begin{cases} \frac{[w(w-1)(w-2)(w-3)]^{\frac{1}{2}}}{4} & \text{if } u_j = v_j \pm 4 \\ & \text{and } u_k = v_k \, \forall k \neq j \\ \frac{(2w-1)\left[(w-1)w\right]^{\frac{1}{2}}}{2} & \text{if } u_j = v_j \pm 2 \\ & \text{and } u_k = v_k \, \forall k \neq j \\ \frac{3(2w^2+2w+1)}{4} & \text{if } u_j = v_j \pm 0 \\ & \text{and } u_k = v_k \, \forall k \neq j \\ 0 & \text{else} \end{cases}, \tag{2.24}$$

and

$$\langle \vec{u}| \, p_j^2 \, |\vec{v}\rangle = \begin{cases} -\frac{1}{2}\left[(w)(w-1)\right]^{\frac{1}{2}} & \text{if } u_j = v_j \pm 2 \\ & \text{and } u_k = v_k \, \forall k \neq j \\ w + \frac{1}{2} & \text{if } u_j = v_j \pm 0 \\ & \text{and } u_k = v_k \, \forall k \neq j \\ 0 & \text{else} \end{cases}. \tag{2.25}$$

Terms involving products of different q's are computed similarly. For example, let $w_j = \max(u_j, v_j)$ and $w_k = \max(u_k, v_k)$. Then,

$$\langle \vec{u}| \, q_j q_k \, |\vec{v}\rangle = \begin{cases} \left(\frac{w_j}{2}\right)^{\frac{1}{2}} \left(\frac{w_k}{2}\right)^{\frac{1}{2}} & \text{if } u_j = v_j \pm 1 \text{ and } u_k = v_k \pm 1 \\ & \text{and } u_l = v_l \, \forall l \neq j,k \\ 0 & \text{else} \end{cases}. \tag{2.26}$$

The matrix **H** will be quite sparse, since, for a fixed vibrational basis function $|\vec{u}\rangle$, equations 2.21-2.26 are only non-zero for a relatively small number of functions,

$\{|\vec{v}\rangle\}$(this corresponds to "for a fixed row of the matrix **H**, matrix entries are non-zero for a relatively small number of columns").

Looking back at equation 2.5, the change from normal coordinates to dimensionless normal coordinates is motivated by equations 2.21-2.24. If the Taylor expansion in equation 2.11 is calculated in terms of normal coordinates Q_i, then computing the entries of **H** would require the calculation of the values

$$\langle\vec{u}| Q_j |\vec{v}\rangle , \langle\vec{u}| Q_j^2 |\vec{v}\rangle , etc., \tag{2.27}$$

where $\langle\vec{u}|$ and $|\vec{v}\rangle$ are harmonic oscillator functions in terms of normal coordinates Q_i (equation 1.54). The terms in equation 2.27 do not depend solely on the quantum numbers of $\langle\vec{u}|$ and $|\vec{v}\rangle$ (like equations 2.21-2.24), but rather additionally depend on the normal-mode frequencies ω_i. This additional dependency complicates the calculation of the matrix entries, so we transform to dimensionless normal coordinates to avoid this inconvenience.

2.2.2 Solving for the eigenvalues of the Hamiltonian matrix

To compute approximations to the energy levels of the KDC model Hamiltonian, we must solve for the eigenvalues of **H**. To this end, I use an iterative eigensolver called the Lanczos algorithm. The Lanczos algorithm is appropriate because

1. **H** is a sparse matrix, which suggests the use of an iterative method rather than a direct solver.

2. **H** is symmetric (this can be seen from equations 2.21-2.26 and the definition of w in those equations).

3. We want to approximately solve for a large number of eigenvalues (ideally the entire spectrum of **H**). The Lanczos algorithm provides (after enough iterations) an approximation to the entire spectrum of a matrix. This is in contrast to other iterative methods that might only solve for a single eigenvalue at a time.

The Lanczos algorithm starts with a properly chosen starting vector v_0, and iteratively builds an orthogonal basis $\mathbf{V}_j$ of the Krylov subspace,

$$\mathcal{K}^j(\mathbf{H}, v_0) = \text{span}\{v_0, \mathbf{H}v_0, \mathbf{H}^2 v_0, \ldots, \mathbf{H}^{j-1} v_0\}, \tag{2.28}$$

one column at a time, such that, in the orthogonal basis $\mathbf{V}_j$, the matrix **H** is represented by a real symmetric tridiagonal matrix:

$$\mathbf{T}_j = \mathbf{V}_j^* \mathbf{H} \mathbf{V} = \begin{bmatrix} \alpha_1 & \beta_1 & & \\ \beta_1 & \alpha_2 & \ddots & \\ & \ddots & \ddots & \beta_{j-1} \\ & & \beta_{j-1} & \alpha_j \end{bmatrix}. \tag{2.29}$$

The procedure for computing the vectors v_j and the resulting tridiagonal representation of **H** (the values α_j and β_j) is described in Algorithm 5. At any iteration j, the eigenpairs ($\theta_i^{(j)}$, $s_i^{(j)}$) of $\mathbf{T}_j$ can be computed. The eigenvalues $\theta_i^{(j)}$ are called Ritz values and approximate eigenvalues of **H**. Similarly, the vectors $x_i^{(j)} = \mathbf{V}_j s_i^{(j)}$ are called Ritz vectors and approximate eigenvectors of **H**.

Algorithm 5 Lanczos algorithm (adapted from [2])

start with $r = v_0$, starting vector
$\beta_0 = ||r||_2$
for $j = 1, 2, ..., N$ **do**
    $v_j = r/\beta_{j-1}$
    operate $r = \mathbf{H}v_j$
    $r = r - v_{j-1}\beta_{j-1}$
    $\alpha_j = v_j^* r$
    reorthogonalize if necessary
    $\beta_j = ||r||_2$
end for
compute approximate eigenvalues $T_j = S\Theta^{(N)}S^*$

2.2.3 Computing transition intensities for the vibronic energies

In addition to solving for the eigenvalues of **H** (which approximate the energy levels of the system), we must also solve for the transition intensities associated with each eigenvalue (energy level). Each eigenstate of **H** represents a possible excited state to which a molecule may transition. Let us label the initial state from which the molecule will absorb radition as $\Psi_X(\mathbf{r}, \mathbf{q})$. Let us label some other attainable excited state as $\Psi_A(\mathbf{r}, \mathbf{q})$. The intensity of a transition from state X to state A is given by $|P_{XA}|^2$, where P_{XA} is the *transition probability* and takes the form

$$P_{XA} = \langle \Psi_X | \, \boldsymbol{\mu} \, | \Psi_A \rangle \, . \tag{2.30}$$

The term $\boldsymbol{\mu}$ is the molecular dipole operator, which is determined by the charge and locations of the electrons and the nuclei:

$$\boldsymbol{\mu} = \boldsymbol{\mu}_e + \boldsymbol{\mu}_n = -e \sum_i \mathbf{r}_i + e \sum_j Z_j \mathbf{R}_j. \tag{2.31}$$

While the definition of the dipole operator above depends on Cartesian nuclear coordinates, it can be rewritten in terms of dimensionless normal coordinates such that the separation of $\boldsymbol{\mu}$ into $\boldsymbol{\mu}_e$ and $\boldsymbol{\mu}_n$ is still possible, and this separation is the only property of $\boldsymbol{\mu}$ necessary for the following derivations.

We can simplify the form of P_{XA} by writing the states $\Psi_X(\mathbf{r},\mathbf{q})$ and $\Psi_A(\mathbf{r},\mathbf{q})$ in the quasidiabatic basis used in the KDC model. For the systems we wish to study in this work, the absorbing state is energetically distant from all other electronic states, and can thus be described by the product of a single electronic state and vibrational wave function:

$$\Psi_X(\mathbf{r},\mathbf{q}) = \psi_x^{\mathrm{QD}}(\mathbf{r};\mathbf{q})\Omega_x(\mathbf{q}). \tag{2.32}$$

If we assume that we are only considering two vibronically coupled electronic states (a and b), the vibronic states Ψ_A take the form

$$\Psi_A(\mathbf{r},\mathbf{q}) = \psi_a^{\mathrm{QD}}(\mathbf{r};\mathbf{q})\Omega_a^{(A)}(\mathbf{q}) + \psi_b^{\mathrm{QD}}(\mathbf{r};\mathbf{q})\Omega_b^{(A)}(\mathbf{q}). \tag{2.33}$$

Substituting equations 2.32 and 2.33 into equation 2.30, we can write P_{XA} as

$$\begin{aligned}
P_{XA} &= \langle\Psi_X|\,\boldsymbol{\mu}\,|\Psi_A\rangle && (2.34)\\
&= \langle\Psi_X|\,\boldsymbol{\mu}_e + \boldsymbol{\mu}_n\,|\Psi_A\rangle && (2.35)\\
&= \langle\psi_x^{\mathrm{QD}}\Omega_x|\,\boldsymbol{\mu}_e + \boldsymbol{\mu}_n\,|\psi_a^{\mathrm{QD}}\Omega_a^{(A)} + \psi_b^{\mathrm{QD}}\Omega_b^{(A)}\rangle && (2.36)\\
&= \langle\psi_x^{\mathrm{QD}}\Omega_x|\,\boldsymbol{\mu}_e\,|\psi_a^{\mathrm{QD}}\Omega_a^{(A)} + \psi_b^{\mathrm{QD}}\Omega_b^{(A)}\rangle \\
&\quad + \langle\psi_x^{\mathrm{QD}}\Omega_x|\,\boldsymbol{\mu}_n\,|\psi_a^{\mathrm{QD}}\Omega_a^{(A)} + \psi_b^{\mathrm{QD}}\Omega_b^{(A)}\rangle && (2.37)\\
&= \langle\psi_x^{\mathrm{QD}}\Omega_x|\,\boldsymbol{\mu}_e\,|\psi_a^{\mathrm{QD}}\Omega_a^{(A)}\rangle + \langle\psi_x^{\mathrm{QD}}\Omega_x|\,\boldsymbol{\mu}_e\,|\psi_b^{\mathrm{QD}}\Omega_b^{(A)}\rangle
\end{aligned}$$

$$+ \langle \psi_x^{\mathrm{QD}} \Omega_x | \, \boldsymbol{\mu}_n \, | \psi_a^{\mathrm{QD}} \Omega_a^{(A)} \rangle + \langle \psi_x^{\mathrm{QD}} \Omega_x | \, \boldsymbol{\mu}_n \, | \psi_b^{\mathrm{QD}} \Omega_b^{(A)} \rangle \tag{2.38}$$

$$= \langle \psi_x^{\mathrm{QD}} \Omega_x | \, \boldsymbol{\mu}_e \, | \psi_a^{\mathrm{QD}} \Omega_a^{(A)} \rangle + \langle \psi_x^{\mathrm{QD}} \Omega_x | \, \boldsymbol{\mu}_e \, | \psi_b^{\mathrm{QD}} \Omega_b^{(A)} \rangle$$
$$+ \underbrace{\langle \psi_x^{\mathrm{QD}} | \psi_a^{\mathrm{QD}} \rangle}_{0} \langle \Omega_x | \, \boldsymbol{\mu}_n \, | \Omega_a^{(A)} \rangle + \underbrace{\langle \psi_x^{\mathrm{QD}} | \psi_b^{\mathrm{QD}} \rangle}_{0} \langle \Omega_x | \, \boldsymbol{\mu}_n \, | \Omega_b^{(A)} \rangle \tag{2.39}$$

due to the orthogonality of the quasidiabatic electronic wave functions,

$$= \langle \psi_x^{\mathrm{QD}} | \, \boldsymbol{\mu}_e \, | \psi_a^{\mathrm{QD}} \rangle \, \langle \Omega_x | \Omega_a^{(A)} \rangle + \langle \psi_x^{\mathrm{QD}} | \, \boldsymbol{\mu}_e \, | \psi_b^{\mathrm{QD}} \rangle \, \langle \Omega_x | \Omega_b^{(A)} \rangle \tag{2.40}$$

by the same approximate integral separation described under equation 1.73.

In our variational calculations, the vibrational wave functions Ω_x, Ω_a, and Ω_b are all taken to be linear combinations of orthogonal harmonic oscillator functions defined in the absorbing electronic state (equation 2.15). As a result, we can compute the inner product of Ω_x and $\Omega_a^{(A)}$ ($\langle \Omega_x | \Omega_a^{(A)} \rangle$) by taking the inner product of the two vectors $\{c_m^{(x)}\}$ and $\{c_m^{(A;a)}\}$ of coefficients of their respective harmonic oscillator basis expansions. Similarly, we can compute $\langle \Omega_x | \Omega_b^{(A)} \rangle$ by taking the inner product of the two vectors $\{c_m^{(x)}\}$ and $\{c_m^{(A;b)}\}$. The transition moments $\langle \psi_x^{\mathrm{QD}} | \, \boldsymbol{\mu}_e \, | \psi_a^{\mathrm{QD}} \rangle$ and $\langle \psi_x^{\mathrm{QD}} | \, \boldsymbol{\mu}_e \, | \psi_b^{\mathrm{QD}} \rangle$ are calculated separately from the Lanczos algorithm.

The eigenvector of $\mathbf{H}$ corresponding to vibronic state A is a concatenation of the expansion coefficients $\{c_m^{(A;a)}\}$ and $\{c_m^{(A;b)}\}$. The transition probability for the transition from vibronic state X to vibronic state A can be computed by taking the inner product between the vector

$$t = \begin{bmatrix} \langle \psi_x^{\mathrm{QD}} | \, \boldsymbol{\mu}_e \, | \psi_a^{\mathrm{QD}} \rangle \, c^{(X)} \\ \langle \psi_x^{\mathrm{QD}} | \, \boldsymbol{\mu}_e \, | \psi_b^{\mathrm{QD}} \rangle \, c^{(X)} \end{bmatrix}, \tag{2.41}$$

which is a concatenation of the vectors $\langle\psi_x^{\text{QD}}|\,\boldsymbol{\mu}_e\,|\psi_a^{\text{QD}}\rangle\,c^{(X)}$ and $\langle\psi_x^{\text{QD}}|\,\boldsymbol{\mu}_e\,|\psi_b^{\text{QD}}\rangle\,c^{(X)}$, and the eigenvector of $\mathbf{H}$ corresponding to state A. When viewing transition probability calculations in this way, it may seem that the computation of transition intensities will require explicit computation of the eigenvectors of the Hamiltonian matrix $\mathbf{H}$. Computing all of the eigenvectors of $\mathbf{H}$, though, requires significantly more computation and memory usage than computing the eigenvalues alone, and should be avoided if possible. Fortunately, the mathematics behind the Lanczos algorithm allows for a way to compute transition intensities without explicitly computing or storing any eigenvectors.

After j iterations of the Lanczos algorithm, the vectors $x_i^{(j)} = \mathbf{V}_j s_i^{(j)}$ are approximations to eigenvectors of $\mathbf{H}$, where $\mathbf{V}_j$ is an orthogonal basis for a Krylov subspace, $s_i^{(j)}$ is the i^{th} eigenvector of $\mathbf{T}_j$, and $\mathbf{T}_j = \mathbf{V}_j^* \mathbf{H} \mathbf{V}_j$ is the projection of $\mathbf{H}$ onto the basis $\mathbf{V}_j$. To compute a transition probability for the transition from an absorbing state to a state described by $x_i^{(j)}$, we can compute the inner product of the vector t defined in 2.41 and the eigenvector $x_i^{(j)}$.

Let the vector t be the initial vector input into the Lanczos algorithm, so that t will be the first column, v_0, of the orthogonal basis $\mathbf{V}_j$. Then, the transition probabilities can be computed by

$$P = t' x_i^{(j)} = v_0' x_i^{(j)} \tag{2.42}$$

$$= v_0' \mathbf{V}_j s_i^{(j)} \tag{2.43}$$

$$= v_0' \left[v_0 | v_1 | v_2 | \ldots | v_j\right] s_i^{(j)} \tag{2.44}$$

$$= v_0' \left(\sum_{k=0}^{j} v_k s_{i,k}^{(j)} \right) \tag{2.45}$$

$$= \sum_{k=0}^{n} (v_0' v_k) s_{i,k}^{(j)} \tag{2.46}$$

$$= s_{i,0}^{(j)}, \tag{2.47}$$

since $v_i' v_j = \delta_{ij}$. So, if the vector t is used as the starting vector in the Lanczos algorithm, the transition intensities for each state $x_i^{(j)}$ can be computed by squaring the first entry of the eigenvector $s_i^{(j)}$ of $\mathbf{T}_j$.

2.3 Parallelization of the Lanczos algorithm

The total size of the Hamiltonian matrix, **H**, is equal to the product of the number of vibronically coupled states in the model and the size of the harmonic oscillator basis used to describe each vibrational wave function. The size of the Hamiltonian matrix, then, can be computed by

$$\text{number of states} \times \prod_{j=1}^{\text{number of modes}} n_j, \tag{2.48}$$

which grows exponentially with the number of vibrational modes. As a result, even calculations for relatively small systems, such as methane (CH_4) [108, 107], quickly become major computational problems. For example, an appropriate KDC model for CH_4 accounts for six vibronically coupled electronic states and nine $(3(5) - 6)$ vibrational normal modes. With this model, even a modest vibrational basis set of 10 functions per mode ($n_j = 10$) yields a Hamiltonian matrix of size

$$\text{(6 electronic states)*(10 functions)\^{}(9 normal modes)} = \mathbf{6 \times 10^9},$$

the storage of which requires vastly more memory than is available on a typical workstation. For larger molecules, it is not uncommon for matrix sizes to reach up

to 100 billion and beyond.

When the matrix **H** becomes too large, its matrix entries cannot be explicitly stored in memory. Although **H** is quite sparse, even storing only the non-zero matrix entries is intractable for large problems.[1] As a result, it is common for implementations of the Lanczos/KDC method to use *matrix-free* matrix vector multiplication operations. In a matrix-free matrix-vector multiplication routine, matrix entries are computed as they are needed and are not stored in memory. Because this approach requires many computations and re-computations of matrix entries, it is only really practical when matrix entries can be computed very quickly. The matrix **H**, then, is a good candidate for matrix-free routines, since its entries are computed using equations 2.21-2.26, which are simple functions of the quantum numbers that index the vibrational basis. Even when using a matrix-free approach, though, there are still significant computational stresses that arise from the large size of **H**:

1. The amount of memory required to perform a calculation can still be too large for a single machine. Even if matrix entries are not stored in memory, the three vectors used in the Lanczos algorithm must be stored. For the CH_4 example introduced earlier, storing these three vectors, each of size $6 * 10^9$, requires 144 GB of memory, which can exceed the memory limits of typically available workstations.

2. In addition to requiring significant amounts of memory, the KDC/Lanczos

[1] Using a fourth-order KDC model and the vibrational basis discussed for the CH_4 example, storing the non-zero matrix entries would require (5784 non-zeros per row * 6 billion rows * 8 bytes (double precision)) = 277.6 Terabytes of memory.

procedure can also require a large amount of time to complete. The amount of time needed to perform a matrix-vector multiplication grows approximately linearly with the size of **H**, and even the simplest LVC model can take several days to run the (often) required $\sim 10^3$ Lanczos iterations for a basis of size $6 * 10^9$. As the order of the polynomial representation of the potential increases, the matrix **H** becomes more dense,[2] and the time required for the calculation grows even longer.

Thus, it is important that we develop improved methods that allow us to more easily utilize the KDC/Lanczos approach for very large systems.

Schuurman *et al.* [90] were the first to propose a parallelization of the KDC/Lanzcos procedure. The utilization of large-scale computing clusters would allow us to distribute the memory requirements and workload of the algorithm across multiple machines, opening the doors to the study of larger molecules. In their work, Schuurman *et al.* describe an "open-ended" solution to the parallelization of the KDC/Lanczos algorithm using the Global Arrays (GA) toolkit [75]. There is reason to believe, though, that their solution may not be the most efficient approach to the parallelization of the KDC/Lanczos problem.

The Global Arrays toolkit is most appropriate for use with problems in which communication patterns are irregular or complicated [74]. If communication patterns are simple, using the GA toolkit will create unnecessary communication

[2] A single-state system with nine normal modes (assuming all $F^{ii'}_{jkl\ldots}$ terms are non-zero) will have a maximum of 19 non-zero entries per row using an LVC model, 181 non-zero entries per row using QVC, 1033 using up to cubic terms, and 5785 using up to quartic terms.

overhead, resulting in an inefficient parallelization. Later in this section, I will show that the KDC/Lanczos problem, distributed in a certain way, has a relatively simple nearest-neighbor communication pattern, suggesting that an alternative approach to parallelization may be preferable to using the Global Arrays toolkit.

In this section, I will introduce a new approach to parallelization of the KDC/Lanczos algorithm. I will discuss how data should be distributed among processes and how communication between processes should occur. I will discuss important details of my implementation such as message ordering and memory consumption. I will then theoretically and experimentally analyze the parallel efficiency of my approach to justify the decisions I've made in the development of my implementation.

2.3.1 Focus for parallelization

The Lanczos algorithm is comprised of only a handful of operations: Matrix-vector multiplication, vector inner product, vector norm, and vector update ($y = \alpha x + y$). The vector update is perfectly parallel (no communication required) and the vector norm and inner product are both easily parallelized. The application of a matrix-vector product for $\mathbf{H}$ (overwhelmingly the most costly step), then, should be the sole focus for parallelization. Because our implementation of the Lanczos procedure uses a matrix-free matrix vector multiplication routine, we cannot simply utilize existing frameworks for parallel sparse linear algebra, like PETSc[3] [3, 4, 5],

[3]Technically, I could use PETSc's shell matrix object to use PETSc for the surrounding iterative method. For this, I would have to supply my own parallel matrix-vector multiplication function

but rather we must develop our own parallel framework for this specific application.

2.3.2 Characterization of stored data

Before discussing the parallelization of the matrix-vector multiplication operation, it is important to understand how data is organized and stored within the algorithm. This information will help to determine appropriate partitionings of the stored data to distribute across processes. In the previous section, I mentioned that a matrix-free implementation of the matrix-vector multiplication operation is used to reduce memory consumption. As a result, the only data of non-negligible size that needs to be stored in memory are the vectors x and y involved in the operation $y = \mathbf{H}x$.

Remember that each index of a vector corresponds to an m-dimensional harmonic oscillator function. For illustration purposes, consider a harmonic oscillator basis for a system with three vibrational modes where each mode can have a maximum quantum number of three ($n_j = 4\ \forall j$). In my shorthand notation, this basis takes the form

$$|v_1 v_2 v_3\rangle \text{ where } v_j \in \{0, 1, 2, 3\}, \tag{2.49}$$

for a total of 64 functions. These functions correspond to vector indices in such a way that, starting at vector index 0, the quanta of v_1 are cycled over (from 0 to 3), then v_2, then v_3. The full set of vibrational basis functions and corresponding array indices is shown in Figure 2.1.

for the matrix-vector operation. Since the majority of the programming effort is within the matrix-vector multiplication operation, though, I did not feel that it was useful to use an external library

Array Index	Basis Function						
0	$\|000\rangle$	16	$\|001\rangle$	32	$\|002\rangle$	48	$\|003\rangle$
1	$\|100\rangle$	17	$\|101\rangle$	33	$\|102\rangle$	49	$\|103\rangle$
2	$\|200\rangle$	18	$\|201\rangle$	34	$\|202\rangle$	50	$\|203\rangle$
3	$\|300\rangle$	19	$\|301\rangle$	35	$\|302\rangle$	51	$\|303\rangle$
4	$\|010\rangle$	20	$\|011\rangle$	36	$\|012\rangle$	52	$\|013\rangle$
5	$\|110\rangle$	21	$\|111\rangle$	37	$\|112\rangle$	53	$\|113\rangle$
6	$\|210\rangle$	22	$\|211\rangle$	38	$\|212\rangle$	54	$\|213\rangle$
7	$\|310\rangle$	23	$\|311\rangle$	39	$\|312\rangle$	55	$\|313\rangle$
8	$\|020\rangle$	24	$\|021\rangle$	40	$\|022\rangle$	56	$\|023\rangle$
9	$\|120\rangle$	25	$\|121\rangle$	41	$\|122\rangle$	57	$\|123\rangle$
10	$\|220\rangle$	26	$\|221\rangle$	42	$\|222\rangle$	58	$\|223\rangle$
11	$\|320\rangle$	27	$\|321\rangle$	43	$\|322\rangle$	59	$\|323\rangle$
12	$\|030\rangle$	28	$\|031\rangle$	44	$\|032\rangle$	60	$\|033\rangle$
13	$\|130\rangle$	29	$\|131\rangle$	45	$\|132\rangle$	61	$\|133\rangle$
14	$\|230\rangle$	30	$\|231\rangle$	46	$\|232\rangle$	62	$\|233\rangle$
15	$\|330\rangle$	31	$\|331\rangle$	47	$\|332\rangle$	63	$\|333\rangle$

Figure 2.1: Vibrational basis functions mapped to array indices (3-mode system, 4 functions in each mode)

To help determine possible partitionings of the vector data, it is useful to visualize these vectors in terms of a *vector index domain*. A vector index domain is an m-dimensional hyper-rectangular grid in which each point corresponds to an m-dimensional harmonic oscillator function, which, in turn, corresponds to an array index. The vector index domain for our example three-dimensional system is shown in Figure 2.2.

Each dimension of the grid corresponds to a normal mode of vibration, and traversal along any dimension j increments or decrements the quanta of the harmonic oscillator function corresponding to the jth mode. The size of this domain is dependent on the truncation of the vibrational basis (the variables n_i). Dimension 1

(PETSc) for only a few inner product and vector update operations.

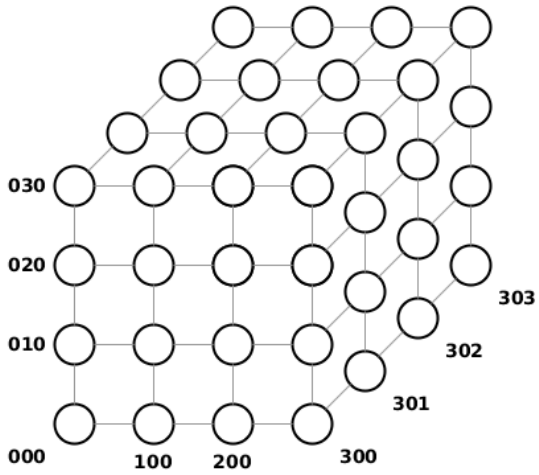


Figure 2.2: Three-dimensional representation of vector indices. Each grid point represents a member of the truncated vibrational basis.

of the domain has a length of n_1 points, dimension 2 has a length of n_2 points, and so on.

2.3.3 Distribution of data

To parallelize the matrix-vector multiplication, we must consider how vector data will be distributed amongst the available processes. Because single-machine memory constraints are one of the primary motivators for parallelization, I chose a partitioning scheme in which the vector data is distributed as evenly as possible across all processes with no overlap of information. Each process is assigned the same indices of x and y to store and compute (i.e. if Process A is assigned indices 0 through n, then it must locally store indices 0 through n of x and compute/store indices 0 through n of y).

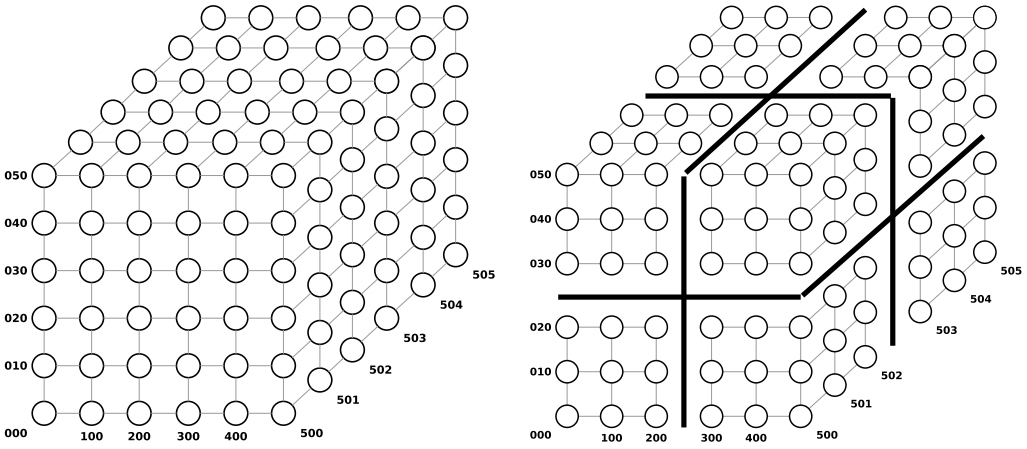


Figure 2.3: Left: Three-dimensional representation of vector indices. Each grid point represents a member of the truncated vibrational basis. Right: Example partitioning of a three-dimensional index domain.

Looking at Figure 2.2, any distribution of vector data will correspond to a partitioning of the grid domain of vibrational basis indices. In my parallel implementation, vectors are distributed by partitioning the m-dimensional index domain into m-dimensional hyper-rectangular sub-blocks. An example in three dimensions is shown in Figure 2.3. This partitioning scheme allows for the use of fixed index offsets to iterate over harmonic oscillator quanta for each normal mode, which greatly simplifies multi-dimensional indexing.

Even within this general partitioning scheme, though, there are many possible options for specific domain partitionings for any fixed number of processes. For example, a three-dimensional domain can be split amongst four processes by

1. partitioning any one dimension into four sections, or

2. bisecting any two of the three dimensions.

The specific choice of partitioning will affect certain properties of the parallel implementation, such as the amount of communication that occurs and load balancing. These effects will be studied in Section 2.4.

2.3.4 Communication of data

In addition to determining the distribution of data, we must determine how data will be communicated between the available processes. To determine communication patterns between processes, we need to understand the data dependencies involved in the matrix-vector multiplication, i.e., which entries of the input vector are needed to compute a given index of the output vector. This information is especially important in distributed-memory environments, since vector data are distributed across machines that do not have direct access all available memory in the environment. If Process A needs to compute an entry of y, the computation of which depends on an entry of x that is stored on Process B, then Process B must explicitly communicate the appropriate entry of x to Process A for Process A to complete its computation. Looking at the equation for a matrix-vector multiplication,

$$y_u = \sum_v \mathbf{H}_{uv} x_v, \tag{2.50}$$

we can see that the computation of any u^{th} entry of y depends on all v^{th} entries of x such that $\mathbf{H}_{uv}$ is non-zero.

2.3.4.1 Structure of Non-zero Entries

Let us look at some examples of where equations (2.21-2.24) produce non-zero entries in $\mathbf{H}$. Consider again the example vibrational basis defined in equa-

tion 2.49. By equations 2.21 and 2.20, any row $|v_1 v_2 v_3\rangle$ will have non-zero entries in columns $|(v_1 - 1)v_2 v_3\rangle$ and $|(v_1 + 1)v_2 v_3\rangle$, for any model in which $F_1^{ii'}$ is not zero. In our three-mode example system, contributions from the q_1 term give row 0 ($|000\rangle$) a non-zero entry in column 1 ($|100\rangle$), row 1 ($|100\rangle$) non-zeros in column 0 ($|000\rangle$) and column 2 ($|200\rangle$), row 2 ($|200\rangle$) non-zeros in column 1 ($|100\rangle$) and column 3 ($|300\rangle$), and so on. This pattern produces two band-like structures of non-zero entries in the matrix, shown in blue in Figure 2.4. Similar patterns for q_2 and q_3 terms produce the red and green (respectively) band-like non-zero structures seen in Figure 2.4. Similarly to q_j terms, q_j^2, q_j^3, and q_j^4 terms create non-zero bands that are 2, 3, and 4 times as far from the diagonal as q_j, respectively.

The breaks in the bands are caused by the necessary truncation of the number of harmonic oscillator functions representing each vibrational mode. For example, row 3 ($|300\rangle$) has a non-zero entry in column 2 ($|200\rangle$) but not in column 4 ($|010\rangle$). This is a result of truncating the number of functions representing mode 1 (v_1) to 4; the basis function $|400\rangle$ does not exist.

By equation 2.26, $q_j q_k$ terms each create 4 bands; e.g. from the $q_1 q_2$ term, row $|v_1 v_2 v_3\rangle$ has non-zeros in columns $|(v_1 + 1)(v_2 + 1)v_3\rangle$, $|(v_1 - 1)(v_2 + 1)v_3\rangle$, $|(v_1 + 1)(v_2 - 1)v_3\rangle$, and $|(v_1 - 1)(v_2 - 1)v_3\rangle$. Likewise, $q_j q_k q_l$ terms each create 8 bands and $q_j q_k q_l q_m$ terms each create 16 bands. The band structures for full (all force constants $F_{jk\ldots}^{ii'}$ are non-zero) LVC, QVC, cubic,[4] and quartic models are

[4]Use of a cubic vibronic coupling model is, of course, not a wise choice as it will result in a potential that is not bounded from below. However, this case is discussed here for clarity and completeness.

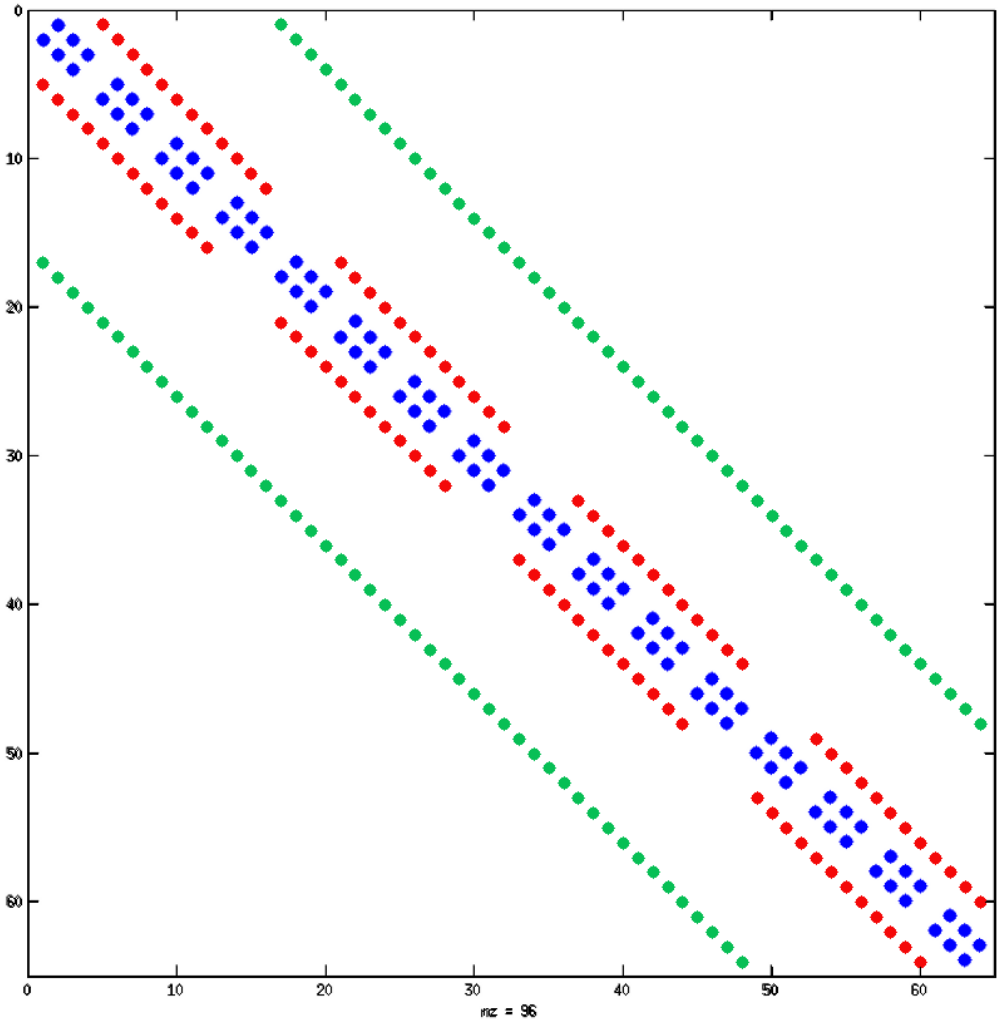


Figure 2.4: Non-zero bands created by first order q_j terms, given a three-mode system. Blue: q_1 Red: q_2 Green: q_3

shown in Figure 2.5.

From equations 2.21-2.26, it is clear that the non-zero entries of **H** have a well-defined structure. Any given row (indexed u) of **H** has non-zero entries in columns v such that v is close to u in quanta (the values v_i differ from u_i by small amounts). This nearest-neighbor dependency structure can be graphically represented in the form of a *stencil* to help visualize which data need to be communicated for a complete matrix-vector multiplication.

2.3.5 Stencils

Stencils are a common tool in numerical analysis, often used to describe relationships between unknowns in partial differential equations [20]. The stencils used in this chapter illustrate data dependencies involved in the matrix-vector multiplication operation. Stencils help to expose structure in the problem that can be used to understand communication patterns in a parallel implementation.

2.3.5.1 Introductory example

Assume that there exists a function, $f(x)$, that needs to be evaluated at all points in a discrete grid domain, and that the evaluation of this function depends on some collection of neighboring points. Consider a one-dimensional domain, x, and index points in this domain as $x[i]$. Define $f(x)$ to be

$$f(x[a]) = x[a-2] + x[a-1] - 2 * x[a] + x[a+1] + x[a+2]. \tag{2.51}$$

The function evaluation at any given point depends on the same range of neighboring points, regardless of the evaluation point; $f(x[a])$ depends on the values of two

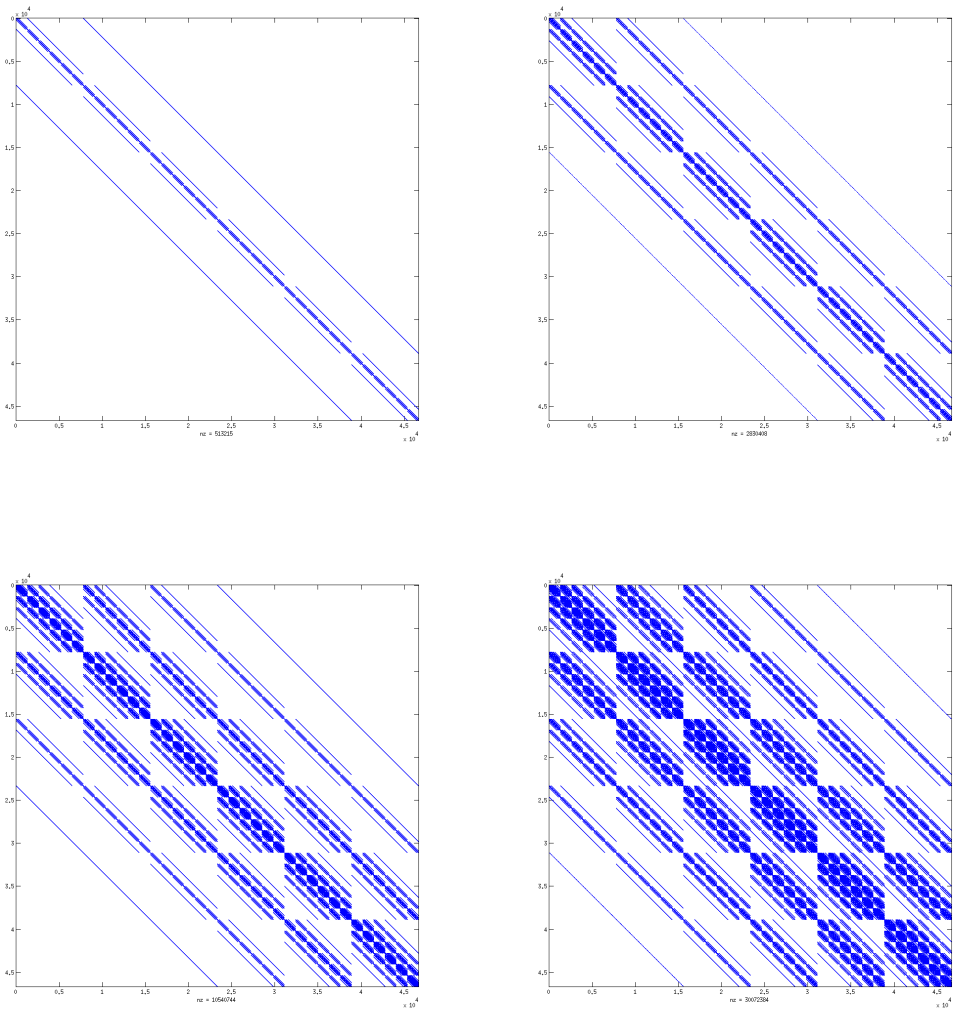


Figure 2.5: Sparsity pattern of **H** for linear, quadratic, cubic, and quartic truncations of the potential polynomial. Images come a 6-mode system with 6 functions representing each mode. Non-zero matrix entries are shown in blue.

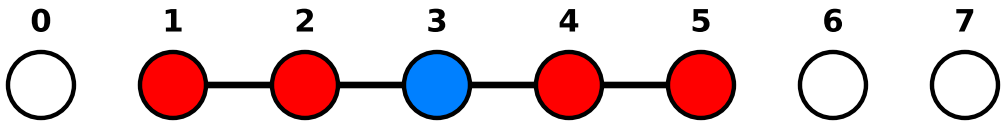


Figure 2.6: Stencil derived from equation 2.51. The center of the stencil is shown in blue. The red points are required for the computation of $f(x)$ at the blue point.

lower points ($x[a-1]$, $x[a-2]$) and two higher points ($x[a+1]$, $x[a+2]$) for all a. This dependency structure can be visualized by a stencil, shown in Figure 2.6. When overlaid atop a domain, a stencil shows dependencies for the evaluation of a function at any given point. The center of the stencil, shown in blue, is placed on the point ($x[a]$) at which the function is to be evaluated, and the nodes of the stencil, shown in red, lie on the points whose values are required for the evaluation of $f(x[a])$. The stencil in Figure 2.6 shows that, to evaluate $f(x[3])$, the values of x at indices $\{1, 2, 3, 4, 5\}$ are needed. A similar visualization can be created for the dependencies involved in the KDC matrix-vector product $y = \mathbf{H}x$ using the vector index domain. The center of the stencil will be placed on the index u at which we wish to compute $y[u]$. The nodes of the stencil will then lay atop indices v for which values $x[v]$ are needed to compute $y[u]$.

2.3.5.2 Stencil for $y = \mathbf{H}x$

Consider a system with three normal modes. The corresponding vibrational basis (and therefore the matrix **H**) is indexed by a vector of three integers, $\vec{u} = \{u_1, u_2, u_3\}$, representing the quantum numbers of each of the three harmonic

oscillator functions that make up a full vibrational basis function. By equation 2.21, the linear q_1 term of $\hat{H}$ causes the computation of y at index $\{u_1, u_2, u_3\}$ to depend on the values of x at indices

$$\vec{v} = \{u_1 + 1, u_2, u_3\} \text{ and } \vec{v} = \{u_1 - 1, u_2, u_3\}, \tag{2.52}$$

since the term $\langle \vec{u} | \, q_1 \, | \vec{v} \rangle$ is non-zero for the above values of $\vec{v}$. Similarly, the q_2 and q_3 terms of $\hat{H}$ cause the computation of y at index $\{u_1, u_2, u_3\}$ to depend on the values of x at indices

$$\vec{v} = \{u_1, u_2 + 1, u_3\} \text{ and } \vec{v} = \{u_1, u_2 - 1, u_3\}, \tag{2.53}$$

and

$$\vec{v} = \{u_1, u_2, u_3 + 1\} \text{ and } \vec{v} = \{u_1, u_2, u_3 - 1\}, \tag{2.54}$$

respectively.

The data dependencies described in equations 2.52-2.54 are encapsulated in the stencil shown on the top-left of Figure 2.7. A similar dependency structure can be composed for quadratic q_j^2 and $q_j q_k$ terms of $\hat{H}$, and the corresponding stencils are shown on the top-right and bottom row of Figure 2.7, respectively. To create a stencil for the full equation $y = \mathbf{H}x$, simply overlay the stencils corresponding to each individual term. A full stencil for up to quartic terms in a 2-D and 3-D domain is shown in Figure 2.8.

2.3.6 Determining data to be communicated

The vector entries that are required by Process A for its computation but are locally stored on a different Process B are called *ghost points*. Any Process B that

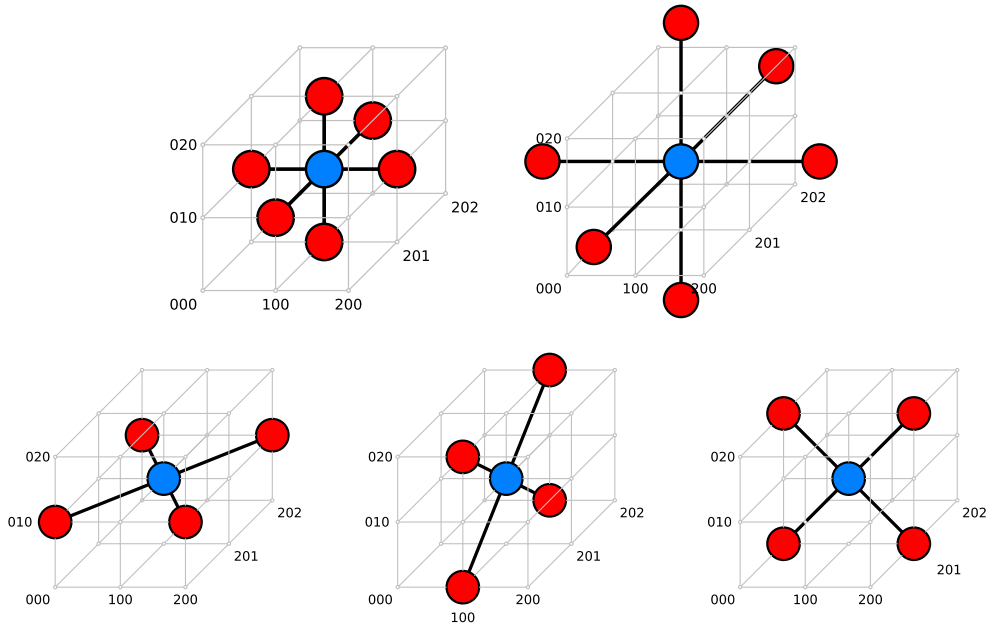


Figure 2.7: Three-dimensional representation of q_i (top-left), q_i^2 (top-right), and $q_i q_j$ (bottom row) stencils

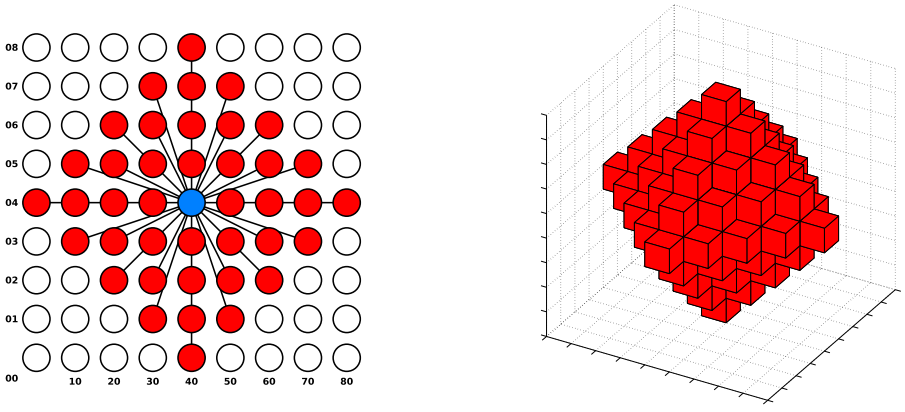


Figure 2.8: Full stencil using up to quartic terms in 2-dimensions (left) and 3-dimensions (right).

locally stores such ghost points is called a *neighboring process* of Process A. The term "neighbor" is used because dependencies $x[v]$ of a given output index $y[u]$ are such that v is close to u with respect to the hyper-rectangular index domain.

To determine the data that a given process needs to receive from other processes, a dependency stencil is centered, in the vector index domain, at all indices stored locally by that process. The union of stencil edges can be used to determine the total amount of non-locally-stored data needed by the process to complete the matrix-vector multiplication operation.

An example in two dimensions is shown in Figure 2.9. The dotted lines partition the full domain. Each process has a local hyper-rectangular (in this 2-D case, rectangular) domain of size 6×12. At the top of Figure 2.9, the quartic model stencil (Figure 2.8) is centered at two points belonging to Process 4. To compute the y index corresponding to the (0,0) point (top left) in Process 4's local domain, Process 4 needs x values belonging to Processes 0, 1, and 3 (and itself, but no communication is required to use locally-stored data). To compute the (5,9) point, Process 4 needs data from Processes 5, 7, and 8. To determine the non-locally-stored data needed to compute y at *all* of Process 4's local points, a stencil is centered at every point belonging to Process 4. The union of stencil nodes (shown at the bottom of Figure 2.9) represents the total data needed by Process 4 for its computation.

To continue to take advantage of fixed index offsets, I chose to communicate data between processes using hyper-rectangular buffers. These buffers are shown as thick black lines in Figure 2.9. The vector entries that need to be communicated

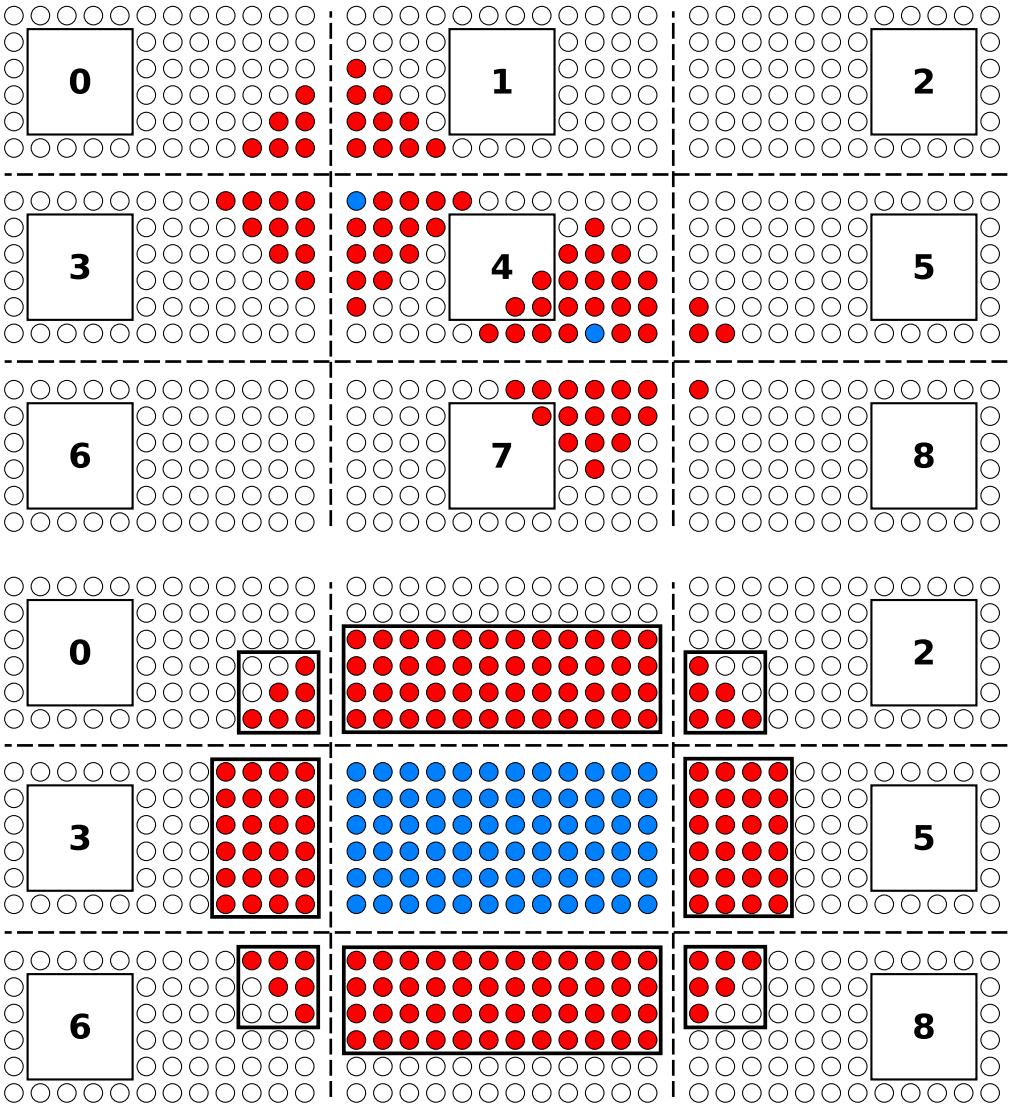


Figure 2.9: Graphical representation of the non-locally-stored data required by Process 4. The stencil for a quartic vibronic coupling model is used. Data is partitioned using dotted lines. The solid black lines surrounding the ghost points encompass the data that is actually communicated.

do not always form an exact hyper-rectangle, though, so processes actually communicate the smallest hyper-rectangular buffer that encapsulates the needed points. While this scheme can lead to unnecessary communication, it can be shown that, at most, $1.61\times$ the necessary amount of data is communicated when using quartic (or lower) order models,[5] and excess communication only occurs when using cubic (or higher) order models. Given the simplifications that accompany the use of hyper-rectangular buffers, I consider the cost of the extra communication to be small.[6]

Because all data buffers are treated as hyper-rectangles, any set of data to be communicated will correspond to an intersection of subsets of contiguous slices of the local domain along each dimension. For example, in Figure 2.9, Process 3 needs to communicate the intersection of rows [1:6] (all rows) and columns [9:12] of its local buffer to Process 4. Similarly, Process 0 needs to communicate the intersection of rows [4:6] and columns [10:12] of its local buffer to Process 4. Since each local index domain has a hyper-rectangular structure, the sets of data that need to be communicated are relatively easy to extract from the entire local domain.

[5]Excess communication for quartic models with m vibrational modes and b local harmonic oscillator functions for each mode can we computed by

$$\frac{4*2*\binom{m}{1}b^3+\mathbf{9}*4*\binom{m}{2}b^2+8*8*\binom{m}{3}b+16*\binom{m}{4}}{4*2*\binom{m}{1}b^3+\mathbf{6}*4*\binom{m}{2}b^2+4*8*\binom{m}{3}b+16*\binom{m}{4}}. \tag{2.55}$$

The bold 6 and 9 coefficients can be understood by looking at Figure 2.9. Note that a 3×3 $(=9)$ square of points is communicated, but only 6 points actually need to be communicated. Maximizing the above equation (for $b>4$ and $m>4$) gives a value of 1.61.

[6]In Section 2.5, I will show that, for cubic and higher-order models, the extra communication has little effect on the parallel scaling because the computation cost far outweighs the communication cost.

2.3.7 Memory usage during communication

When any process receives data from a neighboring process, it must store the received data in memory, at least temporarily, in order to use it to update its local buffer. It is important, then, to consider how each process should allocate memory to handle the reception of ghost points from its neighbors.

One approach is to allocate enough memory to store all incoming ghost points. The parallel algorithm would then go as follows:

1. Communicate with all neighboring processes, storing all ghost points simultaneously in local memory.

2. Perform the computation, using the stored ghost points as needed.

This approach, while relatively easy to implement, suffers from excess memory consumption. To illustrate this, consider a 9-mode KDC model using up to quartic terms. If the vibrational basis used has 10 functions in each mode ($n_i = 10$) and is distributed across 512 processes by bisecting each of the 9 dimensions, the number of ghost points for each process will be more than $25\times$ the number of locally-stored entries.[7] This means that a parallel matrix-vector product implementation

[7]The number of ghost points relative to the number of local points is calculated by the equation

$$\binom{9}{1} * \frac{4}{5} + \binom{9}{2} * \left(\frac{3}{5}\right)^2 + \binom{9}{3} * \left(\frac{2}{5}\right)^3 + \binom{9}{4} * \left(\frac{1}{5}\right)^4 = 25.73. \tag{2.56}$$

To explain the first term, there are $\binom{9}{1}$ neighbors to which a process sends $\frac{4}{5}$ of its local buffer (quartic stencils reach up to 4 points away from the center, and each local buffer has 9 dimensions, each of length 5).

that stores all ghost points simultaneously will require at least $25\times$ as much memory as a serial implementation. This is extremely excessive and effectively demands that another strategy be used.

To avoid excessive memory allocation, I have implemented a communication scheme in which communication occurs in *rounds*. In each round, processes send and receive data to/from *a single neighboring process* and use the received data to advance the matrix-vector multiplication operation. After the received data has been used to update a process's local output vector, the next round of communication begins, *reusing the data buffer* used in the previous round to receive new information.

This communication scheme requires the use of $(4/3)\times$ as much memory as a serial implementation for any number of processes (each process will have one extra memory buffer of size equal to the size of its local domain to receive data from neighboring processes, bringing the memory consumption to $4 \cdot$ (size of local domain), since the Lanczos algorithm requires the storage of three vectors by default.

2.3.8 Communication pattern

In setting up the communication, each process compiles a list of its neighboring processes to whom it will send information (called $\mathrm{send\_neighbors}$) and a list of its neighboring processes from whom it will receive information (called $\mathrm{recv\_neighbors}$). The two lists necessarily contain the same processes (due to symmetry properties of the stencil), but not necessarily in the same order. Over the

course of all rounds of communication, each process will iterate through both lists of neighbors and attempt to communicate (send or receive, depending on the list) with each one. Processes will not skip list entries; in a given round, if Process A is attempting to send information to Process B, and Process B is not available to receive information, then Process A will wait idly during subsequent rounds until Process B becomes available.

In this communication scheme, the ordering of the send_neighbors and recv_neighbors lists is extremely important. An inefficient ordering can cause certain processes to unnecessarily wait idly for its neighboring processes to become available for communication, lengthening the total communication time for all processes. To illustrate this point, let us look at an example. The communication patterns are easiest to understand in the case of a single-dimension partitioning of the vector index domain, so we will focus on 1-D partitioning for the time being.

Take, for example, a QVC model and a vibrational basis that contains 8 basis functions in one of its modes. Assume the total vibrational basis is partitioned along this mode and distributed amongst 8 processes. The result is 8 processes that communicate their entire local domains with neighboring processes according to the stencil shown in Figure 2.6 (Process 0 must send/receive information to/from Processes 1 and 2, Process 1 must send/receive information to/from Processes 0, 2, and 3, and so on.).

2.3.8.1 Sorted ordering scheme

Let us first look at an inefficient ordering for this problem. Assume that both of the neighboring process lists for each process are sorted with increasing rank (shown in Figure 2.10). This means that Process 0 will attempt to send *and* receive information to/from Process 1 before moving on to send *and* receive information to/from Process 2. Likewise, Process 1 will attempt to send and receive information to/from Process 0 before moving on to send and receive information to/from Process 2 before finally moving on to send and receive information to/from Process 3.

The resulting rounds of communication can be found in Figure 2.11. For each round, the communication occurring for a given Process p is described using two numbers, one on top of the other. The top number represents the rank of the process from whom Process p is receiving information, while the bottom number represents the rank of the process to whom Process p is sending information. Processes with blank spaces for a given round do not perform communication that round (either because they are waiting idly for a process to communicate with, or because they are done with all of their communication).

In the first round of communication, Processes 0 and 1 pair up and exchange information with each other. At the same time, Process 2 attempts to send/receive information to/from Process 0, but Process 0 is busy communicating with Process 1, so Process 2 must wait idly for Process 0 to become available in Round 2. Process 2's unfortunate situation is a common one, with Processes 3-7 also waiting idly for many rounds before communicating with others. The end result is that the

Ordering A:

p					p				
0	1	2			4	2	3	5	6
	1	2				2	3	5	6
1	0	2	3		5	3	4	6	7
	0	2	3			3	4	6	7
2	0	1	3	4	6	4	5	7	
	0	1	3	4		4	5	7	
3	1	2	4	5	7	5	6		
	1	2	4	5		5	6		

Figure 2.10: The send_neighbors and recv_neighbors lists produced by ordering processes in order of increasing rank. Each process p has two lists associated with it. The top list is recv_neighbors and the bottom list is send_neighbors

Communication rounds for ordering A:

p	1	2	3	4	5	6	7	8	9	10	11	12	13
0	1 1	2 2											
1	0 0		2 2	3 3									
2		0 0	1 1		3 3	4 4							
3				1 1	2 2		4 4	5 5					
4						2 2	3 3		5 5	6 6			
5								3 3	4 4		6 6	7 7	
6										4 4	5 5		7 7
7												5 5	6 6

Figure 2.11: The communication rounds produced by ordering A (Figure 2.10)

entire communication process (in which each process only has to communicate with at most 4 other processes) requires 13 rounds of communication because of an inefficient ordering of sends and receives. In general, for a 1-D process domain with P processes and a communication stencil that extends 2 processes in each direction, the communication ordering scheme just described requires $(2P - 3)$ rounds of communication (assuming $P \geq 4$).[8]

One can imagine that an "optimal" ordering of neighboring processes will complete the above example communication in 4 rounds (since the maximum number of neighbors that any one process has is 4).[9] In the general case, an "optimal" ordering will complete communication in R rounds, where R is the maximum number of neighbors for any one process. By this criteria, I have developed and implemented (in my program xsim_hpc) an ordering scheme that can be considered optimal, and I will describe this ordering scheme now.

2.3.8.2 The xsim_hpc ordering scheme

I will start by introducing some variables that are used in my program for the compilation of the neighbor lists:

number_of_processes_per_mode[] - an array that contains the number of

[8]This number comes from the fact that there are $4(P-4) + 2*3 + 2*2) = 4P - 6$ total messages to communicate (when $P \geq 4$), and, in each round, only two messages are completed; $(4P-6)/2 = 2P - 3$.

[9]To be precise, it is likely possible that some sort of recursive doubling algorithm could be used to reduce the number of rounds below 4, but this would only reduce latency costs, not bandwidth (which is the main bottleneck), and would require processes to store more than one buffer of ghost points simultaneously.

MPI processes partitioning each dimension of the index domain

local_rank[] - an array that contains a process's rank within a given partitioned dimension of the index domain (for a 1-D domain, this would just be the global rank)

For now, let us continue to assume that our process grid is one-dimensional (i.e. all processes partition the same dimension of the vector index domain). In this single-dimension example, both of the above variables are simply integers, rather than arrays of integers. The compilation of the send_neighbors and recv_neighbors lists then proceeds as follows:

Let i iterate from 1 to number_of_processes_per_mode

1. Select a potential neighbor to *send information to* with local rank equal to

$$(\text{local\_rank} + i) \pmod{\text{number\_of\_processes\_per\_mode}} \quad (2.57)$$

   and select a potential neighbor to *receive information from* with local rank equal to

$$(\text{local\_rank} - i) \pmod{\text{number\_of\_processes\_per\_mode}} \quad (2.58)$$

2. Test to see if either of the potential neighbors are actually reached by a stencil edge:

   (a) If neither of the two neighbors are reached, the potential neighbors are thrown out.

Ordering B:

p				
0			2	1
	1	2		
1	0		3	2
	2	3		0
2	1	0	4	3
	3	4	0	1
3	2	1	5	4
	4	5	1	2

p				
4	3	2	6	5
	5	6	2	3
5	4	3	7	6
	6	7	3	4
6	5	4		7
	7		4	5
7	6	5		
			5	6

Figure 2.12: The send_neighbors and recv_neighbors lists produced by the ordering scheme described in Section 2.3.8.2. Each process p has two lists associated with it. The top list is recv_neighbors and the bottom list is send_neighbors

(b) If one of the two neighbors is reached (but not both), the neighbor that is reached is added to its respective list and a "blank" is added to the other list. (blanks represent a skipping of communication; i.e. if a process reads a blank in its recv_neighbor list, it will not receive any messages that round).

(c) If both of the neighbors are reached, both potential neighbors are added to their respective lists for communication.

For the example problem that produced Figures 2.10 and 2.11, the ordering scheme described here produces an ordering found in Figure 2.12, and the resulting communication rounds can be found in Figure 2.13.

Communication rounds for ordering B:

p	1	2	3	4
0			2	1
	1	2		
1	0		3	2
	2	3		0
2	1	0	4	3
	3	4	0	1
3	2	1	5	4
	4	5	1	2
4	3	2	6	5
	5	6	2	3
5	4	3	7	6
	6	7	3	4
6	5	4		7
	7		4	5
7	6	5		
			5	6

Figure 2.13: The communication rounds produced by ordering B (Figure 2.12)

2.3.8.2.1 Extending to higher-dimensional process grids

To extend the ordering scheme just described to work for m-dimensional process grids, xsim_hpc categorizes neighboring processes in terms of dimension. The dimension of a neighboring process is the number of dimensions in the process grid that must be traversed in order to reach the neighboring process (e.g. in a 3-dimensional process grid, moving from a process at (0,0,1) to a process at (0,1,2) requires traversal over 2 dimensions (since coordinates 2 and 3 are changed); thus, process (0,1,2) is a 2-dimensional neighbor of (0,0,1), and vice versa). In xsim_hpc, the list of neighbors for each processes is built in order of dimension (the 1-dimensional neighbors are added first, then the 2-dimensional neighbors, and so on).

For each possible dimension of neighbors, all dimension-tuples of process domain dimensions are looped over. For each dimension-tuple, potential neighbors are determined by recursively performing Step 1 (equations 2.57 and 2.58) for each dimension. Step 2 remains the same as in the 1-D algorithm.

2.4 Theoretical analysis of the parallel implementation

In Section 2.3.3, I alluded to the fact that, for a fixed number of processes, there exist multiple options for partitioning the vector index domain, and that the choice of partitioning can have an effect on the total cost of communication and amount of load imbalance. It is important to understand precisely how the choice of partitioning affects these properties so that the most efficient partitioning may

be chosen for an implementation. These effects can be measured theoretically, and this section is devoted to that theoretical analysis. For simplicity, this section will only consider the theoretical aspects of an LVC KDC model (5-point stencil problem in 2-D, 7-point in 3-D, etc.), but the resulting conclusions should extend to higher-order models. First, let me introduce some notation that will be used in the upcoming theoretical analysis:

n - the length of the vector index domain in each dimension

m - the number of dimensions of the vector index domain

N - the total number of vibrational basis functions ($N = n^m$)

P - the total number of processes used in the parallel implementation

For simplicity, we assume that the vector index domain is a hypercube (all dimensions have the same length, n).

2.4.1 Communication costs of various partitionings

To analyze the communication costs of various domain partitionings, it is perhaps most appropriate to compare the communication costs of each partitioning *in the weak scaling limit*. Because the primary goal of this parallelization research is to allow for the study of larger molecules than is currently possible (i.e. using more processes to make tractable larger global problem sizes), the weak scaling limit ($P \to \infty$ while N/P remains constant) is the limit case of interest. When looking at weak scaling for this problem, $N = n^m$ can grow with P in one of

two ways: 1) by increasing n, or 2) by increasing m. For most vibronic coupling problems, the number of vibrational modes in the system is a fixed value, and the basis size needs to increase to converge the variational calculation. As a result, as N grows with P, n will increase, and m will remain fixed.

In this section, I will compare the communication costs involved in two different partitioning schemes that can be thought of as representing two extremes in the space of possible partitionings:

1. **"Naive" partitioning**: In this partitioning scheme, the vectors x and y are sliced into consecutive N/P sized chunks, each assigned to a process. Using this partitioning scheme, as P grows to infinity, processes partition the first dimension until $P = n$, then, with the first dimension completely saturated, processes partition the second dimension until $P = n^2$, and so on. In the weak scaling limit ($n, P \to \infty$ such that N/P is constant), every dimension of the index domain will be completely saturated with processes except for the last. In practice, this limit case is currently impossible to achieve for a reasonable N/P,[10] but it is still useful to look at the behavior of these partitioning schemes in the limit as $P \to \infty$.

2. **Even distribution**: In this partitioning scheme, processes are distributed across dimensions as evenly as possible; i.e. for $P = 2^m$, the index domain

[10] "Reasonable N/P", here, means that the local problem size (N/P) is large enough that the global problem is worth parallelizing with P processes. Assuming N/P is roughly 100 million (800 MB of memory required per vector per process), to reach the point at which P processes saturate $m - 1$ dimensions of the vector index domain, (100 million)$^{m-1}$ processes must be used, which is very much outside the realm of possibility for $m > 2$.

is partitioned such that each of its dimensions is bisected. In the weak limit, each dimension of the vector index domain will be partitioned by $p = P^{(1/m)}$ processes and each process will have a *local* m-dimensional vector index domain of length n/p in each dimension.

The LVC stencil reaches only one point in each direction in each dimension. For the "naive" partitioning scheme, each process will have to communicate its entire local domain twice (once in each direction) for each of the $m - 1$ saturated dimensions of the vector index domain, plus one point in each direction for the final dimension. The total amount of communication performed will then be

$$2(m-1) \cdot \frac{N}{P} + 2 \approx 2(m-1) \cdot \frac{N}{P}. \tag{2.59}$$

In constast, for the even distribution scheme, each process will have to send a fraction

$$\frac{p}{n} = \left(\frac{P}{N}\right)^{(1/m)} \tag{2.60}$$

of its local domain twice (once in each direction) for all m partitioned dimensions of the vector index domain. The total amount of communication is then

$$2m \cdot \left(\frac{P}{N}\right)^{(1/m)} \frac{N}{P}, \tag{2.61}$$

We can quantify the extra communication that the naive approach requires by looking at the ratio of the naive partitioning communication cost to the even partitioning communication cost:

$$\frac{2(m-1) \cdot (N/P)}{2m \cdot (P/N)^{(1/m)} \cdot (N/P)} = \frac{m-1}{m} \cdot \left(\frac{N}{P}\right)^{1/m}. \tag{2.62}$$

The value $(N/P)^{1/m}$ determines the extent of the excess communication that the naive partitioning creates. This ratio remains constant in the weak scaling limit.

The practical maximum value that this ratio can attain is highly dependent on the number of dimensions m in the vector index domain, although it might not seem that way at first glance. The reason for this dependency is that *in practice* the local domain size (N/P) has a machine-dependent maximum value in that it must be small enough to fit in a single process's available memory. For example, let us say that the maximum value that (N/P) can take (constrained by the amount of available memory) is roughly 100 million (100 million * 8 bytes $\approx$ 1GB). When $m = 4$, this means that the largest value that $(N/P)^{1/m}$ can attain is 100, since

$$100^4 = 100,000,000. \tag{2.63}$$

This means that, in the weak limit, the naive splitting scheme requires $100\times$ the amount of communication that the even splitting scheme requires, which is quite a significant difference. When $m = 8$, though, $(N/P)^{1/m}$ can only reach a maximum of 10, and when $m = 16$, the maximum possible value of $(N/P)^{1/m}$ is reduced even further to roughly 3.

In this section, we have determined that the even splitting partitioning scheme provides a factor of $(N/P)^{1/m}$ improvement in the amount of communication over the naive approach. The next logical question to ask is "Does this result matter in practical application?" If the computation cost of the algorithm far outweighs the communication cost (e.g. the communication accounts for $< 1\%$ of the total run time), then a factor of 4, 8, or even 64 reduction in communication cost will not

make a significant difference in the total run time. To determine the practical value of choosing an efficient partitioning scheme, I will now analyze the ratio of computation cost to communication cost in the matrix-vector multiplication operation.

2.4.1.1 Computation cost vs. communication cost

I will begin this analysis by defining some constants associated with computation and communication bandwidth:

γ - Computation rate (in flop/s)

β - Bandwidth (in byte/s)

Often, communication latency plays an important role in this type of analysis, but for my coarse-grained parallel implementation, bandwidth costs far outweigh latency costs, so I will ignore the cost of latency in this section.

2.4.1.1.1 Volume-to-surface ratio

For stencil problems that have two- or three-dimensional domains, a very popular argument exists (called the *volume-to-surface ratio* argument) to show that the cost of computation necessarily far outweighs the cost of communication, and, as a result, great parallel efficiency is guaranteed. Before analyzing the m-dimensional problem, I will first describe the volume-to-surface ratio argument and explain why it is not applicable to the vibronic coupling problem (where m is often much greater than 2 or 3).

To help explain the argument, consider an example problem on an $n \times n$ domain ($N = n \times n$ total points) distributed across P processes. The $n \times n$ domain is divided into 2-D patches of size $(n/\sqrt{P}) \times (n/\sqrt{P})$ (this corresponds to an even partitioning scheme as described in the previous section). To simplify notation, let $b = (n/\sqrt{P})$ (b is the size of each dimension of the *local domain*; in m dimensions, $b = (N/P)^{1/m}$).

In a 5-point stencil problem, communication occurs on the boundary of each process's local $b \times b$ domain, resulting in the communication of $4 \cdot b$ total points (or $32 \cdot b$ total bytes). Computation, on the other hand, occurs on all points in the process's local $b \times b$ domain (2*5 floating-point operations (flops) for each point), resulting in $10 \cdot b^2$ flops. The ratio of computation cost to communication cost can then be equated to the ratio of the "volume" of the local domain (in the 2-D case, "area") to the "surface" (the boundary elements).

Let the cost of computation be labeled T^{comp} and the cost of communication be labeled T^{comm}. The ratio of computation to communication is then

$$\frac{T^{comp}}{T^{comm}} = \frac{10 \cdot b^2 \cdot \gamma^{-1}}{32 \cdot b \cdot \beta^{-1}} = \left[\frac{5}{16} \cdot b\right] \frac{\beta}{\gamma}. \tag{2.64}$$

If we assume that β and γ have roughly equivalent values (not an unreasonable assumption, as we will soon see), this ratio is on the order of b. The argument for guaranteed efficient parallel scaling comes from the fact that, for 2-D and 3-D problems, the volume of the local domain is usually orders of magnitude larger than its surface. For example, in a 2-D domain, if the local domain of each process (b^2) is assumed to be large enough that the global problem is worth parallelizing, b must

be on the order of 100 to 1000, resulting in the computation cost outweighing the communication cost by a factor of 100 to 1000.

2.4.1.1.2 Moving to higher-dimensional domains

For an m-dimensional domain, the volume-to-surface ratio is

$$\frac{T^{comp}}{T^{comm}} = \frac{2(2m+1)\cdot(b^m)\cdot\gamma^{-1}}{8(2m)\cdot b^{m-1}\cdot\beta^{-1}} = \frac{(2m+1)\cdot b}{8m}\cdot\frac{\beta}{\gamma}. \tag{2.65}$$

For higher-dimensional domains, the ratio of computation to communication (for an even partitioning of the domain) is still on the order of b, but, when m is large (greater than 2 or 3, as is common in vibronic coupling problems), producing a local domain large enough for the global problem to be worth parallelizing requires a much smaller value of b. For example, when $m = 10$, a local domain of roughly size $1,000,000$ can be formed with $b = 4$, resulting in a computation to communication ratio of $1.05\cdot(\beta/\gamma)$. It is interesting to note that, for $b \geq 4$, the computation-to-communication ratio (assuming $\gamma = \beta$) will never be less than one[11], but the ratio is often *much* closer to one in vibronic coupling applications than the typical 2-D or 3-D domain problems.

When m is large (i.e. T^{comp}/T^{comm} is close to β/γ), the practical importance of choosing a good partitioning scheme is highly machine-dependent, since the communication cost can easily outweigh the computation if the machine being used has low communication bandwidth (small β) or a high flop/s rate (large γ) (or,

[11] At minimum, the ratio is only nearly 1/2 ($b = 2$).

in this case, a high memory bandwidth would be more appropriate, since the computational performance of a sparse matrix-vector multiplication is often bandwidth limited).

2.4.1.1.3 Computation cost vs. communication cost for naive partitioning

So far, I have only compared computation costs to communication costs in the weak limit of an even partitioning scheme. Using results from Sections 2.4.1 and 2.4.1.1.2, a similar ratio can be derived for the weak limit of a naive partitioning scheme. From equation 2.65,

$$\frac{T^{comp}}{T^{comm}_{even}} = \frac{2(2m+1)\cdot(b^m)\cdot\gamma^{-1}}{8(2m)\cdot b^{m-1}\cdot\beta^{-1}} = \frac{(2m+1)\cdot b}{8m}\cdot\frac{\beta}{\gamma}, \tag{2.66}$$

and from equation 2.62,

$$\frac{T^{comm}_{naive}}{T^{comm}_{even}} = \frac{2(m-1)\cdot(N/P)}{2m\cdot(P/N)^{(1/m)}\cdot(N/P)} = \frac{m-1}{m}\cdot\left(\frac{N}{P}\right)^{1/m} = \frac{m-1}{m}\cdot b. \tag{2.67}$$

With some algebraic rearrangement of terms, we arrive the following ratio of computation to communication for the naive partitioning scheme:

$$\frac{T^{comp}}{T^{comm}_{naive}} = \frac{2m+1}{8(m-1)}\cdot\frac{\beta}{\gamma}. \tag{2.68}$$

If we again assume (β/γ) is equal to one, then, in the weak limit of the naive partitioning scheme, communication cost outweighs computation cost by approximately a factor of four, regardless of the size of b.

2.4.1.2 Discussion: Importance of choice of partitioning

Using the results above, we can analyze the effect of the choice of partitioning on the final run time of the algorithm. Let T_{naive} be the total run time of the algorithm (in the weak limit) using a naive partitioning:

$$T_{naive} = T^{comp} + T^{comm}_{naive}. \tag{2.69}$$

Using equation 2.68, we can rewrite this as

$$T_{naive} \approx T^{comp} + 4 \cdot T^{comp} = 5 \cdot T^{comp}. \tag{2.70}$$

Let T_{even} be the total run time of the algorithm (in the weak limit) using a even partitioning:

$$T_{even} = T^{comp} + T^{comm}_{even}. \tag{2.71}$$

Using equation 2.65, we can rewrite this as

$$T_{even} \approx T^{comp} + \frac{4}{b} \cdot T^{comp} = (1 + \frac{4}{b}) \cdot T^{comp}. \tag{2.72}$$

The ratio of the total run time of a calculation (in the weak limit) that uses a naive partitioning scheme to that of one that uses an even partitioning scheme is then

$$\frac{T_{naive}}{T_{even}} = \frac{5}{1 + \frac{4}{b}} \tag{2.73}$$

From this result, we can conclude that an evenly partitioned calculation can result in a speedup of, at most, $5\times$ over a naively partitioned calculation. While this is not an extremely large factor, it is significant enough to motivate the use of an even partitioning scheme over a naive one.

2.4.1.3 Extending to higher-order models

In the analysis so far, we assumed, for simplicity, that our model is a linear vibronic coupling model. In this section, we will take a quick look at how the analysis changes when higher-order models are considered. Let $b = \left(\frac{N}{P}\right)^{1/m}$. Then, the amount of communication M_i that occurs in a KDC model of order i (the necessary communication, not with the additional data actually communicated in our implementation; see Section 2.3.6) is

$$M_1 = \underbrace{\left[2m \cdot b^{-1} \cdot (N/P)\right]}_{\text{1-D neighbors}} \tag{2.74}$$

$$M_2 = \underbrace{\left[2 \cdot (2m) \cdot b^{-1} \cdot (N/P)\right]}_{\text{1-D neighbors}} + \underbrace{\left[4\binom{m}{2} \cdot b^{-2} \cdot (N/P)\right]}_{\text{2-D neighbors}} \tag{2.75}$$

$$\begin{aligned} M_3 = &\underbrace{\left[3 \cdot (2m) \cdot b^{-1} \cdot (N/P)\right]}_{\text{1-D neighbors}} + \underbrace{\left[3 \cdot 4\binom{m}{2} \cdot b^{-2} \cdot (N/P)\right]}_{\text{2-D neighbors}} \\ &+ \underbrace{\left[1^3 \cdot 8\binom{m}{3} \cdot b^{-3} \cdot (N/P)\right]}_{\text{3-D neighbors}} \end{aligned} \tag{2.76}$$

$$\begin{aligned} M_4 = &\underbrace{\left[4 \cdot (2m) \cdot b^{-1} \cdot (N/P)\right]}_{\text{1-D neighbors}} + \underbrace{\left[6 \cdot 4\binom{m}{2} \cdot b^{-2} \cdot (N/P)\right]}_{\text{2-D neighbors}} \\ &+ \underbrace{\left[4 \cdot 8\binom{m}{3} \cdot b^{-3} \cdot (N/P)\right]}_{\text{3-D neighbors}} + \underbrace{\left[1^4 \cdot 16\binom{m}{4} \cdot b^{-4} \cdot (N/P)\right]}_{\text{4-D neighbors}} \end{aligned} \tag{2.77}$$

while the amount of computation C_i that occurs in a KDC model of order i is

$$C_1 = \underbrace{\left[2m \cdot (N/P)\right]}_{\text{1-D edges}} \tag{2.78}$$

$$C_2 = \underbrace{[2 \cdot (2m) \cdot (N/P)]}_{\text{1-D edges}} + \underbrace{\left[4\binom{m}{2} \cdot (N/P)\right]}_{\text{2-D edges}} \tag{2.79}$$

$$C_3 = \underbrace{[3 \cdot (2m) \cdot (N/P)]}_{\text{1-D edges}} + \underbrace{\left[3 \cdot 4\binom{m}{2} \cdot (N/P)\right]}_{\text{2-D edges}} + \underbrace{\left[1^3 \cdot 8\binom{m}{3} \cdot (N/P)\right]}_{\text{3-D edges}} \tag{2.80}$$

$$C_4 = \underbrace{[4 \cdot (2m) \cdot (N/P)]}_{\text{1-D edges}} + \underbrace{\left[6 \cdot 4\binom{m}{2} \cdot (N/P)\right]}_{\text{2-D edges}} + \underbrace{\left[4 \cdot 8\binom{m}{3} \cdot (N/P)\right]}_{\text{3-D edges}} + \underbrace{\left[1^4 \cdot 16\binom{m}{4} \cdot (N/P)\right]}_{\text{4-D edges}} \tag{2.81}$$

Note that both the communication and computation costs for each order of model include similar terms (since the same stencil determines both the computation and communication), but, for the communication cost only, each term of increasing dimension includes an additional factor of b^{-1} (1-D neighbor terms are weighted by b^{-1}, 2-D neighbor terms are weighted by b^{-2}, etc.). This is because the communication that arises from d-dimensional stencil edges only occurs on $(m-d)$-dimensional surfaces of the vector index domain. In constrast, computation that arises from d-dimensional stencil edges is performed on the entire volume of the vector index domain, so the corresponding terms in equations 2.78-2.81 are not weighted by powers of b^{-1}.

The important observation here is that, as the KDC model becomes of higher and higher order, both the amount of computation and communication increase,

but the amount of computation increases by a larger factor. As a result, we can expect computation to further outweigh communication as the order of the model increases, resulting in higher parallel efficiency (we will see this effect in Section 2.5). Because of this, it is not worth investigating the difference in communication cost between various partitionings for higher-order models, since communication cost becomes less important to the overall run time.

2.4.2 Load imbalance of various partitionings

So far, we have discussed the effects of the choice of partitioning on the amount of communication that occurs and how much these differences affect the total run time of the algorithm. In addition to the communication cost, the choice of partitioning also affects the amount of computation that occurs on each process. Let us quantify this effect to better understand it.

Specifically, the choice of partitioning affects the amount of computational *load imbalance* that exists in the parallel implementation (how well the computational workload is distributed among the available processes). In the linear vibronic coupling (LVC) framework, the total amount of computation is a sum of m separate computations, each defined by a one-dimensional stencil problem, since the stencil edges only extend across one dimension at a time. As a result, load imbalance can be calculated by only considering one dimension of the vector index domain at a time.

Let us consider a vector index domain of length n in each dimension, partitioned in a single dimension using P processes. The global computation for the

partitioned dimension is comprised of $2(n-2)+2$ multiply-add operations (2 multiply-adds for each interior point and 1 multiply-add for each boundary point; in this section, we ignore diagonal matrix elements (i.e. the center of the stencil)) for each point in the global domain, resulting in $(2(n-2)+2) \cdot N$ multiply-add operations.

If only two processes are used to split a dimension (the dimension is bisected), the workload is distributed evenly across the processes and no load imbalance exists; each of the two processes contains the same number of interior points and boundary points and thus performs the same number of multiply-add operations. As a result, to study load imbalance, we must consider the case when $P > 2$ and focus on the interior processes in the process grid, as these are assigned the most work (more work than the boundary processes).

When $P > 2$, interior processes perform 2 multiply-adds for each point in their local domain, resulting in $2 \cdot (N/P)$ multiply-adds. The amount of computational imbalance can be quantified by dividing this number by the number of multiply-adds that each process would perform if work were split evenly ($(2(n-2)+2) \cdot N/P$). The amount of load imbalance is then

$$\frac{2n}{2(n-2)+2} = \frac{2n}{2n-2} = \frac{n}{n-1}. \tag{2.82}$$

It is interesting to note that this fraction does not depend at all on P; i.e. any parallel splitting that produces interior processes $(P > 2)$ will have the same amount of load imbalance.

The value in 2.82 refers to the load imbalance of the computation in a sin-

gle dimension. To calculate the load imbalance of the full computation, we must calculate the sum of the load imbalance in each dimension, divided by the number of dimensions.

For example, consider a domain with $n = 4$ and $m = 15$. If $P = 1024$, we can distribute the processes either as pairs across 10 dimensions ($P = 2^{10}$; $p = 2$ for 10 dimensions) or in sets of 4 across 5 dimensions ($P = 4^5$; $p = 4$ for 5 dimensions). In the first case, work is distributed evenly across processes and no load imbalance occurs. In the second case, we can expect a load imbalance factor of

$$\frac{5 \cdot \left(\frac{4}{3}\right) + 10 \cdot 1}{15} \approx 1.11, \tag{2.83}$$

meaning that the parallel calculation with $p = 4$ should run roughly 1.11 times slower than the calculation that distributes work evenly ($p = 2$) (ignoring differences in the amount of communication that occurs with each splitting). This example should help illuminate the relative insignificance of the choice of partitioning's effect on load imbalance (for the LVC model, at least); even in a somewhat "worst case scenario" example ($n = 4$ is quite low, bringing $(n/(n-1))$ to an almost maximum), the load imbalance caused by using an inefficient partitioning only results in a 10% increase in run time.

2.5 Experimental results

This section is devoted to showcasing timing information that I have measured for various model configurations (model order, number of processes, choice of partitioning scheme, etc.).

2.5.1 Parallel scaling efficiency

To measure the effectiveness of a parallelization scheme, it is common to perform a *scalability* analysis. There are two main approaches to analyzing parallel scalability:

1. **Strong scaling**: Measure how the solution time varies with the number of processes for a fixed *total* problem size. The strong scaling efficiency of a run with P processes is calculated by

$$E_P = \frac{T_1}{P \times T_P} \tag{2.84}$$

   where T_1 is the run time using a single process, and T_P is the run time *of the same problem* using P processes.

2. **Weak scaling**: Measure how the solution time varies with the number of processes for a fixed problem size *per process*. The weak scaling efficiency of a run with P processes is calculated by

$$E_P = \frac{T_1}{T_P} \tag{2.85}$$

   where the term T_1 is the run time using a single process, and T_P is the run time *of a problem P times larger than the original* using P processes.

Because the goal of this research is to extend the reach of the vibronic coupling model to larger and larger systems (increasing total problem size), I felt that it was more appropriate to perform a weak scaling analysis of the implementation. The results of my analysis are summarized throughout this section.

To perform a weak-scaling analysis of my parallel implemenntation of the KDC matrix-vector multiplication, I began with a single process performing a matrix-vector multipication for a single-electronic-state computation using a basis of size 67 million. The vibrational basis used was comprised of 13 modes with 4 harmonic oscillator basis functions for each mode. I then repeatedly doubled the total size of the basis along with the number of processes. The basis size was initially doubled (when P doubled from 1 to 2) by doubling the number of harmonic oscillator functions for the 13th mode. It was next doubled (when P doubled from 2 to 4) by doubling the number of harmonic oscillator in the 12th mode, and so on. The total basis was partitioned using an "even" partitioning scheme: the first doubling of the number of processes occurred by bisecting the 13th mode; the second by bisecting the 12th mode, and so on. With this approach, the *local* basis of each process remained a constant size of 4 functions per mode throughout the analysis. The vibrational basis compositions and partitions for up to 32 processes can be found in Table 2.1. All computations assume a complete stencil (all force constants $F^{ij}_{klmn...}$ are non-zero) at each level of truncation.

All computations were run on the Stampede supercomputer maintained by the Texas Advanced Computing Center (TACC) in Austin, Texas. The Stampede supercomputer has 6400 nodes, organized in 160 racks of 40 nodes. Each rack of Stampede is organzied into two groups of 20 nodes with an interconnect as shown in Figure 2.14. Each node of Stampede contains two sockets, each with a Xeon Intel 8-core 64-bit E5-processor with core frequency 2.7 GHz, and 32 GB of memory.

	$P = 1$	$P = 2$
Basis	4 4 4 4 4 4 4 4 4 4 4 4 4	4 4 4 4 4 4 4 4 4 4 4 4 8
Even Partition	1 1 1 1 1 1 1 1 1 1 1 1 1	1 1 1 1 1 1 1 1 1 1 1 1 2
Naive Partition	1 1 1 1 1 1 1 1 1 1 1 1 1	1 1 1 1 1 1 1 1 1 1 1 1 2

	$P = 4$	$P = 8$
Basis	4 4 4 4 4 4 4 4 4 4 4 8 8	4 4 4 4 4 4 4 4 4 4 8 8 8
Even Partition	1 1 1 1 1 1 1 1 1 1 1 2 2	1 1 1 1 1 1 1 1 1 1 2 2 2
Naive Partition	1 1 1 1 1 1 1 1 1 1 1 1 4	1 1 1 1 1 1 1 1 1 1 1 1 8

	$P = 16$	$P = 32$
Basis	4 4 4 4 4 4 4 4 4 8 8 8 8	4 4 4 4 4 4 4 4 8 8 8 8 8
Even Partition	1 1 1 1 1 1 1 1 2 2 2 2	1 1 1 1 1 1 1 1 2 2 2 2 2
Naive Partition	1 1 1 1 1 1 1 1 1 1 2 8	1 1 1 1 1 1 1 1 1 1 4 8

Table 2.1: The vibrational basis composition and even/naive partitionings for each of the calculations performed for the weak scaling analysis (for up to $P = 32$). Each number in the Basis row represents the quanta trunction n_i for each of the thirteen modes. The partitioning rows specify into how many sections each dimension of the vector index domain is partitioned (2 represents a bisection, 4 represents a partitioning into 4 sections, etc.).

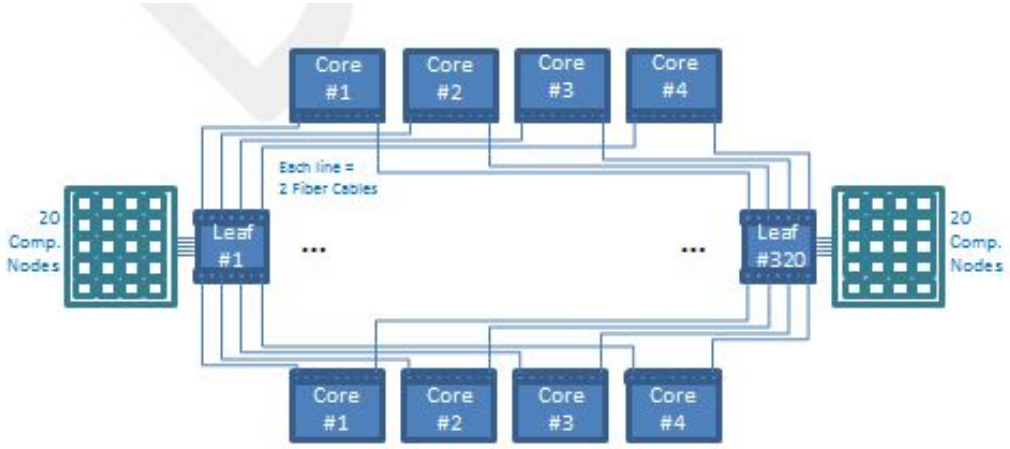


Figure 2.14: The interconnect used in the Stampede supercomputer; taken from https://portal.tacc.utexas.edu/user-guides/stampede

For each computation performed in my weak scaling analysis, I allowed a maximum of one MPI process per node (even though 16 cores are available on each node). My reasoning for this restriction is two-fold:

1. The KDC matrix-vector multiplication operation is very memory-bandwidth bound. If more than one process is used per socket, processes must share the available bandwidth. The result is a local computation time that is longer than if only one process had been used. Using more than one process per node would then artificially lower the value of (discussed in Section 2.4.1.1). Since one of my goals for this scaling analysis was to observe the value of γ/β in practice, I decided to use only one process per node.

2. If more than one MPI process were assigned per node, then both intranode *and* internode communication would occur. Since intranode communication is often faster than internode communication, using multiple processes per node would create a heterogeneous communication environment. For the analysis in this section, I hoped to obtain scaling results that were consistent with the theoretical results reported in the previous section. In order for this goal to be a possibility, I needed to create a message-passing environment that was as homogenous as possible.

For an accurate scaling analysis, each measured run time had to be normalized by the number of floating-point operations performed in the run, relative to the number of floating-point operations performed for a the single-process calculation. Normalization was required because each doubling of the basis (as was done for the

weak scaling analysis) increased the total number of floating-point operations by a factor of slightly more than two. A simple normalization of the measured execution time was not appropriate, however; rather, only the execution time attributed to computation (as opposed to the execution time attributed to communicaiton) should be normalized. To perform the normalization, then, the total execution time had to be separated into two summands: the time attributed to computation (computation time) and the time attributed to communication (communication time).

In general, the total execution time of a run can be written as the sum of computation time and communication time:

$$T = T^{comp} + T^{comm}. \tag{2.86}$$

Let us define T_P as the total execution time for a run using P processes. Then,

$$T_P = T_P^{comp} + T_P^{comm}. \tag{2.87}$$

In practice, we can accurately measure the total execution time: place MPI_Barrier() calls around the matrix-vector multiplication routine[12], place MPI_Wtime() calls after each barrier, and subtract the two times. Measuring separate computation times and communication times, though, can be difficult[13]. As a result, for this analysis, I determined the values of T_P^{comp} and T_P^{comm} from T_P analytically using the following procedure.

[12] While it is true that not all processes will complete their work in *exactly* the same amount of time, in the Lanczos algortihm, there will be an implicit barrier in the form of a vector inner product after the matrix-vector multiplication, so ending the routine with a barrier is appropriate.

[13] We can not simply place MPI_Wtime() calls around each MPI_Send() / MPI_Recv() call; those measurements can be misleading, especially for partitionings in which communication is not evenly distributed across all processes.

When only one process is used ($P = 1$), the communication time is necessarily zero, so

$$T_1 = T_1^{comp}. \tag{2.88}$$

To determine T_P^{comm}, then, for any given P, I first assume that all processes achieve the same number of FLOP/s for all runs.[14] With this assumption in place, T_P^{comp} can be calculated theoretically from T_1^{comp} (which we can measure): Let $[\text{FP}]_P$ be the number of floating-point operations involved in a run using P processes; then,

$$T_P^{comp} = \frac{[\text{FP}]_P}{[\text{FP}]_1} \times T_1^{comp}. \tag{2.89}$$

The communication time T_P^{comm} for each run can then be determined from T_1^{comp} by using the formula

$$T_P^{comm} = T_P - \frac{[\text{FP}]_P}{[\text{FP}]_1} \times T_1^{comp}. \tag{2.90}$$

The normalized execution time for a run with P processes is then

$$T_P^{norm} = T_1^{comp} + T_P^{comm}. \tag{2.91}$$

The timings (and resulting scaling efficiencies) for each run of the weak scaling analysis can be seen in Table 2.2. To create a message-passing environment for this analysis that was as homogenous as possible, the first five timings for each scaling test (using 1, 2, 4, 8, and 16 processes) were performed with processes

[14] In actuality, a (very) small decrease in a process's FLOP/s rate may be observed with increasing process counts. With each increase in process count, more and more of each process's computation is partitioned into chunks (calculations on ghost points). Performing the required flops in individual chunks can be less efficient than performing them all at once, but I consider this decrease in performance to be negligible.

		LVC			QVC		
Basis size	Cores	T_P	T_P^{comm}	E_P	T_P	T_P^{comm}	E_P
67 m	1	1.865	—	—	18.95	—	—
134 m	2	1.909	0.044	0.977	19.03	0.079	0.995
68 m	4	1.963	0.098	0.950	19.15	0.207	0.989
536 m	8	2.009	0.143	0.928	19.25	0.305	0.984
1.07 b	16	2.053	0.188	0.908	19.42	0.477	0.975
2.14 b	32	2.189	0.323	0.852			
4.28 b	64	2.222	0.357	0.839			
8.56 b	128	2.342	0.476	0.796			
17.12 b	256	2.466	0.601	0.756			

Table 2.2: Normalized matrix-vector multiplication timings T_P, communication times T_P^{comm} (in seconds), and corresponding parallel efficiencies E_P for LVC and QVC vibronic coupling models.

assigned to consecutive nodes contained in the same rack. This ensured that all processes were spawned within a single group of 20 (seen in Figure 2.14). As a result, the scaling observed across the first five runs is very consistent with theoretical expectations. For example, for the LVC calculations, each doubling of P is accompanied by the bisecting of a new dimension of the vector index domain. With each new bisection, we expect, theoretically, for the total communication to increase by one additional message of the same size as each existing message. In Table 2.2, we can see that the linear scaling of the communication time T_P^{comm} supports the theoretical expectations. Any runs using 32 or more processes will deviate from theory due to inhomogeneous communication, but since many machines in practice do not have a completely homogenous network, it is still interesting to analyze scaling beyond 16 cores (as is done for the LVC model).

Overall, it is difficult to come to a conclusion of "good" or "bad" scaling efficiency, since we do not have any other implementation to compare against, but it is still possible to draw some interesting conclusions from our results. From the LVC scaling results, we can hypothesize about the practical value of γ/β. Remember, from Section 2.4.1.1, that the ratio of computation to communication for an LVC model with even partitioning (in the weak scaling limit) is

$$\frac{T^{comp}}{T^{comm}_{even}} = \frac{(2m+1)\cdot b}{8m} \cdot \frac{\gamma}{\beta} \qquad \text{(2.65 revisited)}$$

For our 13-mode model with $b = 4$, this becomes

$$\frac{T^{comp}}{T^{comm}_{even}} = \frac{(27)\cdot 4}{104} \cdot \frac{\gamma}{\beta} \approx 1.04 \cdot \frac{\gamma}{\beta} \qquad (2.92)$$

At 256 cores, with a 13-mode model, we have not yet reached the weak scaling limit (as is discussed in Section 2.4.1.1), but we can attempt to extrapolate the communication time T^{comm}_P in the weak limit from the information that we do have. Using linear regression (since, theoretically, the communication time for an LVC calculation should increase linearly with each new bisected dimension), we can estimate that the communication time for a 8192 process calculation (a calculation for which *all* 13 dimensions are bisected) will be roughly 0.9 seconds. As the process count increases further to reach the weak limit, the communication time should reach a maximum of 1.8 seconds (double the communication time for the 8192 process calculation). We expect a doubling of communication time becuase, in the weak limit, interior processes must communicate along all 13 dimensions (as is done for the 8192 process calculation), but in both directions. The ratio

T^{comp}/T^{comm}_{even} then becomes $(1.865/1.8)$, so

$$\frac{1.865}{1.8} = 1.04 \cdot \frac{\gamma}{\beta} \tag{2.93}$$

$$\rightarrow \frac{\gamma}{\beta} \approx 1 \tag{2.94}$$

Interestingly, the observed value of γ/β is almost exactly the value we assumed in Section 2.4.1.1. This information could be useful for any future theoretical predictions related to comparing computation costs and communication costs.

Another result that we can draw from the weak scaling analysis is a confirmation of the theoretical hypothesis proposed in Section 2.4.1.3: As the order of the KDC model increases (LVC to QVC to cubic and so on), the communication costs increase at a much slower rate than the computation costs, and, as a result, good parallel scalability can be assumed for higher-order models. Looking at Table 2.2, the QVC models scales significantly better than the LVC model, and we can expect models of even higher order to scale even better.

2.5.2 Even vs. naive partitioning

In Section 2.4.1, I introduced two possible domain partitioning schemes and discussed how the use of each can affect the amount of communication that occurs throughout the matrix-vector multiplication operation. I wanted to compare the two partitioning schemes in practice, so I performed a weak scaling analysis using the same KDC model used in the previous section but with a "naive" approach to partitioning the index domain instead of an "even" approach. With each doubling of the number of processes, the highest non-saturated dimension is partitioned further.

Once a dimension is completely saturated with processes, the next highest dimension is partitioned, and so on. The specific partitioning for each calculation (up to $P = 32$) can be found in Table 2.1.

To calculate T_P^{comm} for each calculation, a procedure similar to the one used for the even partitioning calculations was used. A slight modification was needed, however. In Section 2.4.2, I talked about the computational load imbalance that can exist with certain partitionings. For up to $P = 256$, with our 13-mode model, the even partitioning scheme only bisects dimensions, resulting in perfect load balance. The naive partitioning scheme, however, creates load imbalance as soon as $P = 4$. It is important that the load imbalance is accounted for when calculating T_P^{comp}. I accounted for load imbalance by calculating the *load imbalance factor* for each run. The load imbalance factor is simply the maximum number of floating point operations performed by any one process (calculated by my code at run time) divided by $[\text{FP}]_P/P$. I then used that factor to calculate the computation time attributed to load imbalance, $[\text{LI}]_P$, using the equation

$$[\text{LI}]_P = T_1^{comp} \times \frac{[\text{FP}]_P}{[\text{FP}]_1} \times \text{load imbalance factor.} \tag{2.95}$$

The communication time T_P^{comm} is then calculated by

$$T_P^{comm} = T_P - \frac{[\text{FP}]_P}{[\text{FP}]_1} \times T_1^{comp} - [\text{LI}]_P. \tag{2.96}$$

The weak scaling results are displayed in Table 2.3. Comparing these results to those in Table 2.2, we can see that there is, in fact, a noticeable difference in scaling efficiency between the two partitioning schemes. At 256 processes, the

		LVC			
Basis size	Cores	T_P	T_P^{comm}	$[\text{LI}]_P$	E_P
67 m	1	1.865	—	—	—
134 m	2	1.909	0.044	—	0.977
268 m	4	2.076	0.189	0.022	0.899
536 m	8	2.257	0.370	0.021	0.826
1.07 b	16	2.302	0.415	0.021	0.810
2.14 b	32	2.605	0.698	0.042	0.716
4.28 b	64	2.792	0.885	0.041	0.668
8.56 b	128	2.915	1.009	0.041	0.639
17.12 b	256	3.148	1.222	0.061	0.592

		QVC			
Basis size	Cores	T_P	T_P^{comm}	$[\text{LI}]_P$	E_P
67 m	1	18.95	—	—	—
134 m	2	19.03	0.079	—	0.995
268 m	4	19.83	0.427	0.457	0.955
536 m	8	20.12	0.716	0.452	0.942
1.07 b	16	20.41	1.016	0.446	0.928

Table 2.3: Normalized matrix-vector multiplication timings T_P, communication times T_P^{comm} (in seconds), and corresponding parallel efficiencies E_P for LVC and QVC vibronic coupling models. Each run used the naive domain partitioning described in Section 2.4.1.

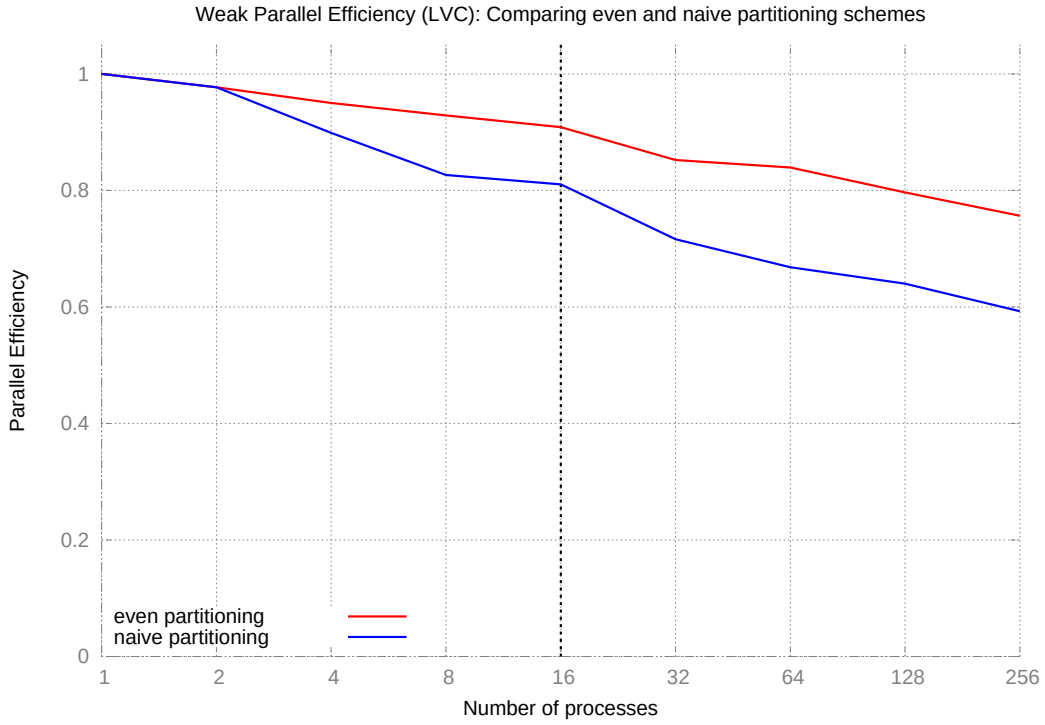


Figure 2.15: Parallel efficiency graph comparing the scaling of calculations using the even and naive domain partitioning schemes. The dotted line represents the point after which we expect the efficiency analysis to deviate slightly from theory, since, for $P = 32$ and higher, processes are no longer assigned to the same rack.

naive partitioning scheme achieves only 59.2% efficiency for the LVC model, while the even partitioning scheme achieves 75.6%. Similarly, at 16 processes, for the QVC model, the naive partitioning scheme achieves 92.8% efficiency while the even partitioning scheme achieves 97.5%. Graphical comparisons of the scaling achieved using even and naive partitioning schemes can be found in Figures 2.15 and 2.16.

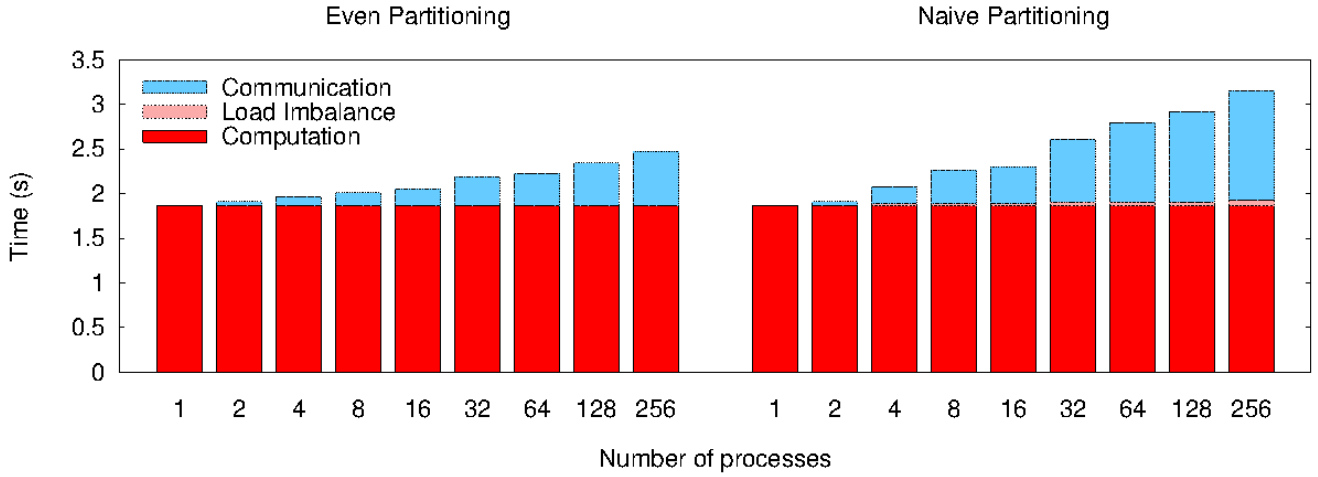


Figure 2.16: Bar graph describing the breakdown of the total execution time T_P into computation time T_P^{comp}, load imbalance $[\text{LI}]_P$, and communication time T_P^{comm} for calculations using the even and naive domain partitioning schemes.

		LVC			QVC		
Basis size	Cores	T_P	T_P^{comm}	E_P	T_P	T_P^{comm}	E_P
67 m	1	1.865	—	—	18.95	—	—
134 m	2	1.909	0.044	0.977	19.03	0.079	0.995
68 m	4	2.016	0.151	0.925	20.19	1.242	0.938
536 m	8	2.196	0.331	0.849	22.01	3.063	0.861
1.07 b	16	2.462	0.597	0.758	25.48	6.540	0.743
2.14 b	32	2.889	1.024	0.646			
4.28 b	64	3.321	1.456	0.562			
8.56 b	128	3.934	2.069	0.474			
17.12 b	256	4.946	3.081	0.377			

Table 2.4: Normalized matrix-vector multiplication timings T_P, communication times T_P^{comm} (in seconds), and corresponding parallel efficiencies E_P for LVC and QVC vibronic coupling models. Each run used the sorted ascending message ordering described in Section 2.3.8.

2.5.3 Comparison of message-ordering schemes

In Section 2.3.8, I introduced two possible message-ordering schemes and discussed how the use of each can affect the communication time involved in the matrix-vector multiplication operation. To see how large of a difference in communication time exists between the two approaches, I performed a weak scaling analysis using the same KDC model as has been used throughout this section with communication performed using the sorted ordering scheme. The weak scaling results are displayed in Table 2.4.

The analysis in Table 2.4 shows that an efficient message ordering is *crucial* to achieving good parallel scaling. At 256 processes, the sorted message ordering calculation only achieves 37.7% efficiency for the LVC model (compared to 75.6%

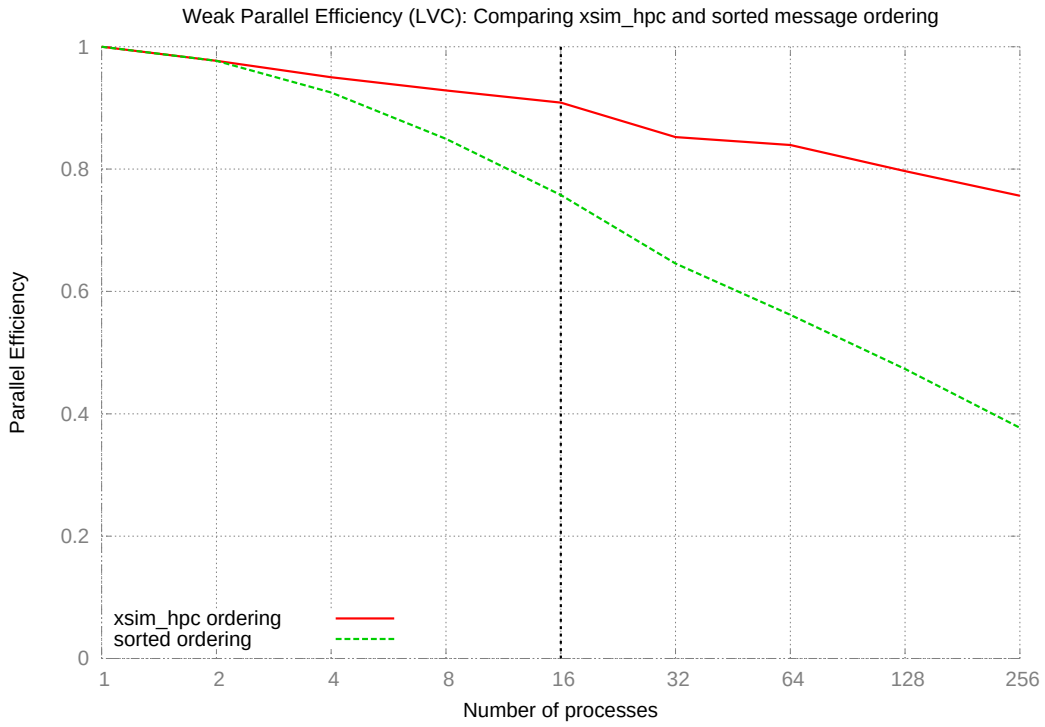


Figure 2.17: Parallel efficiency graph comparing the scaling of calculations using the xsim_hpc and sorted ascending message ordering schemes. The black vertical dotted line represents the point after which we expect the efficiency analysis to deviate slightly from theory, since, for $P = 32$ and higher, processes are no longer assigned to the same rack.

achieved using the xsim_hpc ordering). At 16 processes, the sorted message ordering calculation achieves just 74.3% efficiency for the QVC model (compared to 97.5% acheived using the xsim_hpc ordering). Graphical comparisons of the scaling achieved using the xsim_hpc and sorted ascending message orderings can be found in Figures 2.17 and 2.18.

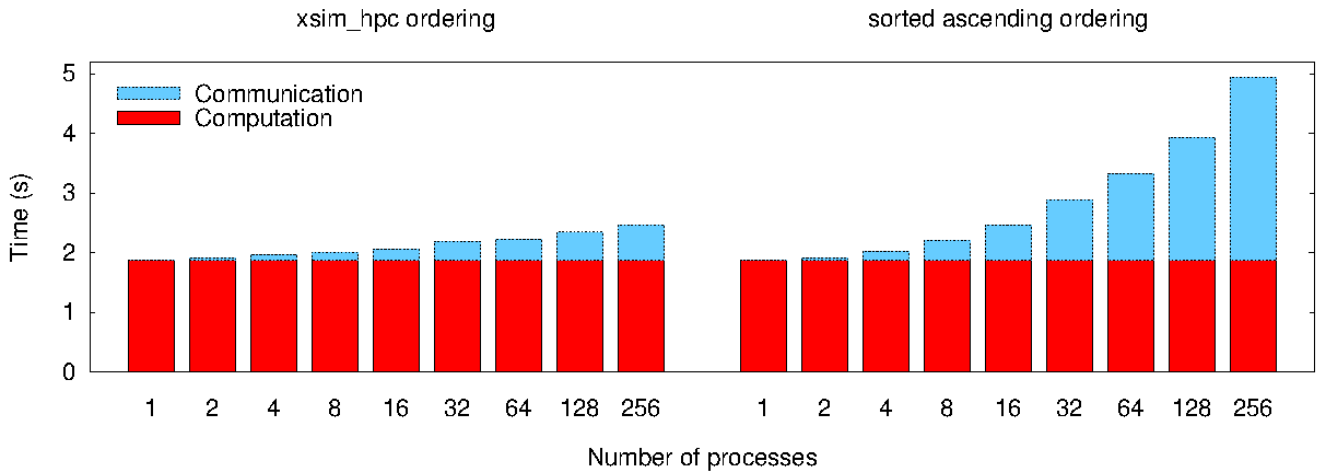


Figure 2.18: Bar graph describing the breakdown of the total execution time T_P into computation time T_P^{comp} and communication time T_P^{comm} for calculations using the xsim_hpc and sorted ascending message ordering schemes.

2.6 Applications: Ethane - C_2H_6

The xsim_hpc parallel implementation of the KDC/Lanczos algorithm has already been utilized for several research projects outside of the University of Texas at Austin. These projects required vibronic coupling spectral simulations that were too large to be tractable with previously available serial implementations. In this section, I will briefly describe the results of one of these projects and computational stresses that xsim_hpc helped to alleviate.

In a paper recently published in the Journal of Physical Chemistry A [62], Lee, Stanton, and I study the vibronic coupling that exists among the low-lying states of the radical cation of ethane. Ethane is an important molecule to study for a number of reasons. It is emitted from the combustion of fossil fuels [1] and is the second most abundant component of natural gas [40]. In astrochemistry, ethane is found on extraterrestrial objects such as ices [10, 37], in Jupiter's atmosphere [7, 83], and in lakes on the surface of Titan (one of Saturn's moons) [65].

To study the low-lying states of the radical cation of ethane, we parameterized a quadratic vibronic coupling (QVC) KDC model in hopes of calculating the photoelectron spectrum of ethane in the 11-15 eV region. The model accounted for coupling between three electronic states (the degenerate ${}^2E_g^{(A,B)}$ states and the ${}^2A_{1g}$ state) and included 9 vibrational modes. Optimized geometries for the ground and excited coupled states can be found in Figure 2.19. Spectral simulations were to be performed using LVC and QVC models to determine the importance of the quadratic terms of the KDC model potential.

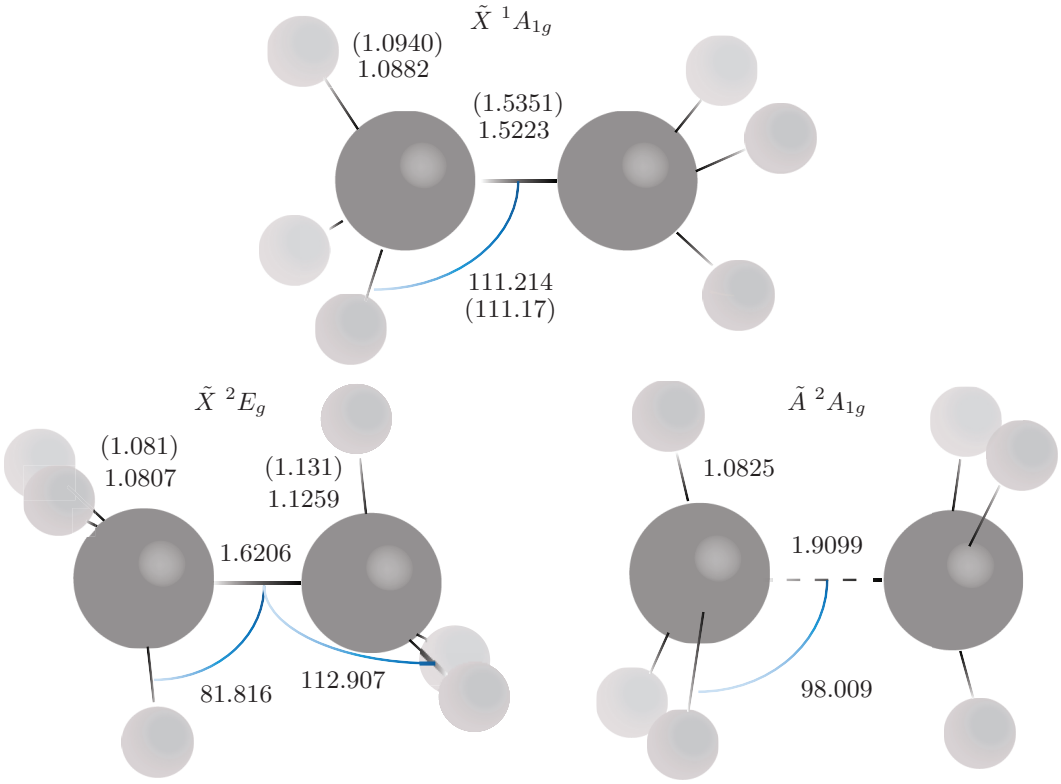


Figure 2.19: Minimum-energy geometries for the ground state ($\bar{X}^1A_{1g}$) of ethane and the low-lying states ($\bar{X}^2E_g$ and $\bar{A}^2A_{1g}$) of the radical cation of ethane. Image taken from [62].

Linear vibronic coupling model

n_1	n_2	n_3	n_4	n_5	n_6	n_7	n_8	n_9
4	24	12	4	4	16	16	16	16

Quadratic vibronic coupling model

n_1	n_2	n_3	n_4	n_5	n_6	n_7	n_8	n_9
5	24	12	4	4	28	28	36	36

Table 2.5: The truncation levels of the quanta for each mode in the vibrational basis for both the linear vibronic coupling model simulation and the quadratic vibronic coupling model simulation.

To produce a converged spectrum, the LVC model required 1.2 billion basis functions in the vibrational basis (a complete description of the maximum quanta trunctions used for each mode can be found in Table 2.5), resulting in a full Hamiltonian matrix of dimension 3.6 billion. With a Hamiltonian matrix of this size, the simulation requires 86.4 GB of memory (Lanczos requires the storage of three vectors, each of size 3.6 billion). Using 256 cores of the Stampede supercomputer (512 GB of available memory), the xsim_hpc parallel implementation performed the necessary 1000 Lanczos iterations in 36 minutes. If we assume 50% scaling efficiency (256 cores achieve 128$\times$ speedup over a serial implementation; this is a conversative estimate), a serial calculation for this model would take three days to run. The resulting spectrum is shown in Figure 2.20.

To produce a converged spectrum using the QVC model, the variational approach required almost 20$\times$ as many vibrational basis functions as was required for

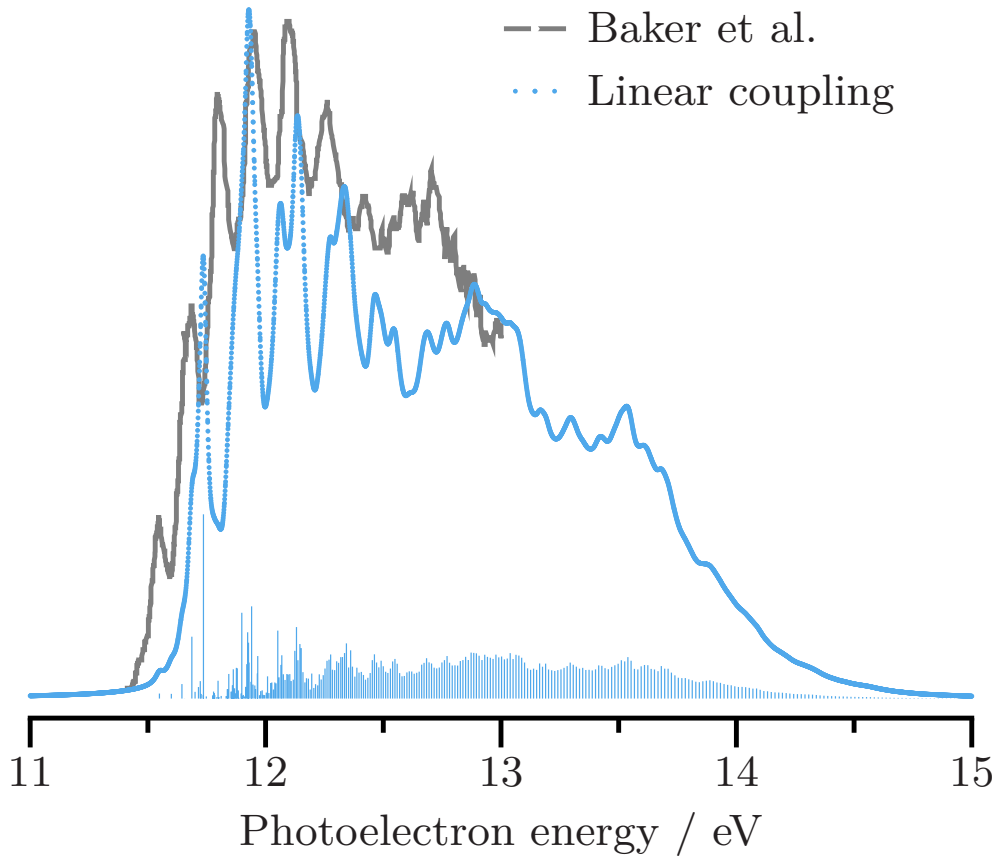


Figure 2.20: Comparison of the simulated spectrum using the LVC model to the observed spectrum. Image taken from [62].

the LVC model. In total, 23.4 billion basis functions were necessary (a complete description of the maximum quanta trunctions used for each mode can be found in Table 2.5), resulting in a full Hamiltonian matrix of dimension 70.2 billion. At this size, the QVC simulation required 1.7 TB of memory. The xsim_hpc parallel implementation was again used for this calculation and, using 2048 cores of the Stampede supercomputer (with a total of 4.1 TB of available memory), performed the necessary 1000 Lanczos iterations in 4.4 hours. If we again assume 50% scaling efficiency (2048 cores achieve 1024$\times$ speedup over a serial implementation; this is a *very* conversative estimate for a QVC model), a serial calculation for this model would take six *months* to run. It is clear that only with a parallel implementation of the KDC/Lanczos procedure is this study even possible. The resulting QVC spectrum is shown in Figure 2.21. Comparing Figures 2.20 and 2.21, it is clear that the inclusion of the quadratic terms of the quasidiabatic potential is necessary to achieve good agreement with the observed spectrum, and the xsim_hpc parallel implementation of the KDC/Lanczos algorithm was a critical component in producing these results.

2.7 Conclusion

In this chapter, I have discussed how to calculate a vibronic spectrum when the Born-Oppenheimer approximation is no longer applicable (when two or more electronic states are energetically close). The KDC model Hamiltonian for vibronic coupling calculations was introduced, and the variational method used to solve the time-independent Schrödinger equation using the Lanczos procedure was

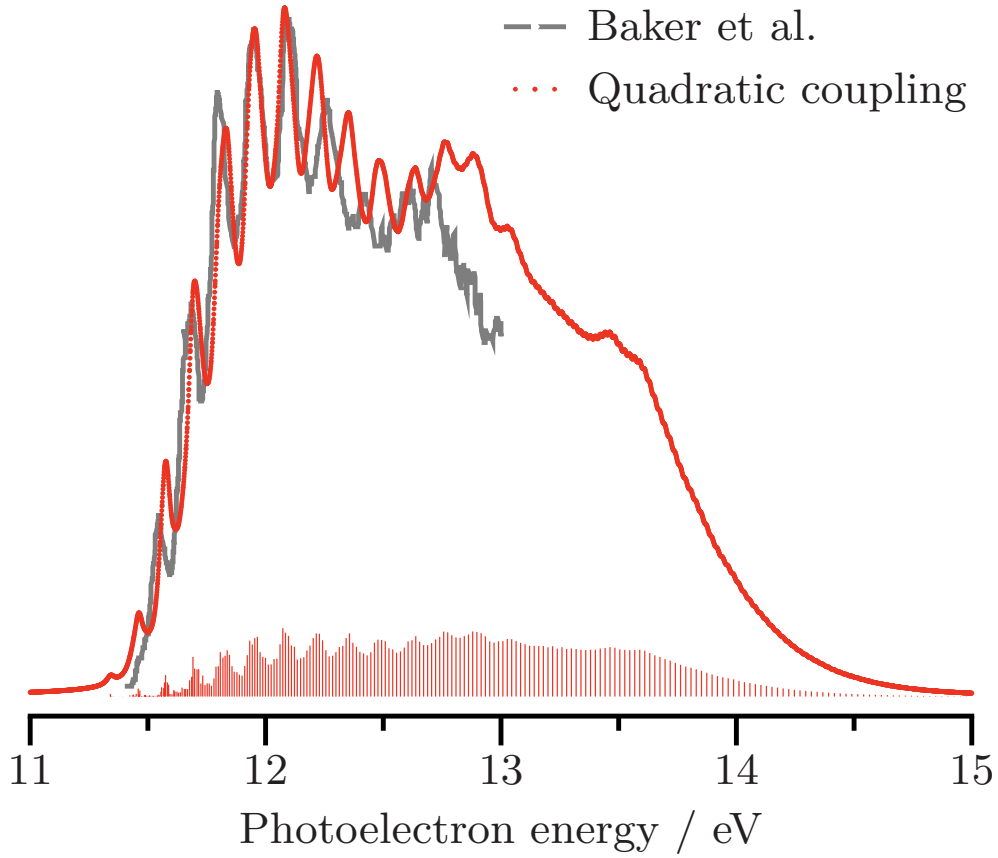


Figure 2.21: Comparison of the simulated spectrum using the QVC model to the observed spectrum. Image taken from [62].

described.

The actual implementation of the described Lanczos procedure can be computationally infeasible on a single machine for molecules larger than three or four atoms due to the memory requirements and computational workloads that the large direct-product vibronic basis requires. To extend the reach of the KDC/Lanczos procedure, I have developed a parallel implementaion of the matrix-vector multiplication operation used in the Lanczos algorithm, which I describe throughout the majority of this chapter.

With a parallel implementation, I have demonstrated that extremely large-scale vibronic coupling calculations are possible, using direct-product basis sets at least an order of magnitude larger than those that have been employed in previous work. The results of this research will therefore allow the KDC vibronic model, which has been shown to provide reliable energies and intensities suitable for the near-quantitative simulation of molecular spectra, to be applied to systems larger than those previously treated. In addition to extending the size of the molecule that can be investigated, the ability to do calculations with $\sim$100 billion or so basis functions will obviate the need to simplify spectral investigations *via* "dropping" possibly unimportant modes or treating them with an inadequate number of basis functions. The importance of extending the range of such calculations in this way was first addressed by Schuurman and Yarkony [90]; I have extended their work by developing a new parallel algorithm which employs coarse-grained parallelism to achieve impressive scalability on distributed memory architectures, as shown in my examples.

Chapter 3

The vibronic spectrum of *trans*-1,3-butadiene

The structure and spectroscopy of *trans*-1,3-butadiene have been well-studied by experimentalists and theorists alike. Experimentalists have probed its geometry [41, 18, 12, 60], its vibrational spectrum [8, 36] and its electronic spectrum [43, 64, 16, 17, 66, 68, 69, 67, 26, 27, 70, 85, 49, 101, 25, 72, 34, 100, 82, 102, 99, 78, 44, 45, 46]. The interest in butadiene is driven by its small size and relative simplicity, along with its importance as the simplest example of a conjugated hydrocarbon. For a long period, it was believed that the molecule was quite well understood, but further investigation began to suggest otherwise. As McDiarmid noted [67], "The subject, extensively studied by the 1950s, was believed to be fundamentally understood except for a few 'anomalies.'" However, these anomalies became of interest as a result of the work of Hudson and Kohler on the "doubly excited state" of longer polyenes [44, 45], the lack of fluorescence observed from butadiene's $\pi \rightarrow \pi^*$ 1^1B_u excitation, and the broad absorption feature of the 1^1B_u state, even at low temperatures [64]. As a result, the "simplicity" of butadiene was no longer self-evident. Among the questions surrounding the electronic spectroscopy of butadiene are: 1) the position (vertical energy) of the assumed (dark) "doubly excited state" (2^1A_g state) and 2) the reasons for the broad feature in the spectrum assigned to the 1^1B_u transition.

1. The vertical energy of the 2^1A_g state has proved to be quite difficult to pin down using *ab initio* electronic structure calculations. A sample of results from the (vast) body of existing theoretical work on the vertical energies of the low-lying states of butadiene is displayed in Table 3.1. It can be seen that the calculated vertical energies for the 2^1A_g state range from 6.23 eV all the way to 7.23 eV, which is hardly definitive.

2. Spectral line broadening can occur for many reasons, the two most common of which are natural broadening (broadening caused by the lifetime of an excited state) and thermal Doppler broadening (broadening due to the Doppler effect at non-zero temperatures). Because the broad absorption feature of the 1^1B_u state exists even at low temperatures, it is likely that natural broadening is the cause of the broad 1^1B_u feature. Natural broadening occurs due to the finite lifetime of excited states, coupled with the uncertainty principle. Since $\Delta E \Delta t > h$, the shorter the lifetime of an excited level Δt, the greater the uncertainty in the energy ΔE of that level, causing a broadening effect. Even if we agree that natural broadening is the cause of the 1^1B_u feature, though, it is not immediately clear why the 1^1B_u state has such a short lifetime. Previous studies have hypothesized that a conical intersection (crossing of potential surfaces caused by vibronic coupling) with the dark 2^1A_g state may to blame, as a conical intersection would allow the molecule to "jump" the from 1^1B_u state to the 2^1A_g without emitting radiation, but no definitive answers have been determined.

Method	Ref.	Δ E(1^1B_u)	Δ E(2^1A_g)	Δ E(1^1B_g)	Δ E(1^1A_u)
MRSDCI+Q	[21]	6.35	6.64	6.29	6.58
MRAQCC	[21]	6.26	6.51		
CASPT2(D/F)	[91]	6.12/6.23	6.23/6.27	6.11/6.29	6.38/6.56
EOM-CCSD		6.42	7.23	6.40	6.61
EOM-CCSD(T)	[104]	6.36	6.92	6.36	6.57
CC2LR	[63]	6.16	7.08	6.25	6.56
CC3LR	[63]	6.20	6.62	6.32	6.63
TDDFT	[63]	5.57	6.49	5.69	5.94
CASPT2	[57]	6.20	6.43		
EOM-CCSD/ CCSDT	[103]	6.21	6.39		
Full CI	[19]	6.30			

Table 3.1: Selected theoretical results for low-lying excited states of *trans*-1,3-butadiene

Both of these questions can be investigated further by simulating the absorption spectrum of *trans*-1,3-butadiene and comparing the results against experimental measurements. The most accurate experimental measurement of the spectrum of butadiene around the 1^1B_u transition energy region was measured by Vaida *et al.* [64] in 1984, and it is the experimental spectrum to which I will compare my simulations in this work.

The work of Domcke and coworkers [77, 57] was the first attempt to simulate the UV spectrum of butadiene. It used CASPT2 results for excitation energies and the Köppel, Domcke, Cederbaum (KDC) vibronic coupling model [15, 53, 55] for the calculation of the electronic spectrum. They included the 1^1B_u and 2^1A_g states in their model and focused on six a_g vibrational modes and a single b_u vibrational mode for their vibrational basis (the vertical excitations from the CASPT2

results are those included in Table 3.1).

In this work, I build on the work of Domcke *et al.* [77, 57] and address a number of important outstanding questions related to the spectrum of butadiene, including:

1. Is it necessary to artificially shift the vertical excitation energy of the 1^1B_u to achieve agreement with the experimental spectrum? The spectrum computed by Domcke and coworkers required an artificial shift to align its first peak with the first peak of the experimental spectrum, but perhaps the inclusion of additional vibrational modes in the KDC model would obviate the need to shift.

2. How important is the presence of the 2^1A_g state to the shape and position of the observed spectrum? Previous studies have implicated a conical intersection (CI) in the lack of 1^1B_u fluorescence, and it is likely that the conical intersection is responsible for the short lifetime (spectral breadth) of the 1^1B_u state, but how important is the 2^1A_g state to the appearance of the observed absorption spectrum? Furthermore, a wide array of theoretical values has been obtained for the vertical transition energy of the 2^1A_g state - how sensitive is the UV spectrum to the energy of this state?

3. How important are vibronic coupling effects between the 1^1B_u and 1^1B_g states? The 1^1B_g state is a low-lying state that has yet to be considered in a vibronic coupling spectral model. Its close proximity to the 1^1B_u state (as

suggested by Table 3.1) suggests that vibronic coupling effects between these two states may be important to consider.

In order to address these questions, I utilize equation-of-motion coupled cluster (EOM-CC) methods for electronic structure calculations using a variety of basis sets and levels of excitation, coupled with the KDC method [15] for vibronic coupling to simulate the UV spectrum of *trans*-1,3-butadiene. This study is intended to shed light on the interactions and mechanisms that most strongly influence the overall appearance of the spectrum, as well as to make a cursory analysis of the question of the "right" vertical excitation energies.

3.1 The KDC Hamiltonian

Vibronic coupling effects between the bright 1^1B_u state and the energetically close 2^1A_g and 1^1B_g states are treated using the quasidiabatic model Hamiltonian popularized by Köppel, Domcke, and Cederbaum (KDC) [54]. The KDC approach to vibronic coupling problems has been described in detail in Chapter 2, so I will only briefly revisit it here. We begin with the time-independent molecular Schrödinger equation,

$$\hat{H}\Psi(\mathbf{r},\mathbf{q}) = E\Psi(\mathbf{r},\mathbf{q}). \tag{3.1}$$

For vibronic coupling problems, it is common to express the molecular wave function $\Psi(\mathbf{r};\mathbf{q})$ in terms of the Born-Huang expansion

$$\Psi(\mathbf{r},\mathbf{q}) = \sum_i \psi_i(\mathbf{r};\mathbf{q})\Omega_i(\mathbf{q}), \tag{3.2}$$

where $\psi_i(\mathbf{r};\mathbf{q})$ are quasidiabatic electronic wave functions and $\Omega_i(\mathbf{q})$ are vibrational wave functions. The summation over i is truncated to only include the relevant coupled electronic states (e.g. $1^1\mathrm{B_u}$, $2^1\mathrm{A_g}$, and $1^1\mathrm{B_g}$, for butadiene). For ease of notation, let butadiene's $1^1\mathrm{B_u}$ state be labeled state A, its $2^1\mathrm{A_g}$ state be labeled state B, and its $1^1\mathrm{B_g}$ state be labeled state C. Substitution of equation 3.2 into equation 3.1, followed by left-multiplication of the electronic basis and integration over $\mathbf{r}$ results in a model Hamiltonian of the form

$$H = \begin{bmatrix} \hat{T}_N & 0 & 0 \\ 0 & \hat{T}_N & 0 \\ 0 & 0 & \hat{T}_N \end{bmatrix} + \begin{bmatrix} V_{AA} & V_{AB} & V_{AC} \\ V_{AB} & V_{BB} & V_{BC} \\ V_{AC} & V_{BC} & V_{CC} \end{bmatrix}. \tag{3.3}$$

The kinetic energy matrix is assumed diagonal (by construction of the quasidiabatic electronic states), and the potential matrix elements $V_{ii'}$ are Taylor expansions in terms of the dimensionless normal coordinates q_i around the absorbing state (for butadiene, the ground state) equilibrium geometry. For our model in particular, the potential elements are of the form

$$V_{ii'} = E(0)_i + \sum_j F_j^{ii'} q_j + \frac{1}{2}\sum_{jk} F_{jk}^{ii'} q_j q_k \qquad \text{for } i \neq i' \tag{3.4}$$

and

$$V_{ii'} = \sum_j F_j^{ii'} q_j \qquad \text{for } i \neq i'. \tag{3.5}$$

The terms $F^{ii'}$ are quasidiabatic potential derivatives, e.g.

$$F_{jk}^{ii'} = \frac{\partial^2}{\partial q_j q_k} \langle \psi_i | \hat{H}_e | \psi_{i'} \rangle, \tag{3.6}$$

and $E(0)_i$ is the vertical energy of state i calculated at the ground-state equilibrium geometry. The computation of the quasidiabatic potential derivatives will be discussed in detail in Section 3.2.2.

It is often the case that the summations over j and k in equations 3.4 and 3.5 do not span the entire set of normal modes. Symmetry properties cause certain derivatives to necessarily be equal to zero. The ground-state equilibrium geometry of butadiene belongs to the C_{2h} symmetry point group, so each of its normal modes belongs to one of the four irreducible representations of the C_{2h} group: a_g, a_u, b_g, and b_u. The derivatives $F_j^{ii'}$ that are necessarily zero are those for which the product of the irreducible representations of ϕ_i, $\phi_{i'}$, and q_j does *not* contain the totally symmetric representation Γ_{a_g}:

$$\Gamma_i \times \Gamma_{q_j} \times \Gamma_{i'} \not\supset \Gamma_{a_g} \tag{3.7}$$

As a result, only the following terms are included in our model:

- F_j^{ii} and F_{jk}^{ii} for each of the totally-symmetric (a_g) modes ($\nu_1 - \nu_9$) for $i = A, B, C$.
- F_{jk}^{ii} for each of the b_u modes ($\nu_{10} - \nu_{17}$), a_u modes ($\nu_{18} - \nu_{21}$), and b_g modes ($\nu_{22} - \nu_{24}$) for $i = A, B, C$.
- F_j^{AB} for each of the b_u modes, and F_j^{AC} for each of the a_u modes.

In addition to removing derivatives from our model due to symmetry considerations, I *effectively* removed certain derivatives to avoid including imaginary quasidiabatic

frequencies in our model. Imaginary frequencies are often caused by the existence of vibronic coupling and are usually resolved by transforming to a quasidiabatic framework. The existence of imaginary *quasidiabatic* frequencies in my calculations, then, suggests the likely presence of vibronic coupling outside of the three states that I currently consider. In my calculations, imaginary quasidiabatic frequencies were found in the set of frequencies for the a_u and b_g modes. As a result, the terms F_{jk}^{ii} for each of the a_u and b_g modes for all three electronic states were set to their ground-state values, effectively removing them from the model.

3.2 Quantum chemical calculations

All excited-state energies and properties were evaluated using the equation-of-motion coupled cluster approach [97], and all coupled-cluster calculations were performed using the CFOUR [94] computational chemistry program package. In all of the reported calculations, I excluded excitations from the core orbitals of the carbon atoms. For each quantum chemical calculation, one of three basis sets was used:

1. The augmented Correlation Consistent aug-cc-pVDZ basis [106], augmented with diffuse functions on C (s,p,d exponents of 0.023, 0.021, 0.015, respectively), denoted simply as aPVDZ.

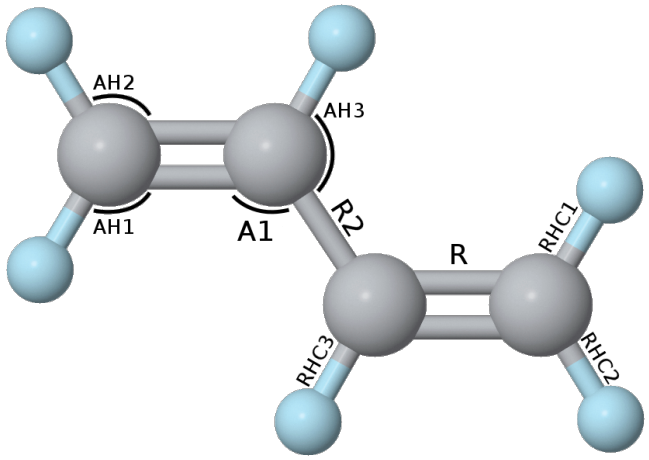
2. The augmented Correlation Consistent aug-cc-pVTZ basis [106], augmented with diffuse functions on C (s,p,d exponents of 0.023, 0.021, 0.015, respectively), denoted simply as aPVTZ.

3. The augmented Correlation Consistent aug-cc-pVQZ basis [106], augmented with diffuse functions on C (s,p,d exponents of 0.023, 0.021, 0.015, respectively), denoted simply as aPVQZ.

The ground-state geometry used in the following calculations was optimized [98] with CC3 [52] using the aPVTZ basis set. The resulting geometry can be found in Figure 3.1 under the 1^1A_g column.

3.2.1 Vertical energy calculations

The vertical energy calculations performed in this work aim to accurately determine the vertical energies of the lowest-lying states of each symmetry and to investigate how increasing levels of theory and basis size affect the calculated vertical energies. In my calculations, three different levels are theory were used (EOM-CCSD, EOM-CC3 [50], EOM-CCSDT [9, 56, 58]). The aPVDZ and aPVTZ bases each were used for all three levels of theory, while the aPVQZ basis was only used with EOM-CCSD due to its large size. The calculated vertical energies can be found in Table 3.2. These results show that the basis set is of little impact with respect to the excitation energies for the 1^1B_u and 2^1A_g states, but the basis set may have some significance for the 1^1B_g and 1^1A_u states. The change from aPVDZ to aPVTZ increases Δ E (1^1B_g) and Δ E (1^1A_u) by roughly 0.1 eV for all three levels of theory; the change from aPVTZ to aPVQZ, however, has little impact for the EOM-CCSD calculations. The results also show that the 2^1A_g state is very sensitive to the inclusion of triple excitations (dropping in VE by roughly 0.4 eV moving from EOM-CCSD to EOM-CC3), and it is even impacted by the difference



	1^1A_g	1^1B_u	2^1A_g	1^1B_g
R	1.3454	1.4221	1.4954	1.3894
R2	1.4602	1.3950	1.4054	1.4090
RHC1	1.0854	1.0859	1.0819	1.0851
RHC2	1.0830	1.0831	1.0801	1.0864
RHC3	1.0878	1.0881	1.0847	1.0832
A1	123.637	123.700	122.144	121.139
AH1	120.992	120.330	120.011	120.829
AH2	121.526	120.969	120.796	120.579
AH3	116.669	118.142	119.404	119.159

Figure 3.1: Minimum-energy geometries (restricted to C_{2h} symmetry) for the ground (1^1A_g) state and excited 1^1B_u, 2^1A_g, and 1^1B_g states. The bond lengths are in Angström units, and the angles are in degrees. Each geometry was optimized using (EOM)-CC3 with an aPVTZ basis.

Method	Basis	Δ E (1^1B_u)	Δ E (2^1A_g)	Δ E (1^1B_g)	Δ E (1^1A_u)
EOM-CCSD	aPVDZ	6.32	7.00	6.25	6.57
	aPVTZ	6.31	7.05	6.37	6.68
	aPVQZ	6.31	7.07	6.41	6.72
EOM-CC3	aPVDZ	6.21	6.62	6.21	6.53
	aPVTZ	6.19	6.61	6.30	6.61
EOM-CCSDT	aPVDZ	6.23	6.53	6.23	6.54
	aPVTZ	6.20	6.54	6.31	6.63

Table 3.2: Vertical excitation energies (in eV) for the lowest-lying excited states of each symmetry of *trans*-1,3-butadiene. All calculations were done at the ground-state equilibrium geometry optimized using CC3/aPVTZ.

between EOM-CC3 and EOM-CCSDT (dropping in VE by an additional 0.1 eV).

My most accurate vertical energy calculations (EOM-CCSDT/aPVTZ) are fairly consistent with values obtained by Watson and Chan in their recent work [103], lending credence to my results (and theirs). In their work, Watson et al. begin with an EOM-CCSD level of theory and add estimated corrections for the inclusion of triple and quadruple excitations. After including their corrections, Watson et al. report vertical energy values of 6.21 ± 0.02 eV and 6.39 ± 0.07 eV for the 1^1B_u and 2^1A_g states, respectively. My Δ E (1^1B_u) values are in complete agreement with their final numbers, and while it looks like our Δ E (2^1A_g) values may differ, the Watson and Chan value *before adding quadruple excitations* is 6.47 ± 0.03 eV, which agrees fairly well with our EOM-CCSDT calculations. There is reason to believe, then, that the actual value of Δ E (2^1A_g) may be lower than our most accurate value by roughly 0.1 eV and that the addition of quadruple excitations might

reveal this change.

3.2.2 KDC Hamiltonian parameterization

All parameters for the KDC vibronic coupling model used in this work were obtained using CC3/EOM-CC3 methods with the aPVTZ basis set. The linear and quadratic coupling constants (F_j^{AB} and F_j^{AC}) were calculated using the BD method developed by Cave and Stanton [13]. A detailed explanation of the BD method and my implementation of it can be found in Appendix B.

There are two commonly used approaches for parameterizing the KDC model Hamiltonian: 1) vertical parameterization and 2) adiabatic parameterization [47]. To perform a vertical parameterization, all parameters of the model ($E(0)_i$ and $F_{jk\ldots}^{ii'}$) are calculated *at the absorbing state equilibrium geometry*. This approach is the simplest, and as such, is the most common approach for parameterization. If the coupled electronic states have minimum-energy geometries that are significantly displaced from the absorbing state geometry, however, it can be difficult to accurately characterize their potential surfaces using calculations performed at the absorbing state geometry. In these situations, it may be best to calculate potential force constants for each of the excited electronic states *at their own minimum-energy geometries*. The constants $F_{jk\ldots}^{ii'}$ that parameterize the KDC model (which are derivatives at the absorbing state geometry) can then be effectively determined using straightforward linear transformations [51]. This concept is the basis of what is known as an adiabatic parameterization of the KDC model.

Since two of the three coupled electronic states of butadiene, 1^1B_u and 2^1A_g,

a_g modes		b_u modes		a_u and b_g modes	
Mode	Frequency	Mode	Frequency	Mode	Frequency
ν_1	507	ν_{10}	287	ν_{18}	165
ν_2	896	ν_{11}	991	ν_{19}	531
ν_3	1221	ν_{12}	1308	ν_{20}	917
ν_4	1305	ν_{13}	1406	ν_{21}	1032
ν_5	1472	ν_{14}	1626	ν_{22}	762
ν_6	1685	ν_{15}	3134	ν_{23}	918
ν_7	3132	ν_{16}	3148	ν_{24}	978
ν_8	3143	ν_{17}	3231		
ν_9	3230				

Table 3.3: Frequencies (in cm^{-1}) of the ground state normal modes

have minimum-energy planar geometries that are significantly displaced from the ground 1^1A_g equilibrium geometry (see Figure 3.1), I chose to use an adiabatic parameterization approach to create my KDC model. The full adiabatic parameterization of a KDC model is performed through the following steps:

1. Determine the minimum-energy geometry of the absorbing state in the totally-symmetric subspace of nuclear displacements and calculate the absorbing state normal modes q_j (see Table 3.3).

2. Determine the minimum-energy geometries for each of the excited electronic states in the totally-symmetric subspace of nuclear displacements (see Figure 3.1). For *trans*-1,3-butadiene, *planar* stationary points are calculated,

as the excited states are not stable to out-of-plane twisting (see discussion in Section 3.2.3). To improve readability, for the remainder of this section, I will use the phrase "excited-state geometry" in place of "excited-state minimum-energy planar geometry". With these geometries in hand, we can calculate the displacements d_j^i of the geometry of state i from the absorbing state geometry along each of the normal modes q_j.

3. Calculate *adiabatic* electronic energies at small perturbations around each excited state geometry. These energies, used in finite-difference formulas, will give us a harmonic approximation to the adiabatic potential of each electronic state. The resulting force constants will be labeled f_{jk}^{ii} (see Table 3.4; for totally-symmetric modes, quasidiabatic force constants F_{jk}^{ii} are identical to their adiabatic counterparts)). Even though these force constants are calculated at the geometry of state i, under the harmonic approximation, the force constants are the same as those calculated at any other geometry (such as the absorbing state geometry, as is needed for the KDC model).

4. Use the displacements d_j^i and force constants f_{jk}^{ii} to calculate the *effective* potential gradients F_j^{ii} at the absorbing state geometry (see Table 3.5) via the following equation:

$$F_j^{ii} = -\sum_k f_{jk}^{ii} d_k^i \tag{3.8}$$

5. For each pair of coupled states i and i', *at each excited state geometry*, calculate quasidiabatic coupling constants ($F_j^{ii'}$, where $i \neq i'$) using the BD

Force constant	$\mathbf{1^1B_u}$	$\mathbf{2^1A_g}$	$\mathbf{1^1B_g}$
$F_{1,1}$	478.41	375.22	491.41
$F_{1,2}$	0.39	4.25	-2.54
$F_{1,3}$	0.94	-66.24	-3.18
$F_{1,4}$	5.14	13.17	0.72
$F_{1,5}$	14.57	33.30	-11.95
$F_{1,6}$	-43.37	-105.39	-22.30
$F_{2,2}$	986.97	979.55	1000.52
$F_{2,3}$	-174.28	-205.92	-110.96
$F_{2,4}$	-13.14	-31.05	11.31
$F_{2,5}$	78.33	97.06	63.49
$F_{2,6}$	-213.67	-292.10	-129.77
$F_{3,3}$	1378.50	1343.42	1365.65
$F_{3,4}$	-11.30	5.21	-18.03
$F_{3,5}$	-90.77	-90.18	-37.35
$F_{3,6}$	132.34	132.07	49.77
$F_{4,4}$	1200.60	1125.10	1295.03
$F_{4,5}$	18.15	27.67	37.71
$F_{4,6}$	-141.21	-235.56	-114.03
$F_{5,5}$	1472.79	1465.91	1500.39
$F_{5,6}$	-71.29	-54.72	-39.53
$F_{6,6}$	1518.17	1364.55	1584.79

Table 3.4: Force constants F_{jk}^{ii} (in cm^{-1}) of each electronic state for the totally-symmetric ($\mathrm{a_g}$) modes.

Mode	Frequency	1^1B_u	2^1A_g	1^1B_g
$1a_g$	507	171	103	-201
$2a_g$	896	-422	-101	-403
$3a_g$	1221	1074	959	375
$4a_g$	1305	677	1241	546
$5a_g$	1472	-626	-706	-327
$6a_g$	1685	2338	3073	1644
$7a_g$	3132	161	294	-62
$8a_g$	3143	110	451	215
$9a_g$	3230	36	70	-107

Table 3.5: Effective potential gradients (F_j^{ii}) (in cm^{-1}) along each of the totally-symmetric modes for each electronic state.

method [13] (see Tables 3.6 and 3.7). I will label coupling constants computed at the state i geometry as $F_j^{ii'(i)}$ and those computed at the state i' geometry as $F_j^{ii'(i')}$.

6. For each pair of coupled states i and i', *at each excited state geometry*, calculate the excitation energy of *both* states (see Table 3.8). The energy of state i at the geometry of state i' will be labeled $E(i')_i$.

7. Calculate *quasidiabatic* force constants (F_{jk}^{ii}; see Table 3.9) for all non-totally-symmetric modes using the formulas

$$F_{jk}^{ii} = f_{jk}^{ii} + \frac{2 * F_j^{ii'(i)} * F_k^{ii'(i)}}{E(i)_{i'} - E(i)_i} \tag{3.9}$$

and

$$F_{jk}^{i'i'} = f_{jk}^{i'i'} + \frac{2 * F_j^{ii'(i')} * F_k^{ii'(i')}}{E(i')_i - E(i')_{i'}} \tag{3.10}$$

Mode	Frequency	F_j^{AB} at 1^1B_u geom.	F_j^{AB} at 2^1A_g geom.
$1b_u$	287	636	620
$2b_u$	991	153	164
$3b_u$	1308	-260	-214
$4b_u$	1406	-12	-26
$5b_u$	1626	200	234
$6b_u$	3134	101	119
$7b_u$	3148	-124	-126
$8b_u$	3231	25	35

Table 3.6: F_j^{AB} linear coupling term values (in cm^{-1}) for each b_u mode, calculated at the 1^1B_u and 2^1A_g minimum-energy planar geometries.

Mode	Frequency	F_j^{AC} at 1^1B_u geom.	F_j^{AC} at 1^1B_g geom.
$1a_u$	165	2	11
$2a_u$	531	470	462
$3a_u$	917	181	174
$4a_u$	1032	113	136

Table 3.7: F_j^{AC} linear coupling term values (in cm^{-1}) for each a_u mode, calculated at the 1^1B_u and 1^1B_g minimum-energy planar geometries.

State	Vertical energy at minimum-energy planar geometry for state: Ground	1^1B_u	2^1A_g	1^1B_g	AP VEs
1^1B_u	49886	44563	41387	46523	49886
2^1A_g	53341	45499	39267	——	52804
1^1B_g	50802	47595	——	48439	50941

Table 3.8: Vertical energies $E(q)_i$ (in cm^{-1}) for each electronic state at the minimum-energy planar geometries of each electronic state.

Force constant	$\mathbf{1^1B_u}$	$\mathbf{2^1A_g}$	$\mathbf{1^1B_g}$
$F_{10,10}$	268.22	264.03	239.05
$F_{10,11}$	-19.46	-41.17	10.61
$F_{10,12}$	59.28	110.46	45.01
$F_{10,13}$	19.98	9.51	-13.14
$F_{10,14}$	55.88	45.76	-48.47
$F_{11,11}$	952.66	916.69	997.78
$F_{11,12}$	10.59	24.06	23.12
$F_{11,13}$	-0.81	-3.88	-11.50
$F_{11,14}$	-17.43	-20.44	-22.31
$F_{12,12}$	1228.85	1194.08	1333.22
$F_{12,13}$	-73.70	-68.02	-44.24
$F_{12,14}$	-139.72	-131.69	-79.76
$F_{13,13}$	1234.38	1226.02	1320.35
$F_{13,14}$	-256.59	-244.92	-162.47
$F_{14,14}$	1086.90	1134.03	1286.82

Table 3.9: Force constants F_{jk}^{ii} (in cm^{-1}) of each electronic state for the b_u modes.

I chose to use the above method for computing quasidiabatic force constants, but it is worth noting that the BD method could also have been used and arrived at near-identical results.

8. *Effectively* calculate gaps between the excited state vertical energies *at the absorbing state geometry* ($E(0)_i - E(0)_{i'}$; see Table 3.8). The term "effectively" is used because the gaps are not determined by calculating electronic energies at the absorbing state geometry, but rather by extrapolating from the gaps calculated at the geometry of the excited state of spectroscopic interest,

$E(i)_i - E(i)_{i'}$.[1] By using F^{ii}_{jk} and $F^{i'i'}_{jk}$ and d^i_j, we can determine the values $E(0)_i - E(0)_{i'}$ that result in the calculated gaps $E(A)_i - E(A)_{i'}$, given the calculated quasidiabatic potential surfaces. It is interesting to note that the gaps produced by this method (see column AP VEs in Table 3.8) are fairly close to the vertical energies calculated at the ground state geometry, lending extra credence to the model (the largest difference is less than 0.1 eV).

The discussion above summarizes the procedure used in the adiabatic parameterization for the vibronic Hamiltonian comprising the three lowest electronic states of trans-butadiene.

3.2.3 The existence of additional B_g coupling

In Section 3.1, I mentioned that my calculations resulted in imaginary quasidiabatic frequencies for the a_u modes (which couple the 1^1B_u state to states of B_g symmetry) and that this phenomenon can be the result of vibronic coupling not accounted for in our model. For the 1^1B_u state frequencies, specifically, the a_u imaginary frequency likely exists because there exist states of B_g symmetry other than 1^1B_g that couple with the 1^1B_u state.

To test this theory, I computed quasidiabatic frequencies of the 1^1B_u state along the a_u modes using the BD method (using EOM-CCSD level of theory with a aPVDZ basis[2]) that account for coupling between the 1^1B_u state and the four

[1] In this case, $E(A)_i - E(A)_{i'}$, since the bright 1^1B_u state is the one of interest

[2] I believe that this level of theory and choice of basis, while not completely robust, is enough for a cursory look into the existence of additional B_g coupling.

Mode	Adiabatic frequencies	Quasidiabatic frequencies using up to			
		1^1B_g	2^1B_g	3^1B_g	4^1B_g
$1a_u$	108	216	376	378	386
$2a_u$	377 i	244 i	243 i	223 i	222 i
$3a_u$	767	786	796	796	796
$4a_u$	871	915	920	920	920

Table 3.10: Adiabatic and quasidiabatic frequencies (in cm^{-1}) of the a_u modes in the 1^1B_u state. Each quasidiabatic frequency was calculated with the BD method, accounting for an increasing number of states of B_g symmetry. For example, the quasidiabatic frequencies in the 2^1B_g column were calculated by accounting for coupling between 1^1B_u, 1^1B_g, and 2^1B_g.

lowest-energy B_g states. The calculated quasidiabatic frequencies are shown in Table 3.10.

It can be seen that the inclusion of each successive B_g state does, in fact, change the quasidiabatic frequencies, suggesting that these higher-energy B_g states may be important to include for a "complete" model. It is also worth noting that each additional state reduces the magnitude of the imaginary frequency, even though it is never made real. It might well be that there is a much higher-energy B_g state with *very* strong coupling that is the "cause" of the imaginary frequency, but a search for this state is beyond the scope of this work. This discussion is meant to highlight the fact that the vibronic coupling model used in this work should in no way be considered complete. It appears as though there exists more vibronic coupling than I account for in my model, and it is important to note this before moving forward.

3.3 Vibronic calculations

To solve for the vibronic energies and nuclear wave functions of the model system, we must solve the nuclear Schrödinger equation

$$H \begin{bmatrix} \Omega_A(\mathbf{q}) \\ \Omega_B(\mathbf{q}) \\ \Omega_C(\mathbf{q}) \end{bmatrix} = E \begin{bmatrix} \Omega_A(\mathbf{q}) \\ \Omega_B(\mathbf{q}) \\ \Omega_C(\mathbf{q}) \end{bmatrix}, \tag{3.11}$$

where H is the KDC model Hamiltonian defined in equation 3.3. To solve the resulting nuclear Schrödinger equation, it is common to use a variational approach in which the functions $\Omega_i(\mathbf{q})$ are assumed to be contained in the space spanned by the basis of an m-dimensional harmonic oscillator defined by the absorbing state potential energy surface and centered at the absorbing state equilibrium geometry:

$$\Omega_i(\mathbf{q}) = \sum_{\vec{v}} c^i_{\vec{v}} \cdot \chi_{\vec{v}}(\mathbf{q}) \tag{3.12}$$

$$= \sum_{v_1} \cdots \sum_{v_m} c^i_{\vec{v}} \cdot \chi_{v_1}(q_1) \cdots \chi_{v_m}(q_m) \quad \text{where} \quad v_j \in \{0, 1, ...\}. \tag{3.13}$$

Each block of the Hamiltonian in equation 3.3 is then projected onto this basis, resulting in a sparse matrix whose eigenvalues approximate the vibronic energy levels of the system and whose eigenvectors can be used to determine the relative intensities of each energy level. In theory, the harmonic oscillator basis is of infinite size. To produce a matrix of finite dimension, the basis must be truncated. Variables n_j are used to set a maximum number of possible quanta for each one-dimensional harmonic oscillator function in the basis:

$$\Omega_i(\mathbf{q}) = \sum_{v_1} \cdots \sum_{v_m} c^i_{\vec{v}} \cdot \chi_{v_1}(q_1) \cdots \chi_{v_m}(q_m) \quad \text{where} \quad v_j \in \{0, 1, ..., n_j - 1\}. \tag{3.14}$$

The eigenstates of the resulting sparse matrix are computed using the Lanczos iterative procedure [61].

All vibronic calculations in this work were performed using a parallel implementation of the Lanczos iterative algorithm via the xsim_hpc module of CFOUR, developed by Rabidoux *et al.* [79]. A parallel implementation was required due to the large size of the vibrational basis and the large number of calculations that had to be performed for this study.

The size of the basis set (values of n_j; see equation 3.14) used for the vibronic coupling calculations can be found in Table 3.11. The vibrational basis used is of dimension 597 million, which results in a full Hamiltonian matrix of dimension 1.8 billion. To be sure that the basis set used was large enough to produce a converged spectral envelope, each of the 21 values of n_j were individually increased, and spectra were calculated using each of these new, larger bases. No noticeable change in the spectral envelope was seen, leading me to conclude that the basis in Table 3.11 is sufficient. Each vibronic calculation required 1000 iterations of the Lanczos algorithm, and, using 64 cores of the Stampede supercomputer, took roughly 3.5 hours to complete. It is worth noting that larger calculations all well within the reach of the xsim_hpc module (see Section 2.6) and will likely be necessary for a model in which all coupled electronic states are accounted for and accurate quasidiabatic surfaces for the non-planar modes are used.

a_g modes		b_u modes		a_u modes		b_g modes	
n_1	5	n_{10}	12	n_{18}	1	n_{22}	1
n_2	6	n_{11}	2	n_{19}	4	n_{23}	1
n_3	6	n_{12}	2	n_{20}	1	n_{24}	1
n_4	10	n_{13}	2	n_{21}	1		
n_5	3	n_{14}	4				
n_6	18	n_{15}	1				
n_7	2	n_{16}	1				
n_8	2	n_{17}	1				
n_9	1						

Table 3.11: Number of quanta allowed for each mode in the vibrational basis used for xsim_hpc calculations

3.3.1 Results

I will now discuss what new evidence my vibronic calculations bring about in relation to the three questions posed in beginning of this chapter. My attempts to answer the aforementioned questions required the calculation of spectra for many KDC models with varying parameters. In order to compare all of the calculated spectra, I have characterized each of them using a handful of easily-determinable properties:

Band I position	The energy (in cm^{-1}) of the first peak of the spectral envelope.
Intervals	The energy gaps (in cm^{-1}) between the peaks I and II, II and III, and I and III of the spectral envelope.
Intensities	The intensities of each of the three peaks (the second peak is normalized to 1.0).

These properties were chosen because they were used by Vaida *et al.* in the characterization of their experimental spectrum.

An important note about the creation of theoretical spectra: A time-independent vibronic calculation merely produces a stick spectrum. To compare against an experimental spectrum, it is necessary to create a spectral envelope by artificially broadening the lines of stick spectrum. This broadening is done by convolving the stick spectrum with Gaussian functions of a specified linewidth. The properties in the table above that will be used to compare spectra are *very* dependent on the linewidth used, so, in order to fairly compare theoretical spectra with the observed data (or with other theoretical spectra, for that matter), it is important to maintain some form of consistency between spectral envelopes. To this end, the linewidth used for each spectrum was chosen so that the resulting envelope has a distance in intensity between the top of the first peak and the bottom of the first valley roughly equal to that seen in the observed data (0.1843). For each reported calculation, the linewidth (in eV) used to produce the corresponding spectral envelope is also reported.

In this section, there are several figures that are used to compare either theoretical spectra with experimental spectra or to compare two (or more) theoretical

spectra with different sets of parameters. In each of these figures, **all plotted spectra have been shifted so that their origin peaks are at the same energy**. This artificial shift is used to better showcase the changes in spectral *shape* that occur with varying parameters. Information regarding *raw* origin peak energies can be found in each figure's corresponding table.

Before diving into the three questions posed at the beginning of this chapter, let me briefly compare the theoretical spectrum calculated using the model parameters discussed above and the data observed by Vaida *et al.* The theoretical spectrum is shown red on top of spectrum observed by Vaida *et al.* in Figure 3.2. A comparison of various properties of the spectral envelope with those measured by Vaida *et al.* can be found in Table 3.12 (compare spectrum $A + B + C$ with Vaida *et al.*). From Table 3.12, it appears that while the model does an adequate job reproducing the first two peaks of the spectrum, the theoretical third peak is significantly lower in intensity and higher in energy than the observed third peak. We will see in the coming sections, though, that the spectrum is quite sensitive to the vertical energies of each electronic state, and adjustments to these values can vastly improve the agreement between simulation and experiment.

3.3.1.1 Vertical energy of the 1^1B_u state

In the beginning of this chapter, I asked the question:

1. Is it necessary to shift the 1^1B_u state's energy in order to achieve good agreement with the experimental UV spectrum?

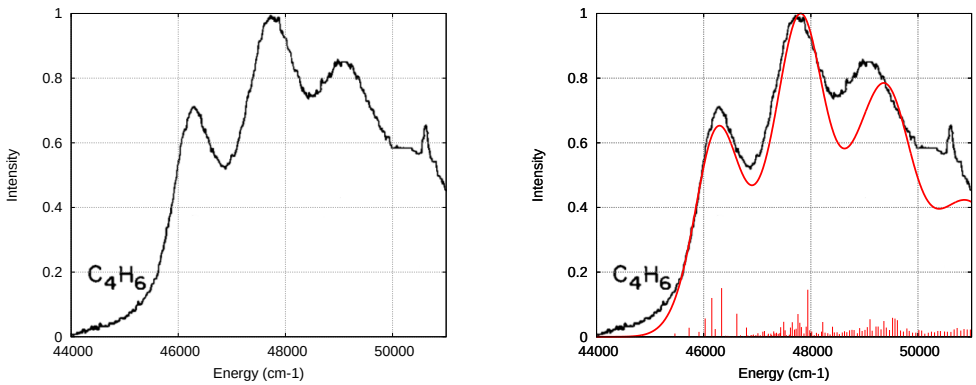


Figure 3.2: Comparison of the three-state vibronic coupling spectrum and the spectrum observed experimentally by Vaida *et al.* The experimental spectrum is shown on the left, and the calculated spectrum is plotted on top of the experimental spectrum on the right. The calculated spectrum was artificially shifted have its origin peak at the same energy as the origin peak of the observed spectrum.

		Band I	Intervals (cm^{-1})			Intensities		
Spectrum		Position	I-II	II-III	I-III	I	II	III
Vaida *et al.*		46260	1460	1340	2800	0.72	1.00	0.86
States	LW (eV)							
A	0.122	47145	1515	1510	3025	0.64	1.00	0.78
$A+B$	0.098	47110	1515	1515	3030	0.65	1.00	0.78
$A+C$	0.123	47110	1510	1540	3050	0.65	1.00	0.79
$A+B+C$	0.099	47070	1515	1550	3065	0.65	1.00	0.79

Table 3.12: Spectral properties for vibronic coupling calculations involving states A, $A+B$, $A+C$, and $A+B+C$, compared to the properties observed by Vaida *et al.* Band position and peak intervals are both in cm^{-1}. Gaussian linewidth (LW) values are in eV.

The spectrum measured by Vaida *et al.* has its origin peak located at 46260 cm^{-1}, while my KDC model produced a spectrum whose origin peak is located at 47070 cm^{-1}; a difference of 810 cm^{-1} or 0.10 eV. The observed difference could be explained by a number of factors:

i) The 1^1B_u energy used in the KDC model is simply too high. It is possible that a larger basis set or a higher level of theory would result in a lower value of Δ E (1^1B_u), but there is not much evidence in the data to support this theory.

ii) Excited state frequencies for the a_u and b_g modes were not included in the KDC model (they were taken to be equal to the ground state frequencies). Including accurate excited state frequencies for these modes would shift the spectrum as a whole, potentially downward if the excited frequencies are less than their ground counterparts.

3.3.1.2 Coupling with the 2^1A_g state

In the beginning of this chapter, two questions were asked related to 2^1A_g coupling:

2a. How important is the presence of the 2^1A_g state to the shape and position of the observed spectrum?

2b. How sensitive is the UV spectrum to the energy of the 2^1A_g state?

Let us begin with the first question. To determine the importance of the 2^1A_g state, I have compared two spectra: the spectrum created using a simple Franck-Condon

calculation for the transition from the ground state to the 1^1B_u state, and the spectrum created using a KDC vibronic coupling model involving only the 1^1B_u and 2^1A_g states. Figure 3.3 shows the change in spectral shape that occurs when accounting for coupling with the 2^1A_g state.

The inclusion of the 2^1A_g state has almost no effect on the shape and position of the spectrum. The exact differences between the two spectra are shown in Table 3.12 (compare A with $A + B$). Both spectra have roughly the same peak intervals and peak intensities. The inclusion of the 2^1A_g state does shift the entire spectrum down by about 35 cm^{-1}, but this effect is insignificant.

There is one interesting difference between the two spectra, though, that can not be seen in Figure 3.3: the linewidth required to create an appropriate spectral envelope. The FC calculation for the 1^1B_u state requires a linewidth of 0.122 eV to create an appropriate spectral envelope, while the $A + B$ KDC calculation only requires a linewidth of 0.098 eV. These results suggest that the 2^1A_g state is important to include to recreate the spectral breadth seen in the observed data; i.e., a conical intersection created by the interaction between the 2^1A_g and 1^1B_u states is likely at least partially responsible for the short lifetime (and thus broad linewidth) seen in the spectrum.

The second question posed deals with the sensitivity of the spectrum in relation to a changing vertical energy of the 2^1A_g state. To investigate this question, I have calculated spectra using a three-state (1^1B_u, 2^1A_g, and 1^1B_g) KDC model with vertical energies of the 2^1A_g state ranging from 2000 cm^{-1} less than the originally calculated value to the originally calculated value, in increments of 200 cm^{-1}. This

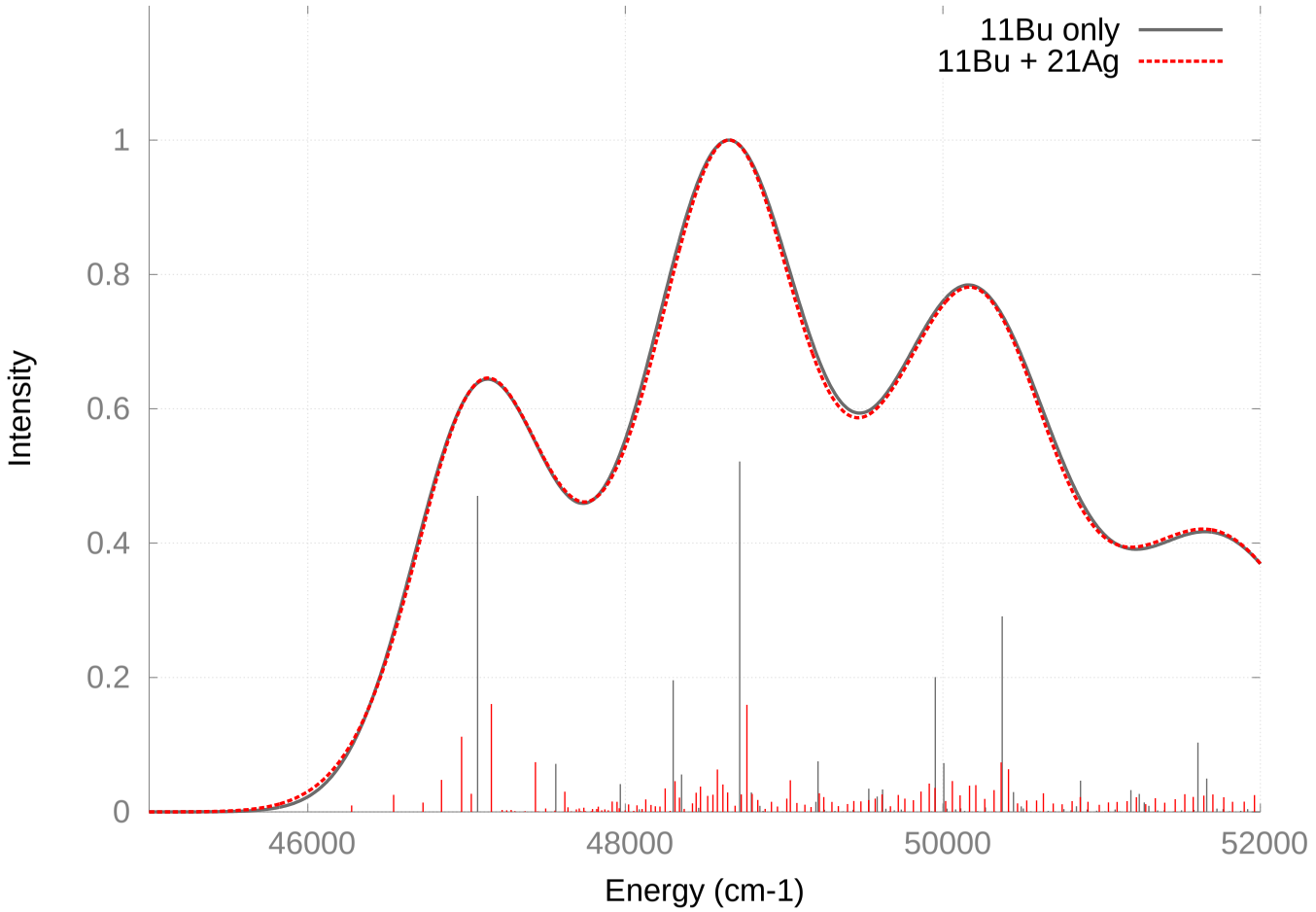


Figure 3.3: Comparison of a two-state (1^1B_u and 2^1A_g) vibronic coupling spectrum and the spectrum produced by a Franck-Condon calculation on the bright 1^1B_u state.

Spectrum		Band I Position	Intervals (cm^{-1}) I-II	II-III	I-III	Intensities I	II	III
Vaida *et al.*		46260	1460	1340	2800	0.72	1.00	0.86
$\Delta E(0)_B$	LW (eV)							
0	0.099	47070	1515	1550	3065	0.65	1.00	0.79
-200	0.099	47080	1515	1550	3065	0.66	1.00	0.79
-400	0.098	47095	1510	1550	3060	0.66	1.00	0.79
-600	0.099	47110	1500	1550	3050	0.66	1.00	0.79
-800	0.100	47125	1490	1540	3030	0.67	1.00	0.79
-1000	0.100	47135	1490	1540	3030	0.67	1.00	0.79
-1200	0.101	47145	1490	1540	3030	0.67	1.00	0.79
-1400	0.103	47150	1490	1540	3030	0.68	1.00	0.79
-1600	0.104	47160	1490	1540	3030	0.68	1.00	0.79
-1800	0.105	47170	1490	1540	3030	0.68	1.00	0.79
-2000	0.106	47170	1500	1540	3040	0.68	1.00	0.79

Table 3.13: Spectrum properties for calculated spectra with increasing vertical energy values for the 2^1A_g state. Band position, vertical energy changes, and peak intervals are all in cm^{-1}. Gaussian linewidth (LW) values are in eV.

range of values was chosen in accordance with changes in Δ E (2^1A_g) suggested by both the calculations in Table 3.2 and the results from Watson and Chan. A complete description of the results can be found in Table 3.13.

These calculations suggest that the spectrum is mostly insensitive to any changes in the 2^1A_g vertical energy. The interval between the first and second gaps of the spectrum does shrink slightly as Δ E (2^1A_g) decreases, until the vertical energy reaches 800 cm^{-1} less than its original value; no further change is observed by lowering Δ E (2^1A_g) more than 800 cm^{-1}. Also, as the vertical energy of 2^1A_g

decreases, the intensity of the first peak rises slightly (from 0.65 to 0.68). Both the I-II interval decrease and first peak intensity increase bring the calculated spectrum closer in agreement to experiment, so a decrease in Δ E (2^1A_g), as predicted by the data, is supported by our spectral calculations as well. A sample of spectra calculated using decreasing values of Δ E (2^1A_g) can be found in Figure 3.4.

3.3.1.3 Coupling with the 1^1B_g state

In the beginning of this chapter, I asked the question

3a. How important are the vibronic coupling effects between the 1^1B_u and 1^1B_g states?

To determine the importance of the 1^1B_g state, I have again compared two spectra: 1) the spectrum created using a simple Franck-Condon calculation for the transition from the ground state to the 1^1B_u state, and 2) the spectrum created using a KDC vibronic coupling model involving only the 1^1B_u and 1^1B_g states. Figure 3.5 shows the change in spectral shape that occurs when accounting for coupling with the 1^1B_g state. The addition of the 1^1B_g state into the KDC model has a (very) small effect on the height and energy of third peak, raising both its intensity (closer to experiment) and its energy (further from experiment). The exact differences between the two spectra are shown in Table 3.12.

From this calculation, it appears as though the 1^1B_g state has little to no effect on the computed spectrum, similar to the conclusion drawn from the same type of calculation performed for the 2^1A_g state. For the 2^1A_g state, though, a

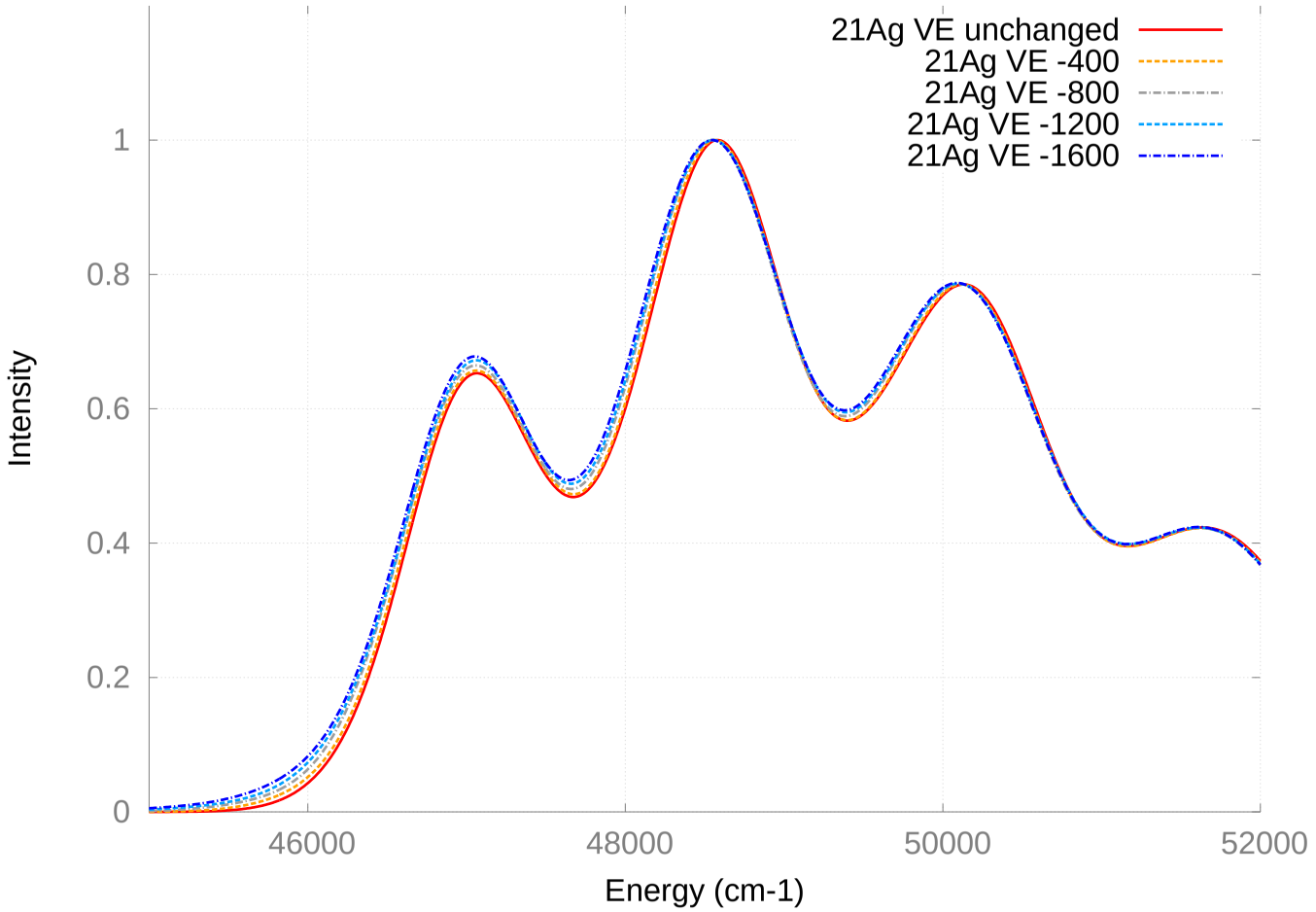


Figure 3.4: Comparison of various three-state vibronic coupling spectra whose 2^1A_g vertical energies have changed from their initial value.

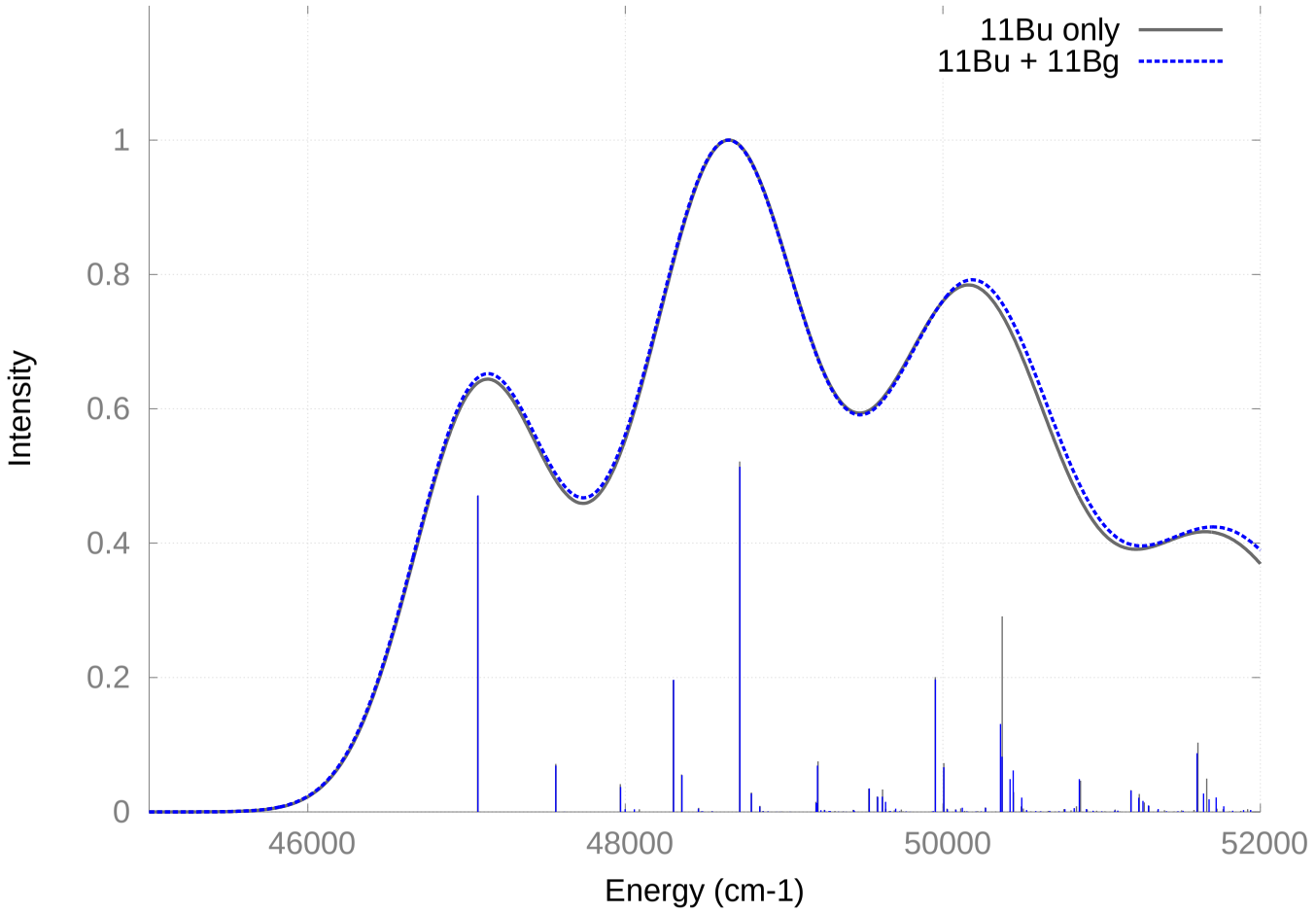


Figure 3.5: Comparison of a two-state (1^1B_u and 1^1B_g) vibronic coupling spectrum and the spectrum produced by a Franck-Condon calculation on the bright 1^1B_u state.

change in vertical energy was required to reveal the true (albeit small) effect of the state on the computed spectrum. I will now perform a similar analysis for the 1^1B_g state:

3b. How sensitive is the UV spectrum to the energy of the 1^1B_g state?

To investigate this question, I have calculated spectra using a three-state (1^1B_u, 2^1A_g, and 1^1B_g) KDC model with vertical energies of the 1^1B_g state ranging from 1400 cm^{-1} less than the originally calculated value to 1000 cm^{-1} more than the originally calculated value, in increments of 200 cm^{-1}. A complete description of the results can be found in Table 3.14.

Surprisingly, it is clear from this data that the spectrum is in fact quite sensitive to the vertical energy of the 1^1B_g state, much more so than for the 2^1A_g state. An increase in Δ E (1^1B_g) has little to no effect on the computed spectrum, so we will ignore positive shifts for the rest of this discussion. A decrease in Δ E (1^1B_g), however, causes significant changes in the computed spectrum that bring it closer in agreement to the observed data:

1. The intensity of the first peak raises slightly with a decrease in Δ E (1^1B_g), and the intensity of third peak raises slightly until Δ E (1^1B_g) is 800 cm^{-1} less than its original value; any further decrease in vertical energy lowers the third peak intensity back to its original value.

2. The interval between the second and third peaks shrinks *significantly*, from 1550 cm^{-1} to 1450 cm^{-1} with a VE decrease of 1200 cm^{-1}.

Spectrum		Band I Position	Intervals (cm^{-1}) I-II	II-III	I-III	Intensities I	II	III
Vaida *et al.*		46260	1460	1340	2800	0.72	1.00	0.86
$\Delta E(0)_C$	LW (eV)							
+1000	0.099	47080	1525	1510	3035	0.65	1.00	0.77
+800	0.099	47080	1515	1510	3025	0.65	1.00	0.77
+600	0.099	47080	1515	1510	3025	0.65	1.00	0.77
+400	0.099	47080	1515	1515	3030	0.65	1.00	0.77
+200	0.099	47080	1515	1530	3045	0.65	1.00	0.77
0	0.099	47070	1515	1550	3065	0.65	1.00	0.79
-200	0.099	47070	1515	1550	3065	0.66	1.00	0.80
-400	0.100	47070	1510	1530	3040	0.66	1.00	0.82
-600	0.100	47070	1510	1500	3010	0.66	1.00	0.82
-800	0.100	47060	1510	1475	2985	0.67	1.00	0.82
-1000	0.101	47060	1500	1460	2960	0.67	1.00	0.81
-1200	0.102	47055	1510	1450	2960	0.68	1.00	0.80
-1400	0.103	47045	1525	1470	2995	0.68	1.00	0.79

Table 3.14: Spectrum properties for calculated spectra with decreasing vertical energy values for the 1^1B_g state. Band position, vertical energy changes, and peak intervals are all in cm^{-1}. Gaussian linewidth (LW) values are in eV.

These effects can be seen in Figure 3.6. One additional effect that is clear from Figure 3.6 but cannot be determined from the data in Table 3.14 is the change in spectral breadth between the second and third peaks that occurs with a lowering of Δ E (1^1B_g). Looking at the Δ E (1^1B_g)-1200 cm^{-1} graph in particular (light blue), the spectral envelope between peaks II and III is much broader than in the original computed spectrum. In Figure 3.2, we can see that the observed data also has this broadness between the second and third peaks.

3.3.2 Discussion: Adjustments to vertical energies

In Sections 3.3.1.2 and 3.3.1.3, we saw that the initial addition of the 2^1A_g and 1^1B_g states to our KDC model had very little impact on the computed spectrum. However, for both states, we observed sensitivity of the spectrum to changes in vertical energy. For the 2^1A_g state, a decrease in VE of anywhere between 800 cm^{-1} and 1600 cm^{-1} reduces the I-II peak interval and raises the intensity of the first peak, bringing the spectrum closer in agreement to the observed data. A change of this magnitude is supported in part by our EOM-CCSDT calculations that report values of Δ E (2^1A_g) 0.07 eV (564 cm^{-1}) lower than their EOM-CC3 counterparts. If we also consider the quadruple excitation correction reported by Watson and Chan (-0.074 eV or 597 cm^{-1}), then our total adjustment becomes roughly 1200 cm^{-1}, which is within the range suggested by the spectral calculations.

The computed spectrum is also sensitive to changes in the 1^1B_g vertical energy, even more so than the 2^1A_g vertical energy. A decrease in Δ E (2^1A_g) results in a *significant* decrease in the II-III interval (from 1550 cm^{-1} down to 1450 cm^{-1}),

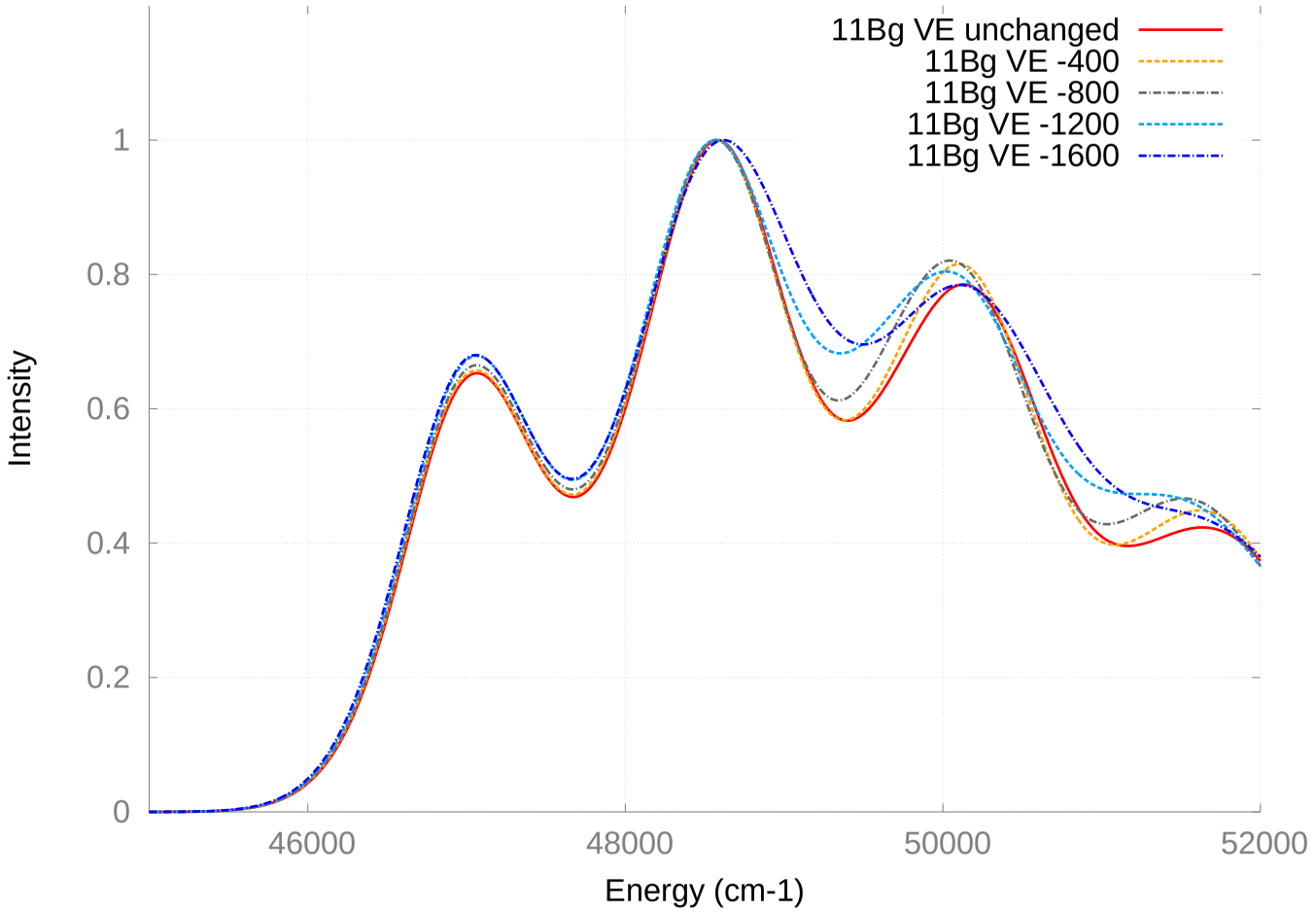


Figure 3.6: Comparison of various three-state vibronic coupling spectra whose 1^1B_g vertical energies have changed from their initial value.

along with a slight increase in the intensity of the first peak. Unfortunately, there is not much evidence in the data to support a lowering of the 1^1B_g vertical energy. In fact, in Table 3.2, an increase in basis size causes an *increase* in Δ E (1^1B_g). It is possible though, that adding more diffuse functions to the aPVTZ basis will significantly lower Δ E (1^1B_g). Using EOM-CCSD, adding the currently included diffuse functions to the aug-cc-pVDZ basis (moving from standard aug-cc-pVDZ to aPVDZ) lowers the vertical energy of 1^1B_g by 0.4 eV, so it is clear that the value of Δ E (1^1B_g) is dependent on whether or not the used basis can capture the state's diffuse nature.

In an attempt to create an "corrected" model spectrum, I implemented changes in the 2^1A_g and 1^1B_g vertical energies that were suggested by my spectral data. I recomputed the spectrum using a value of Δ E (2^1A_g) that was 800 cm^{-1} lower than the original and a value of Δ E (1^1B_g) that was 1200 cm^{-1} lower than the original and compared the results again with the observed data. The results are displayed in Figure 3.7. It is clear that the "corrected" model agrees *much* better with experiment, with an almost exact replication of the observed data for energies up to 48200 cm^{-1}. The computed third peak is still slightly lower in intensity and higher in energy than the experimental third peak, but this difference in significantly smaller than in our original, uncorrected calculation. The breadth of the computed spectrum from 48200 cm^{-1} onwards also agrees fairly well with experiment, in contrast to the original, calculated spectrum.

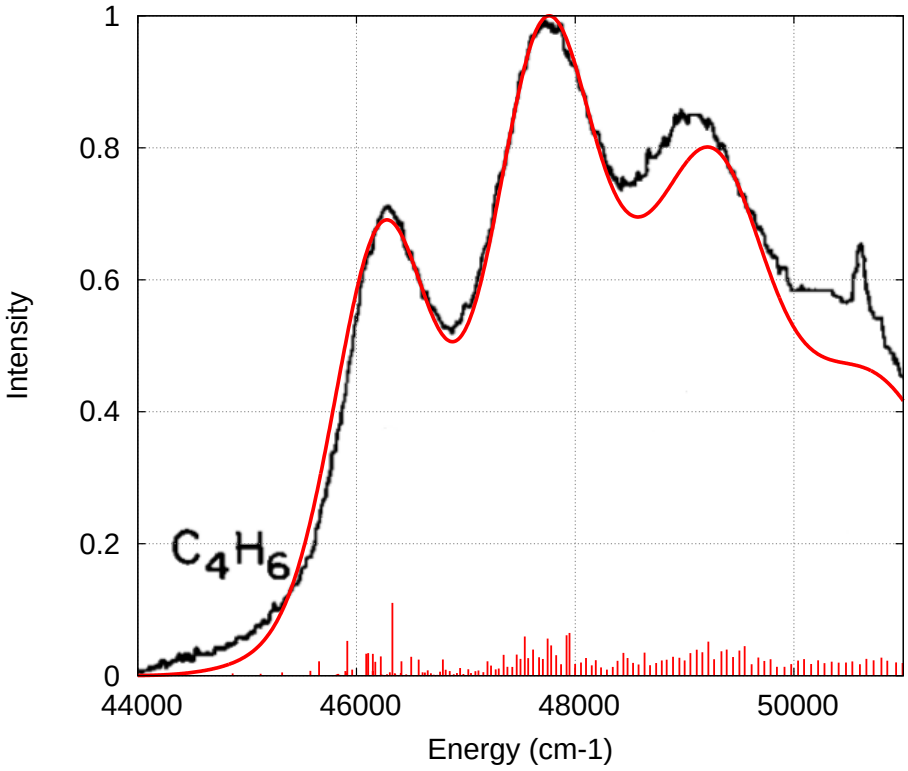


Figure 3.7: Comparison of the three-state vibronic coupling spectrum with adjusted vertical energies (Δ E (2^1A_g) lowered by 800 cm^{-1} and Δ E (1^1B_g) lowered by 1200 cm^{-1}) and the spectrum observed experimentally by Vaida *et al.* The calculated spectrum was artificially shifted have its origin peak at the same energy as the origin peak of the observed spectrum.

3.4 Conclusion and future work

A variety of new insights arise, I believe, from the above calculations on the spectrum of *trans*-1,3-butadiene. I will now summarize the answers to the questions posed in the introduction of this chapter and follow this with possible avenues for related future work. My three questions were:

1. Is it necessary to shift the 1^1B_u state's energy in order to achieve good agreement with the experimental UV spectrum?

2. How important are the presence and position of the 2^1A_g state to the shape and position of the observed spectrum?

3. How important are vibronic coupling effects between the 1^1B_u and 1^1B_g states?

The results presented above provide detailed information to support the answers to these questions, and, on the basis of them, I summarize my findings as follows:

1. It does not appear to be necessary to *significantly* shift the vertical excitation energy of the 1^1B_u state in order to reproduce the experimental spectrum. Within 0.1 eV (or about 2.1 kcal/mole) the spectra calculated above all align with the first peak in the experimental spectrum. This difference can likely be at least partially attributed to the exclusion of upper state frequencies for the a_u and b_g modes.

2. The broad features of the spectrum (qualitative peak heights and widths) appear to be captured relatively well by the basic FC spectrum, and the addition of the 2^1A_g state has very little effect on these features. In addition, the overall spectrum is relatively insensitive to 0.1 eV shifts of the 2^1A_g state vertical energy. The 2^1A_g state does have an important effect on the linewidth, though, producing the broad features of the spectrum using a much smaller artificial linewidth that is required for the FC spectrum. This suggests that 2^1A_g likely plays an role in the rapid decay (short lifetime) of the 1^1B_u state.

3. With respect to the impact of the 1^1B_g state on the spectrum, I was quite surprised. Vibronic coupling with the 1^1B_g state has generally been neglected in discussions of the spectrum of the 1^1B_u state (though the Vaida spectrum does evidence the 1^1B_g state as a sharp peak near 6.2 eV.) Nevertheless, its inclusion in my simulations has a decidedly larger effect on the II-III peak spacing than the 2^1A_g state does. The overall spectral shape is also much more sensitive to the position of the 1^1B_g state than that of the 2^1A_g state. The spectrum improves considerably when one lowers the 1^1B_g state excitation energy, indicating that the calculated aPVTZ value in Table 3.2 may overestimate the actual value.

3.4.1 Future work

The model for *trans*-1,3-butadiene used in this chapter should in no way be considered complete, though it does appear that we are moving in the right direction with the improvements applied to the KDC model in this work. Future related work

can improve upon the methods used here in several ways:

1. Improve basis sets and level of theory used for vertical energy calculations: The vertical energy calculations reported in Table 3.2 do not seem to converge on a given value, save for the 1^1B_u state. Calculating an accurate vertical energy for 2^1A_g will likely require the consideration of quadruple excitations, but an EOM-CCSDTQ/aPVDZ calculation for butadiene is extremely large and would likely require at least several months of computation time. The vertical energies for 1^1B_g and 1^1A_u seem to fluctuate with a growing basis set; perhaps the addition of more diffuse functions to the basis would converge the calculations.

2. Include vibronic coupling effects between electronic states other than 2^1A_g and 1^1B_g: While the 2^1A_g and 1^1B_g states are the two closest in proximity to the bright 1^1B_u, it is likely that coupling with higher-energy B_g states, and possibly A_u states, exists and has an impact on the spectrum. From the results in Table 3.10, the consideration of higher B_g states is necessary if one hopes to produce accurate quasidiabatic potential surfaces along the a_u modes. Accurate upper-state frequencies for a_u and b_g modes could obviate the need for any artificial shift of the computed spectrum.

3. Increase the order of the potential polynomial expansions used in the KDC model: The model used in this work only included up to quadratic terms in the description of the quasidiabatic potential surfaces. Increasing the order of the model to include up to quartic terms in the potential expansions might

improve the accuracy of the resulting spectrum, though parameterization of a quartic model for butadiene is a significant computational endeavor.

Adding more coupled electronic states to the model and increasing the model order will both worsen the computational stresses that currently exist for a KDC/Lanczos calculation. Fortunately, with the parallel implementation described in Chapter 2, the size of future models is only limited by the amount of computing power available, which continues to grow rapidly over time.

Chapter 4

Conclusion

In this work, a new algorithm for the implementation of the Doktorov reccurence equations for Franck-Condon overlap calculation has been detailed. The algorithm improves upon existing methods by significantly reducing the amount of execution time required for the recovery of overlaps from memory. Timing results for several calculations were reported and compared against those for existing codes, revealing an overall run time speedup of over thirty times. Several variants of the algorithm were discussed to show compatibility with prescreening methods, applicability in the case of insufficient available memory, and applicability for finite-temperature calculations. The new algorithm has been used to study the the spectroscopy of C_5H_4O and HC_7H, and the study of HC_7H has been described in detail here.

A parallelization strategy for the matrix-vector multiplication operation used in the implementation of the KDC vibronic coupling model has been developed. Theoretical and experimental measurements of the communication time required by the parallelization strategy have been reported, and various algorithmic properties related to parallelization (data distribution, amount of communication, load imbalance, etc.) have been discussed in detail. The developed parallel implementa-

tion has already been used to apply the KDC model to several previously intractable problems. Related work on the simulation of the photoelectron spectrum of ethane (requiring 2048 cores and 1.7 TB of memory) has been summarized.

Finally, a simulation of the vibronic spectrum of *trans*-1,3-butadiene has been reported. *Ab initio* calculations of the vertical energies, electronic state geometries, and potential energy surfaces were performed for the parameterization of a KDC model. Several questions surrounding butadiene's spectroscopy were investigated, including the importance of the 2^1A_g and 1^1B_g states to the overall shape and position of the spectrum and the spectrum's sensitivity to the vertical energies of those states.

Appendices

Appendix A

Detailed fc_squared code

Below is a complete description (in C code) of the proposed algorithm that includes support for various truncation methods such as maximum sum of quantum numbers, maximum quanta for each mode, maximum energy, and maximum number of simultaneously excited modes.

First, it is useful to show a definition of the Tree_Options structure that will be used throughout the algorithm:

```
struct Tree_Options{
  double max_energy; // maximum allowed energy for target vibrational states
  int* max_quanta; // array of maximum allowed quantum numbers for each vibrational mode
  double* frequency; // array of normal mode frequencies associated with target electronic state
  int max_generation; // maximum sum of quantum numbers for a target vibrational state
  int max_number_of_simultaneously_excited_modes; // maximum number of simultaneously excited modes
  Tree* node_array; // Array of pre-allocated Tree objects
  int* grandnode_modes; // mode numbers associated with each grandnode (updated throughout algorithm)
  double intensity_threshold; // intensity threshold used for choosing which FC overlaps to write to output
};
```

Each line of code in the algorithm below is accompanied by a colored dot on the right hand margin. The color of the dot corresponds to the purpose of the line of code (which truncation method requires the line of code). The colors correspond to truncation methods in the following way:

- ● - Base code
- ●●●● - Code required due to:
  - ● - trunctation by maximum sum of quantum numbers
  - ● - trunctation by maximum quanta for each mode
  - ● - trunctation by maximum energy
  - ● - trunctation by maximum number of simultaneously excited modes

```
void Compute_FC_Factors(Tree &parent, Tree_Options &target_state, int quanta[],
                    int incremented_mode, double current_energy, int global_children_index,
                    Tree* grandnodes[], int num_grandnodes, int generation){

  parent.children = target_state.node_array + global_children_index;

  // If current generation exceeds the user-specified maximum, return;
  if(generation > target_state.max_generation){
    quanta[incremented_mode]--;
    return;
  }

  // Count the number of children that current node (Tree &parent) has.
  int num_possible_children = incremented_mode+1;
  int num_skipped_children = 0;
  for(int i = 0; i < num_possible_children; i++){
    double child_energy = current_energy + target_state.frequency[i];
    if(child_energy > target_state.max_energy){
      num_possible_children = i;
      break;
    }
    if(quanta[i] >= target_state.max_quanta[i]){
      num_skipped_children++;
    }
  }

  int start = 0;
  // If parent has the maximum number of simultaneously excited modes, then the only possible
  // child is created by incrementing the same mode that was just incremented (incremented_mode)
  if(num_grandparents == target_state.max_number_of_simultaneously_excited_modes){
    start = incremented_mode;
  }

  // If no children exist, return
  if((num_possible_children-start) - num_skipped_children <= 0){
    quanta[incremented_mode]--;
    return;
  }

  global_children_index += (num_possible_children-start) - num_skipped_children;
```

```
41
42  int child_index = 0;
43
44  // Loop over children and 1) compute their FC factor and 2) make them the parent of a new
45  // function call
46  for(int i = start; i < num_possible_children; i++){
47
48    // Skip child if child has quanta larger than specified threshold
49    if(quanta[i] >= target_state.max_quanta[i]){
50      continue;
51    }
52
53    double child_energy = current_energy + target_state.frequency[i];
54
55    // Compute the contribution from the parent to the current child's FC overlap
56    int vk = quanta[i];
57    double temp_overlap = [(I - P)δ]_i * (2/(vk+1))^(1/2) * parent.overlap;
58
59    // Compute grandnode terms ( (2P - I)_{kl} * sqrt(v'_l / v'_k + 1) * grandnode->overlap )
60    for(int j = 0; j < num_grandnodes; j++){
61
62      int l = target_state.grandnode_modes[j]; int vl = quanta[l];
63      temp_overlap += (2P - I)_{il} * (vl/(vk+1))^(1/2) * grandnodes[j]->overlap;
64
65    }
66
67    // Set the current child's overlap to the computed value
68    parent.children[child_index].overlap = temp_overlap;
69
70    // *** Update grandnodes for next child ***************************
71    Tree* new_grandnodes[num_grandnodes+1];
72    // If current parent has the maximumn number of simultaneously excited modes
73    if(num_grandnodes == target_state.max_number_of_simultaneously_excited_modes){
74      for(int j = 0; j < num_grandnodes; j++){
75        // If current grandparent ALSO has the maximum number of simultaneously excited modes,
76        // then update to its only possible child (children[0])
77        if(quanta[target_state.grandnode_modes[j]] > 1){
78          new_grandnodes[j] = &(grandnodes[j]->children[0]);
79        }
80        // Otherwise, update to the appropriate child (child at index i)
81        else{
```

```
82            new_grandnodes[j] = &(grandnodes[j]->children[i]);
83          }
84        }
85      }
86      else{
87        // Otherwise, update normally
88        for(int j = 0; j < num_grandnodes; j++){
89          new_grandnodes[j] = &(grandnodes[j]->children[child_index]);
90        }
91      }
92      // ****************************************************************
93
94      // If child is making a new mode non-zero, then spawn a new grandnode
95      if(quanta[i] == 0){
96        new_grandnodes[num_grandnodes] = &parent;
97        target_state.grandnode_modes[num_grandnodes] = i;
98        num_grandnodes++;
99      }
100
101     quanta[i]++;
102     If current intensity is greater than the user-specified threshold,
103     add current overlap to C++ vector of overlaps to be written to output
104     Compute_FC_Overlaps(parent.children[child_index], target_state, quanta, i, child_energy,
105                         global_children_index, new_grandnodes, num_grandnodes, generation+1);
106
107     child_index++;
108
109   }
110
111   quanta[incremeneted_mode]--;
112   return;
113
114 };
```

Below are short descriptions for various sections of the above code:

Lines 13 - 32	Pre-processing stage to determine the number of children that the parent state will have. This part is necessary because of the "self-referencing" approach that we use to form the tree using a linear array of nodes. The number of children that a node will have is dependent on all of the possible approaches for truncating the total number of FC overlap calculations.
Lines 50 - 62	Computation of the FC overlap for a given child of the parent node using the recurrence equations.
Lines 64 - 87	Updating grandnodes in preparation for setting the current child as the next parent node.
Lines 89 - 100	Spawning a new grandnode if necessary, and calling the recursive function with the current child set as the parent node.

Appendix B

The Block-diagonalization (BD) method

The block-diagonalization (BD) method, developed by Cave and Stanton [14], is used to approximately compute quasidiabatic electronic Hamiltonian derivatives using a finite-difference approach:

$$\left(\frac{\partial}{\partial q_j}\left\langle\psi_i^L\right|\hat{H}_e\left|\psi_{i'}^R\right\rangle\right)_{\mathbf{q}=\mathbf{q}_0} \approx \frac{\left\langle\psi_i^L\right|\hat{H}_e\left|\psi_{i'}^R\right\rangle(\mathbf{q}_0+\delta q_j)-\left\langle\psi_i^L\right|\hat{H}_e\left|\psi_{i'}^R\right\rangle(\mathbf{q}_0-\delta q_j)}{2\delta}, \tag{B.1}$$

where

$$\left\langle\psi_i^L\right|\hat{H}_e\left|\psi_{i'}^R\right\rangle(\mathbf{q}_0+\delta q_j)=\left\langle\psi_i^L(\mathbf{r};\mathbf{q}_0+\delta q_j)\right|\hat{H}_e\left|\psi_{i'}^R(\mathbf{r};\mathbf{q}_0+\delta q_j)\right\rangle. \tag{B.2}$$

Remember from Section 2.1.1 that quasidiabatic electronic wave functions are derived from adiabatic electronic wave functions via a unitary rotation operation:

$$\begin{bmatrix}\psi_a^{\mathrm{QD}}(\mathbf{r};\mathbf{q})\\ \psi_b^{\mathrm{QD}}(\mathbf{r};\mathbf{q})\\ \vdots\end{bmatrix}=\begin{bmatrix}M_{aa}(\mathbf{q}) & M_{ab}(\mathbf{q}) & \cdots\\ M_{ba}(\mathbf{q}) & M_{bb}(\mathbf{q}) & \cdots\\ \vdots & \vdots & \ddots\end{bmatrix}\begin{bmatrix}\psi_a(\mathbf{r};\mathbf{q})\\ \psi_b(\mathbf{r};\mathbf{q})\\ \vdots\end{bmatrix} \tag{B.3}$$

where

$$[\mathbf{M}(\mathbf{q})]^\dagger[\mathbf{M}(\mathbf{q})]=[\mathbf{M}(\mathbf{q})][\mathbf{M}(\mathbf{q})]^\dagger=\mathbb{I}. \tag{B.4}$$

Precisely, the BD method is a technique used to determine the rotation matrix $\mathbf{M}(\mathbf{q})$ at small displacements $\delta\mathbf{q}$ from any point $\mathbf{q}_0$ that exhibits enough nuclear symmetry

that the electronic states $\psi_i(\mathbf{r};\mathbf{q}_0)$ belong to different symmetry groups. The approach that the BD method uses to calculate transformation matrices $\mathbf{M}(\mathbf{q}_0+\delta\mathbf{q})$ is built upon the idea that the states $\psi_i(\mathbf{q})$ can not mix at the point $\mathbf{q}_0$ (or any other geometry at which the states have different symmetries) [54, 57, 77, 93, 51, 96]. This means that the quasidiabatic and adiabatic electronic wave functions are equivalent at $\mathbf{q}_0$,

$$\psi_i(\mathbf{r};\mathbf{q}_0) = \psi_i^{\mathrm{QD}}(\mathbf{r};\mathbf{q}_0). \tag{B.5}$$

For spectroscopic problems, the quasidiabatic states of interest are often those preserving the non-interacting characteristic of states at high-symmetry geometries. The BD method then aims to define the quasidiabatic electronic wave functions at stepped geometries, $\psi_i^{\mathrm{QD}}(\mathbf{q}_0+\delta\mathbf{q})$, in such a way that preserves the symmetry distinct nature of the states at the geometry $\mathbf{q}_0$.

It does this by defining the set of quasidiabatic electronic wave functions at stepped geometries, $\psi_i^{\mathrm{QD}}(\mathbf{r};\mathbf{q}_0+\delta\mathbf{q})$, as the orthogonal projection of the set of adiabatic electronic states at the $\mathbf{q}_0$ geometry, $\psi_i(\mathbf{r};\mathbf{q}_0)$ (equivalent to $\psi_i^{\mathrm{QD}}(\mathbf{r};\mathbf{q}_0)$), onto the space of adiabatic electronic wave functions at the stepped geometry, $\psi_i(\mathbf{r};\mathbf{q}_0+\delta\mathbf{q})$. The quasidiabatic electronic states are then written as

$$\psi_i^{\mathrm{QD}}(\mathbf{r};\mathbf{q}_0+\delta\mathbf{q}) = \sum_j \mathbf{M}_{ij}(\mathbf{r};\mathbf{q}_0+\delta\mathbf{q})\psi_j(\mathbf{r};\mathbf{q}_0+\delta\mathbf{q}) \tag{B.6}$$

where

$$\mathbf{M}_{ij} = \langle\psi_i(\mathbf{r};\mathbf{q}_0)|\psi_j(\mathbf{r};\mathbf{q}_0+\delta\mathbf{q})\rangle \tag{B.7}$$

As defined above, the matrix $\mathbf{M}$ does not have the relationship in equation B.4, so a Löwdin orthogonalization technique is used to force the relationship.

The BD method was formulated with the equation-of-motion coupled cluster (EOM-CC) method of computing adiabatic electronic energies in mind. EOM-CC is a non-Hermitian method, so there exist distinct left *and* right wave functions ($\psi^L(\mathbf{r};\mathbf{q})$ and $\psi^R(\mathbf{r};\mathbf{q})$) for each electronic state. This new consideration changes the above equations, but the extension is relatively straightforward.

When left and right wave functions exist, there exist two distinct rotation matrices $\mathbf{M}^L$ and $\mathbf{M}^R$ to transform states $\psi^L(\mathbf{r};\mathbf{q})$ and $\psi^R(\mathbf{r};\mathbf{q})$ into $\psi^{\mathrm{QD};L}(\mathbf{r};\mathbf{q})$ and $\psi^{\mathrm{QD};R}(\mathbf{r};\mathbf{q})$, respectively:

$$\psi_i^{\mathrm{QD};R}(\mathbf{r};\mathbf{q}_0+\delta\mathbf{q}) = \sum_j \mathbf{M}_{ij}^R \psi_j^R(\mathbf{r};\mathbf{q}_0+\delta\mathbf{q}) \tag{B.8}$$

$$\psi_i^{\mathrm{QD};L}(\mathbf{r};\mathbf{q}_0+\delta\mathbf{q}) = \sum_j \mathbf{M}_{ij}^L \psi_j^L(\mathbf{r};\mathbf{q}_0+\delta\mathbf{q}). \tag{B.9}$$

The rotation matrices are defined as

$$\mathbf{M}_{ij}^R = \langle \psi_i^R(\mathbf{r};\mathbf{q}_0) | \psi_j^L(\mathbf{r};\mathbf{q}_0+\delta\mathbf{q}) \rangle \tag{B.10}$$

$$\mathbf{M}_{ij}^L = \langle \psi_i^L(\mathbf{r};\mathbf{q}_0) | \psi_j^R(\mathbf{r};\mathbf{q}_0+\delta\mathbf{q}) \rangle \tag{B.11}$$

and satisfy the relationship

$$\mathbf{M}^L(\mathbf{M}^R)^\dagger = (\mathbf{M}^R)^\dagger \mathbf{M}^L = \mathbf{I}. \tag{B.12}$$

We can use the matrices $\mathbf{M}^L$ and $\mathbf{M}^R$ to compute the values

$$\langle \psi_i^{\mathrm{QD};L} | \, \hat{H}_e \, | \psi_{i'}^{\mathrm{QD};R} \rangle \, (\mathbf{r};\mathbf{q}_0+\delta\mathbf{q}) \tag{B.13}$$

by the following equation:

$$\begin{bmatrix} \langle \psi_a^{\mathrm{QD};L} | \hat{H}_e | \psi_a^{\mathrm{QD};R} \rangle (\mathbf{r}; \mathbf{q}_0 + \delta\mathbf{q}) & \langle \psi_a^{\mathrm{QD};L} | \hat{H}_e | \psi_b^{\mathrm{QD};R} \rangle (\mathbf{r}; \mathbf{q}_0 + \delta\mathbf{q}) & \cdots \\ \langle \psi_b^{\mathrm{QD};L} | \hat{H}_e | \psi_a^{\mathrm{QD};R} \rangle (\mathbf{r}; \mathbf{q}_0 + \delta\mathbf{q}) & \langle \psi_b^{\mathrm{QD};L} | \hat{H}_e | \psi_b^{\mathrm{QD};R} \rangle (\mathbf{r}; \mathbf{q}_0 + \delta\mathbf{q}) & \cdots \\ \vdots & \vdots & \ddots \end{bmatrix} \tag{B.14}$$

$$= \begin{bmatrix} \langle \psi_a^{\mathrm{QD};L}(\mathbf{r}; \mathbf{q}_0 + \delta\mathbf{q}) | \\ \langle \psi_b^{\mathrm{QD};L}(\mathbf{r}; \mathbf{q}_0 + \delta\mathbf{q}) | \\ \vdots \end{bmatrix} \hat{H}_e \begin{bmatrix} | \psi_a^{\mathrm{QD};R}(\mathbf{r}; \mathbf{q}_0 + \delta\mathbf{q}) \rangle \\ | \psi_b^{\mathrm{QD};R}(\mathbf{r}; \mathbf{q}_0 + \delta\mathbf{q}) \rangle \\ \vdots \end{bmatrix}^\dagger \tag{B.15}$$

$$= \mathbf{M}^L \begin{bmatrix} \langle \psi_a^{L}(\mathbf{r}; \mathbf{q}_0 + \delta\mathbf{q}) | \\ \langle \psi_b^{L}(\mathbf{r}; \mathbf{q}_0 + \delta\mathbf{q}) | \\ \vdots \end{bmatrix} \hat{H}_e \begin{bmatrix} | \psi_a^{R}(\mathbf{r}; \mathbf{q}_0 + \delta\mathbf{q}) \rangle \\ | \psi_b^{R}(\mathbf{r}; \mathbf{q}_0 + \delta\mathbf{q}) \rangle \\ \vdots \end{bmatrix}^\dagger (\mathbf{M}^R)^\dagger \tag{B.16}$$

$$= \mathbf{M}^L \begin{bmatrix} V_a(\mathbf{q}_0 + \delta\mathbf{q}) & 0 & \cdots \\ 0 & V_b(\mathbf{q}_0 + \delta\mathbf{q}) & \cdots \\ \vdots & \vdots & \ddots \end{bmatrix} (\mathbf{M}^R)^\dagger, \tag{B.17}$$

since the adiabatic states $\psi_i(\mathbf{r}; \mathbf{q})$ diagonalize the electronic Hamiltonian with eigenvalues $V_i(\mathbf{q})$. So, by equation B.17, once $\mathbf{M}^L$ and $\mathbf{M}^R$ are known, computing

$$\langle \psi_i^{\mathrm{QD};L} | \hat{H}_e | \psi_{i'}^{\mathrm{QD};R} \rangle (\mathbf{r}; \mathbf{q}_0 + \delta\mathbf{q}) \tag{B.18}$$

is straightforward.

B.1 How do we compute the rotation matrices?

To compute the rotation matrices $\mathbf{M}^R$ and $\mathbf{M}^L$, we need to be able to compute the values

$$\mathbf{M}^R_{ij} = \langle \psi_i^R(\mathbf{r}; \mathbf{q}_0) | \psi_j^L(\mathbf{r}; \mathbf{q}_0 + \delta\mathbf{q}) \rangle \tag{B.19}$$

and

$$\mathbf{M}^L_{ij} = \langle \psi_i^L(\mathbf{r}; \mathbf{q}_0) | \psi_j^R(\mathbf{r}; \mathbf{q}_0 + \delta\mathbf{q}) \rangle \,. \tag{B.20}$$

Let us start by looking at the form of the wave functions $\psi(\mathbf{r};\mathbf{q})$. In the equation-of-motion coupled cluster (EOM-CC) approach, the eigenfunctions ψ_i^L and ψ_i^R are of the following form:

$$\psi_i^R(\mathbf{r};\mathbf{q}) = \sum_{k=1}^{n} R_k^{(i)}(\mathbf{q})\Phi_k(\mathbf{r};\mathbf{q}) \tag{B.21}$$

and

$$\psi_i^L(\mathbf{r};\mathbf{q}) = \sum_{k=1}^{n} L_k^{(i)}(\mathbf{q})\Phi_k(\mathbf{r};\mathbf{q}), \tag{B.22}$$

where $\Phi_k(\mathbf{r};\mathbf{q})$ are Slater determinants excited from some reference state $\Phi_0(\mathbf{r};\mathbf{q})$ via single, double, etc. excitations (these will be covered in more detail in the next section). The rotation matrix entries (equations B.19 and B.20) can then be calculated by

$$\mathbf{M}_{ij}^R = \langle\psi_i^R(\mathbf{r};\mathbf{q}_0)|\psi_j^L(\mathbf{r};\mathbf{q}_0+\delta\mathbf{q})\rangle \tag{B.23}$$

$$= \langle\sum_{k=1}^{n} R_k^{(i)}(\mathbf{q}_0)\Phi_k(\mathbf{r};\mathbf{q}_0)|\sum_{l=1}^{n} L_l^{(j)}(\mathbf{q}_0+\delta\mathbf{q})\Phi_l(\mathbf{r};\mathbf{q}_0+\delta\mathbf{q})\rangle\,. \tag{B.24}$$

If we assume that the determinants $\Phi_k(\mathbf{r};\mathbf{q}_0)$ and $\Phi_l(\mathbf{r};\mathbf{q}_0+\delta\mathbf{q})$ are nearly equivalent, then the calculation of $\mathbf{M}_{ij}^R$ simplifies to

$$\mathbf{M}_{ij}^R = \sum_{k=1}^{n} R_k^{(i)}(\mathbf{q}_0)L_k^{(j)}(\mathbf{q}_0+\delta\mathbf{q}), \tag{B.25}$$

which is easily calculable by an inner product of the two eigenvectors containing $R^{(i)}(\mathbf{q}_0)$ and $L^{(j)}(\mathbf{q}_0+\delta\mathbf{q})$. Unfortunately, the two sets of determinants are not always near-equivalent. But, we can *make* the two sets of determinants as similar as possible by rotating the molecular orbitals at the stepped geometry (which make up the determinants $\Phi_l(\mathbf{r};\mathbf{q}_0+\delta\mathbf{q})$) to be as similar as possible to the molecular

orbitals at the reference geometry (which make up the determinants $\Phi_k(\mathbf{r};\mathbf{q}_0)$). To calculate this rotation, we use a modification the BD orbital procedure of Domcke *et al.* [30, 29], which I will now describe.

Let us start by defining $C^{SCF}_{\mathbf{q}_0+\delta\mathbf{q}}$ and $C^{SCF}_{\mathbf{q}_0}$ as the canonical SCF orbitals at the stepped and reference geometries, respectively. At the stepped geometry ($\mathbf{q}_0+\delta\mathbf{q}$), the orbitals calculated at the reference geometry $C^{SCF}_{\mathbf{q}_0}$ are projected onto the space spanned by the orbitals calculated at the stepped geometry $C^{SCF}_{\mathbf{q}_0+\delta\mathbf{q}}$ (occupied and virtuals separately),

$$U^{SCF}_{\mathbf{q}_0+\delta\mathbf{q}} = C^{SCF,T}_{\mathbf{q}_0+\delta\mathbf{q}} S(\mathbf{q}_0+\delta\mathbf{q}) C^{SCF}_{\mathbf{q}_0} \tag{B.26}$$

and

$$C^{SCF,rotated}_{\mathbf{q}_0+\delta\mathbf{q}} = C^{SCF}_{\mathbf{q}_0+\delta\mathbf{q}} U^{SCF}_{\mathbf{q}_0+\delta\mathbf{q}}, \tag{B.27}$$

where $S(\mathbf{q}_0+\delta\mathbf{q})$ is an overlap matrix for the basis functions used to represent the molecular orbitals at the stepped geometry. Much like the CSF expansion, these new, rotated orbitals are not orthonormal, but we can force orthogonality by using a Löwdin orthogonalization technique: Define

$$S_{LO}(\mathbf{q}_0+\delta\mathbf{q}) = C^{SCF,rotated,T}_{\mathbf{q}_0+\delta\mathbf{q}} S(\mathbf{q}_0+\delta\mathbf{q}) C^{SCF,rotated}_{\mathbf{q}_0+\delta\mathbf{q}}. \tag{B.28}$$

Then, the orthogonalized rotated orbitals are calculated by

$$C^{SCF,rotated,LO}_{\mathbf{q}_0+\delta\mathbf{q}} = C^{SCF,rotated}_{\mathbf{q}_0+\delta\mathbf{q}} S^{-0.5}_{LO}(\mathbf{q}_0+\delta\mathbf{q}). \tag{B.29}$$

B.2 Should we rotate orbitals or eigenvectors?

In order to compute the rotation matrices via simple inner product operations, it is important that we compute the EOM adiabatic energies at stepped geometries using the rotated orbitals defined by equation B.29. Performing an EOM calculation with non-canonical orbitals can be complicated, though, and is not always supported by common, available codes. To avoid this, we can instead achieve the same result (EOM eigenvectors in terms of rotated orbitals), by rotating the *eigenvectors determined using canonical SCF orbitals* (which I will call EOM/SCF eigenvectors), rather than the orbitals explicitly.

Before explaining how to rotate the EOM/SCF eigenvectors for the BD method, I need to introduce EOM-CC determinants and creation/annihilation operators. EOM-CC eigenfunctions take the form

$$\psi(\mathbf{r};\mathbf{q}) = \underbrace{\sum_{i=1}^{n_{occ}}\sum_{a=1}^{n_{virt}} R_i^a(\mathbf{q})\Phi_i^a(\mathbf{r};\mathbf{q})}_{\text{single excitations}} + \underbrace{\sum_{i,j=1}^{n_{occ}}\sum_{a,b=1}^{n_{virt}} R_{ij}^{ab}(\mathbf{q})\Phi_{ij}^{ab}(\mathbf{r};\mathbf{q})}_{\text{double excitations}} + \ldots, \tag{B.30}$$

where each determinant $\Phi_{ij\ldots}^{ab\cdots}(\mathbf{r};\mathbf{q})$ is a Slater determinant in which the occupied orbitals $\{\phi_i, \phi_j, \ldots\}$ have been replaced by virtual orbitals $\{\phi_a, \phi_b, \ldots\}$.

Each excited determinant can be written in terms of creation/annihilation operators: Let $|\phi_i\phi_j\ldots\rangle$ be a Slater determinant comprised of the orbitals $\{\phi_i, \phi_j, \ldots\}$, then we can define the creation operator $a_i^\dagger$ with the property

$$|\phi_i\rangle = a_i^\dagger |0\rangle \tag{B.31}$$

where $|0\rangle$ is a vacuum state. Annihilation operators a_i are defined as having an

opposite effect:

$$|0\rangle = a_i \, |\phi_i\rangle \, . \tag{B.32}$$

Each EOM-CC determinant can then be written in terms of creation/annihilation operators acting on a reference determinant Φ_0 made up entirely of occupied orbitals:

$$\Phi_{ij\ldots}^{ab\ldots}(\mathbf{r};\mathbf{q}) = a_a^\dagger a_b^\dagger \ldots a_i a_j \ldots \Phi_0. \tag{B.33}$$

The above creation/annihilation operators work on a certain basis of orbitals $\{\phi_i\}$. We can carry out the same procedure using a different basis of orbitals $\{\phi_i'\}$. Let us call these new operators as $a_{i'}^\dagger$ and $a_{i'}$. If we assume the two bases are related via a unitary transformation, then we get

$$|\phi_i'\rangle = \sum_j U_{ji} \, |\phi_j\rangle \, . \tag{B.34}$$

So,

$$a_{i'}^\dagger \, |0\rangle = \sum_j U_{ji} a_j^\dagger \, |0\rangle \, . \tag{B.35}$$

This gives

$$a_{i'}^\dagger = \sum_j U_{ji} a_j^\dagger \quad \Rightarrow \quad a_i^\dagger = \sum_j U_{ij} a_{j'}^\dagger. \tag{B.36}$$

The conjugate of the above equations gives us

$$a_{i'} = \sum_j U_{ji} a_j \quad \Rightarrow \quad a_i = \sum_j U_{ij} a_{j'}, \tag{B.37}$$

since **U** is a real matrix.

The following theorem will be referenced in all future future theorems in this section and must be introduced before moving forward.

Theorem B.2.1. *Let* $\boldsymbol{T}$ *be an* $n_{occ} \times n_{occ}$ *rotation matrix that rotates a set of* n_{occ} *molecular orbitals* $\{\phi_i\}$ *to create a new set of molecular orbitals* $\{\phi'_i\}$*:*

$$\begin{aligned}\begin{bmatrix}\phi'_1 & \phi'_2 & \dots & \phi'_{n_{occ}}\end{bmatrix} \\ = \begin{bmatrix}\phi_1 & \phi_2 & \dots & \phi_{n_{occ}}\end{bmatrix} \begin{bmatrix} T_{11} & T_{12} & \dots & T_{1n_{occ}} \\ T_{21} & T_{22} & \dots & T_{2n_{occ}} \\ \vdots & \vdots & \ddots & \vdots \\ T_{n_{occ}1} & T_{n_{occ}2} & \dots & T_{n_{occ}n_{occ}}\end{bmatrix}\end{aligned} \tag{B.38}$$

and let Φ_0 *be a reference determinant comprised entirely of the occupied orbitals* $\{\phi_i\}$*. If* $\Phi_{0'}$ *is a reference determinant comprised entirely of the occupied orbitals* $\{\phi'_i\}$*, then*

$$\Phi_{0'} = \Phi_0 \tag{B.39}$$

Proof.

$$\begin{aligned}\Phi_{0'} &= \begin{vmatrix}\begin{bmatrix} \phi'_1(r_1) & \phi'_2(r_1) & \cdots & \phi'_{n_{occ}}(r_1) \\ \phi'_1(r_2) & \phi'_2(r_2) & \cdots & \phi'_{n_{occ}}(r_2) \\ \vdots & \vdots & \ddots & \vdots \\ \phi'_1(r_{n_{occ}}) & \phi'_2(r_{n_{occ}}) & \cdots & \phi'_{n_{occ}}(r_{n_{occ}})\end{bmatrix}\end{vmatrix} \\ &= \begin{vmatrix}\begin{bmatrix} \phi_1(r_1) & \phi_2(r_1) & \cdots & \phi_{n_{occ}}(r_1) \\ \phi_1(r_2) & \phi_2(r_2) & \cdots & \phi_{n_{occ}}(r_2) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_1(r_{n_{occ}}) & \phi_2(r_{n_{occ}}) & \cdots & \phi_{n_{occ}}(r_{n_{occ}})\end{bmatrix} \begin{bmatrix} T_{11} & T_{12} & \dots & T_{1n_{occ}} \\ T_{21} & T_{22} & \dots & T_{2n_{occ}} \\ \vdots & \vdots & \ddots & \vdots \\ T_{n_{occ}1} & T_{n_{occ}2} & \dots & T_{n_{occ}n_{occ}}\end{bmatrix}\end{vmatrix} \\ &= \begin{vmatrix}\begin{bmatrix} \phi_1(r_1) & \phi_2(r_1) & \cdots & \phi_{n_{occ}}(r_1) \\ \phi_1(r_2) & \phi_2(r_2) & \cdots & \phi_{n_{occ}}(r_2) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_1(r_{n_{occ}}) & \phi_2(r_{n_{occ}}) & \cdots & \phi_{n_{occ}}(r_{n_{occ}})\end{bmatrix}\end{vmatrix} * |\mathbf{T}| \\ &= \Phi_0 * 1 = \Phi_0\end{aligned} \tag{B.40}$$

□

With the above information, we can determine relationships between EOM-CC eigenvectors with determinants comprised of SCF orbitals and EOM-CC eigenvectors with determinants comprised of rotated orbitals. For simplicity, let us consider first only the single-excitation portion of the EOM-CC eigenvectors (labeled "single excitations" in equation B.30).

Theorem B.2.2. *Let* $\boldsymbol{T}$ *be an* $n_{occ} \times n_{occ}$ *rotation matrix that rotates a set of* n_{occ} *molecular orbitals* $\{\phi_i\}$ *to create a new set of molecular orbitals* $\{\phi'_i\}$*:*

$$\begin{aligned}\begin{bmatrix}\phi'_1 & \phi'_2 & \dots & \phi'_{n_{occ}}\end{bmatrix} \\ = \begin{bmatrix}\phi_1 & \phi_2 & \dots & \phi_{n_{occ}}\end{bmatrix}\begin{bmatrix} T_{11} & T_{12} & \dots & T_{1n_{occ}} \\ T_{21} & T_{22} & \dots & T_{2n_{occ}} \\ \vdots & \vdots & \ddots & \vdots \\ T_{n_{occ}1} & T_{n_{occ}2} & \dots & T_{n_{occ}n_{occ}}\end{bmatrix}\end{aligned} \tag{B.41}$$

and let the matrix R *contain the coefficients for the single-excitation portion of an EOM-CC wave function whose determinants* Φ_i^a *use* $\{\phi_i\}$ *as the set of occupied orbitals,*

$$\sum_{i=1}^{n_{occ}}\sum_{a=1}^{n_{virt}} R_i^a(\mathbf{q})\Phi_i^a(\mathbf{r};\mathbf{q}). \tag{B.42}$$

If the EOM-CC wave function determinants instead used the set $\{\phi'_i\}$ *as occupied orbitals, i.e.*

$$\sum_{i'=1}^{n_{occ}}\sum_{a=1}^{n_{virt}} R_{i'}^a(\mathbf{q})\Phi_{i'}^a(\mathbf{r};\mathbf{q}). \tag{B.43}$$

the corresponding coefficient matrix

$$\boldsymbol{R'} = \begin{bmatrix} R_{1'}^1 & R_{2'}^1 & \dots & R_{n'_{occ}}^1 \\ R_{1'}^2 & R_{2'}^2 & \dots & R_{n'_{occ}}^2 \\ \vdots & \vdots & \ddots & \vdots \\ R_{1'}^{n_{virt}} & R_{2'}^{n_{virt}} & \dots & R_{n'_{occ}}^{n_{virt}}\end{bmatrix} \tag{B.44}$$

can be determined through the transformation

$$\boldsymbol{R}' = \boldsymbol{R} * \boldsymbol{T} \tag{B.45}$$

Proof. To begin, let us equate the two EOM-CC wave functions that use different, but related, sets of molecular orbitals:

$$\sum_{i'=1}^{n_{occ}} \sum_{a=1}^{n_{virt}} R_{i'}^{a}(\mathbf{q}) \Phi_{i'}^{a}(\mathbf{r};\mathbf{q}) = \sum_{i=1}^{n_{occ}} \sum_{a=1}^{n_{virt}} R_{i}^{a}(\mathbf{q}) \Phi_{i}^{a}(\mathbf{r};\mathbf{q}). \tag{B.46}$$

or

$$\sum_{i'=1}^{n_{occ}} \sum_{a=1}^{n_{virt}} R_{i'}^{a}(\mathbf{q}) a_a^\dagger a_{i'} \Phi_{0'}(\mathbf{r};\mathbf{q}) = \sum_{i=1}^{n_{occ}} \sum_{a=1}^{n_{virt}} R_{i}^{a}(\mathbf{q}) a_a^\dagger a_i \Phi_{0}(\mathbf{r};\mathbf{q}). \tag{B.47}$$

We know, from equation B.37, that $a_i = \sum_{k=1}^{n_{occ}} T_{ik} a_{k'}$, and, from Theorem B.2.1, that $\Phi_0(\mathbf{r};\mathbf{q}) = \Phi_{0'}(\mathbf{r};\mathbf{q})$, so we can replace equation B.47 with

$$\sum_{i'=1}^{n_{occ}} \sum_{a=1}^{n_{virt}} R_{i'}^{a}(\mathbf{q}) a_a^\dagger a_{i'} \Phi_{0'}(\mathbf{r};\mathbf{q}) \tag{B.48}$$

$$= \sum_{i=1}^{n_{occ}} \sum_{a=1}^{n_{virt}} R_{i}^{a}(\mathbf{q}) a_a^\dagger \left[\sum_{k=1}^{n_{occ}} T_{ik} a_{k'} \right] \Phi_{0'}(\mathbf{r};\mathbf{q}). \tag{B.49}$$

We can get an expression for a specific $R_{\gamma'}^{\alpha}$ by taking the inner product of $a_\alpha^\dagger a_{\gamma'} \Phi_{0'}(\mathbf{r};\mathbf{q})$ with both sides:

$$\sum_{i'=1}^{n_{occ}} \sum_{a=1}^{n_{virt}} R_{i'}^{a}(\mathbf{q}) \left\langle a_\alpha^\dagger a_{\gamma'} \Phi_{0'}(\mathbf{r};\mathbf{q}) | a_a^\dagger a_{i'} \Phi_{0'}(\mathbf{r};\mathbf{q}) \right\rangle \tag{B.50}$$

$$= \sum_{i=1}^{n_{occ}} \sum_{a=1}^{n_{virt}} R_{i}^{a}(\mathbf{q}) \left\langle a_\alpha^\dagger a_{\gamma'} \Phi_{0'}(\mathbf{r};\mathbf{q}) | a_a^\dagger \left[\sum_{k=1}^{n_{occ}} T_{ik} a_{k'} \right] \Phi_{0'}(\mathbf{r};\mathbf{q}) \right\rangle. \tag{B.51}$$

and since the functions $a^\dagger a_{i'} \Phi_{0'}(\mathbf{r};\mathbf{q})$ are orthogonal to each other, we get

$$R_{\gamma'}^{\alpha}(\mathbf{q}) = \sum_{i=1}^{n_{occ}} T_{i\gamma} R_{i}^{\alpha}(\mathbf{q}). \tag{B.52}$$

In matrix form, this is

$$\mathbf{R}' = \mathbf{R} * \mathbf{T} \tag{B.53}$$

□

Theorem B.2.3. *Let* $\boldsymbol{T}$ *be an* $n_{virt} \times n_{virt}$ *rotation matrix that rotates a set of* n_{virt} *molecular orbitals* $\{\phi_a\}$ *to create a new set of molecular orbitals* $\{\phi'_a\}$*:*

$$\begin{aligned}\begin{bmatrix}\phi'_1 & \phi'_2 & \dots & \phi'_{n_{virt}}\end{bmatrix} \\ = \begin{bmatrix}\phi_1 & \phi_2 & \dots & \phi_{n_{virt}}\end{bmatrix} \begin{bmatrix} T_{11} & T_{12} & \dots & T_{1n_{virt}} \\ T_{21} & T_{22} & \dots & T_{2n_{virt}} \\ \vdots & \vdots & \ddots & \vdots \\ T_{n_{virt}1} & T_{n_{virt}2} & \dots & T_{n_{virt}n_{virt}}\end{bmatrix}\end{aligned} \tag{B.54}$$

and let the matrix $\boldsymbol{R}$ *contain the coefficients for the single-excitation portion of an EOM-CC wave function whose determinants* Φ_i^a *use* $\{\phi_a\}$ *as the set of virtual orbitals,*

$$\sum_{i=1}^{n_{occ}} \sum_{a=1}^{n_{virt}} R_i^a(\mathbf{q}) \Phi_i^a(\mathbf{r}; \mathbf{q}). \tag{B.55}$$

If the EOM-CC wave function determinants instead used the set $\{\phi'_a\}$ *as virtual orbitals, i.e.*

$$\sum_{i=1}^{n_{occ}} \sum_{a'=1}^{n_{virt}} R_i^{a'}(\mathbf{q}) \Phi_i^{a'}(\mathbf{r}; \mathbf{q}). \tag{B.56}$$

the corresponding coefficient matrix

$$\boldsymbol{R}' = \begin{bmatrix} R_1^{1'} & R_2^{1'} & \dots & R_{n_{occ}}^{1'} \\ R_1^{2'} & R_2^{2'} & \dots & R_{n_{occ}}^{2'} \\ \vdots & \vdots & \ddots & \vdots \\ R_1^{n'_{virt}} & R_2^{n'_{virt}} & \dots & R_{n_{occ}}^{n'_{virt}}\end{bmatrix} \tag{B.57}$$

can be determined through the transformation

$$\boldsymbol{R}' = \boldsymbol{T} * \boldsymbol{R} \tag{B.58}$$

Proof. The proof follows in the same vein as that of Theorem B.2.2. Note that, from equation B.36, $a_a^\dagger = \sum_{c=1}^{n_{virt}} T_{ca} a_{a'}^\dagger$. □

Corollary B.2.4. *Let $\boldsymbol{T}_1$ and $\boldsymbol{T}_2$ be $n_{occ} \times n_{occ}$ and $n_{virt} \times n_{virt}$ rotation matrices, respectively, that rotate sets of orbitals according to equations B.41 and B.54, respectively. Let the matrix R contain the coefficients for the single-excitation portion of an EOM-CC wave function whose determinants Φ_i^a use $\{\phi_i\}$ as the set of occupied orbitals and $\{\phi_a\}$ as the set of virtual orbitals,*

$$\sum_{i=1}^{n_{occ}} \sum_{a=1}^{n_{virt}} R_i^a(\mathbf{q}) \Phi_i^a(\mathbf{r}; \mathbf{q}). \tag{B.59}$$

If the EOM-CC wave function determinants instead used the set $\{\phi_i'\}$ as occupied orbitals and $\{\phi_a'\}$ as virtual orbitals, i.e.

$$\sum_{i'=1}^{n_{occ}} \sum_{a'=1}^{n_{virt}} R_{i'}^{a'}(\mathbf{q}) \Phi_{i'}^{a'}(\mathbf{r}; \mathbf{q}). \tag{B.60}$$

the corresponding coefficient matrix

$$\boldsymbol{R}' = \begin{bmatrix} R_{1'}^{1'} & R_{2'}^{1'} & \dots & R_{n'_{occ}}^{1'} \\ R_{1'}^{2'} & R_{2'}^{2'} & \dots & R_{n'_{occ}}^{2'} \\ \vdots & \vdots & \ddots & \vdots \\ R_{1'}^{n'_{virt}} & R_{2'}^{n'_{virt}} & \dots & R_{n'_{occ}}^{n'_{virt}} \end{bmatrix} \tag{B.61}$$

can be determined through the transformation

$$\boldsymbol{R}' = \boldsymbol{T}_2 * \boldsymbol{R} * \boldsymbol{T}_1 \tag{B.62}$$

The theorems used to calculate the double-excitation portion of the EOM-CC eigenvectors are similar.

Theorem B.2.5. *Let* $\boldsymbol{T}$ *be an* $n_{occ} \times n_{occ}$ *rotation matrix that rotates a set of* n_{occ} *molecular orbitals* $\{\phi_i\}$ *to create a new set of molecular orbitals* $\{\phi'_i\}$*:*

$$\begin{aligned}\begin{bmatrix}\phi'_1 & \phi'_2 & \dots & \phi'_{n_{occ}}\end{bmatrix} \\ = \begin{bmatrix}\phi_1 & \phi_2 & \dots & \phi_{n_{occ}}\end{bmatrix} \begin{bmatrix} T_{11} & T_{12} & \dots & T_{1n_{occ}} \\ T_{21} & T_{22} & \dots & T_{2n_{occ}} \\ \vdots & \vdots & \ddots & \vdots \\ T_{n_{occ}1} & T_{n_{occ}2} & \dots & T_{n_{occ}n_{occ}} \end{bmatrix}\end{aligned} \tag{B.63}$$

and let the matrix R *contain the coefficients for the double-excitation portion of an EOM-CC wave function whose determinants* Φ_{ij}^{ab} *use* $\{\phi_i\}$ *as the set of occupied orbitals,*

$$\sum_{i,j=1}^{n_{occ}} \sum_{a,b=1}^{n_{virt}} R_{ij}^{ab}(\mathbf{q}) \Phi_{ij}^{ab}(\mathbf{r};\mathbf{q}). \tag{B.64}$$

If the EOM-CC wave function determinants instead used the set $\{\phi'_i\}$ *as occupied orbitals, i.e.*

$$\sum_{i',j'=1}^{n_{occ}} \sum_{a,b=1}^{n_{virt}} R_{i'j'}^{ab}(\mathbf{q}) \Phi_{i'j'}^{ab}(\mathbf{r};\mathbf{q}). \tag{B.65}$$

the corresponding coefficient matrix (for ease of notation, let $m = n_{virt}$ *and* $n = n_{occ}$*)*

$$\boldsymbol{R'} = \left[\begin{array}{ccc|ccc|c|ccc} R_{1'1'}^{11} & \dots & R_{n'1'}^{11} & R_{1'2'}^{11} & \dots & R_{n'2'}^{11} & \cdots & R_{1'n'}^{11} & \dots & R_{n'n'}^{11} \\ R_{1'1'}^{21} & \dots & R_{n'1'}^{21} & R_{1'2'}^{21} & \dots & R_{n'2'}^{21} & \cdots & R_{1'n'}^{21} & \dots & R_{n'n'}^{21} \\ \vdots & & \vdots & \vdots & & \vdots & & \vdots & & \vdots \\ R_{1'1'}^{mm} & \dots & R_{n'1'}^{mm} & R_{1'2'}^{mm} & \dots & R_{n'2'}^{mm} & \cdots & R_{1'n'}^{mm} & \dots & R_{n'n'}^{mm} \end{array}\right] \tag{B.66}$$

can be determined through the transformation

$$\boldsymbol{R'} = \boldsymbol{R} * (\boldsymbol{T} \otimes \boldsymbol{T}) \tag{B.67}$$

Proof. To begin, let us equate the two EOM-CC wave functions that use different, but related, sets of molecular orbitals:

$$\sum_{i',j'=1}^{n_{occ}} \sum_{a,b=1}^{n_{virt}} R^{ab}_{i'j'}(\mathbf{q}) \Phi^{ab}_{i'j'}(\mathbf{r};\mathbf{q}) = \sum_{i,j=1}^{n_{occ}} \sum_{a,b=1}^{n_{virt}} R^{ab}_{ij}(\mathbf{q}) \Phi^{ab}_{ij}(\mathbf{r};\mathbf{q}). \tag{B.68}$$

or

$$\sum_{i',j'=1}^{n_{occ}} \sum_{a,b=1}^{n_{virt}} R^{ab}_{i'j'}(\mathbf{q}) a^\dagger_b a'_j a^\dagger_a a'_i \Phi_{0'}(\mathbf{r};\mathbf{q}) = \sum_{i,j=1}^{n_{occ}} \sum_{a,b=1}^{n_{virt}} R^{ab}_{ij}(\mathbf{q}) a^\dagger_b a_j a^\dagger_a a_i \Phi_0(\mathbf{r};\mathbf{q}). \tag{B.69}$$

We know, from equation B.37, that $a_i = \sum_{k=1}^{n_{occ}} T_{ik} a'_k$, so we can replace equation B.69 with

$$\sum_{i',j'=1}^{n_{occ}} \sum_{a,b=1}^{n_{virt}} R^{ab}_{i'j'}(\mathbf{q}) a^\dagger_b a_{j'} a^\dagger_a a_{i'} \Phi_{0'}(\mathbf{r};\mathbf{q}) \tag{B.70}$$

$$= \sum_{i,j=1}^{n_{occ}} \sum_{a,b=1}^{n_{virt}} R^{ab}_{ij}(\mathbf{q}) a^\dagger_b \left[\sum_{l=1}^{n_{occ}} T_{jl} a'_l\right] a^\dagger_a \left[\sum_{k=1}^{n_{occ}} T_{ik} a'_k\right] \Phi_{0'}(\mathbf{r};\mathbf{q}). \tag{B.71}$$

We can get an expression for a specific $R^{\alpha\beta}_{\gamma'\delta'}$ by taking the inner product of $a^\dagger_\beta a_{\delta'} a^\dagger_\alpha a_{\gamma'} \Phi_{0'}(\mathbf{r};\mathbf{q})$ with both sides:

$$\sum_{i',j'=1}^{n_{occ}} \sum_{a,b=1}^{n_{virt}} R^{ab}_{i'j'}(\mathbf{q}) \left\langle a^\dagger_\beta a_{\delta'} a^\dagger_\alpha a_{\gamma'} \Phi_{0'}(\mathbf{r};\mathbf{q}) | a^\dagger_b a_{j'} a^\dagger_a a_{i'} \Phi_{0'}(\mathbf{r};\mathbf{q}) \right\rangle \tag{B.72}$$

$$= \sum_{i,j=1}^{n_{occ}} \sum_{a,b=1}^{n_{virt}} R^{ab}_{ij}(\mathbf{q}) \left\langle a^\dagger_\beta a_{\delta'} a^\dagger_\alpha a_{\gamma'} \Phi_{0'}(\mathbf{r};\mathbf{q}) | a^\dagger_b \left[\sum_{l=1}^{n_{occ}} T_{jl} a_{l'}\right] a^\dagger_a \left[\sum_{k=1}^{n_{occ}} T_{ik} a_{k'}\right] \Phi_{0'}(\mathbf{r};\mathbf{q}) \right\rangle . \tag{B.73}$$

and since the functions $a^\dagger_b a_j a^\dagger_a a_i \Phi_{0'}(\mathbf{r};\mathbf{q})$ are orthogonal to each other, we get

$$R^{\alpha\beta}_{\gamma'\delta'}(\mathbf{q}) = \sum_{i,j=1}^{n_{occ}} T_{j\delta} T_{i\gamma} R^{\alpha\beta}_{ij}(\mathbf{q}). \tag{B.74}$$

In matrix form, this is

$$\mathbf{R} = \mathbf{R} * (\mathbf{T} \otimes \mathbf{T}) \tag{B.75}$$

□

Theorem B.2.6. *Let **T** be an $n_{virt} \times n_{virt}$ rotation matrix that rotates a set of n_{virt} molecular orbitals $\{\phi_a\}$ to create a new set of molecular orbitals $\{\phi'_a\}$:*

$$\begin{aligned}\begin{bmatrix}\phi'_1 & \phi'_2 & \dots & \phi'_{n_{virt}}\end{bmatrix} \\ = \begin{bmatrix}\phi_1 & \phi_2 & \dots & \phi_{n_{virt}}\end{bmatrix} \begin{bmatrix} T_{11} & T_{12} & \dots & T_{1n_{virt}} \\ T_{21} & T_{22} & \dots & T_{2n_{virt}} \\ \vdots & \vdots & \ddots & \vdots \\ T_{n_{virt}1} & T_{n_{virt}2} & \dots & T_{n_{virt}n_{virt}} \end{bmatrix}\end{aligned} \tag{B.76}$$

and let the matrix R contain the coefficients for the double-excitation portion of an EOM-CC wave function whose determinants Φ_{ij}^{ab} use $\{\phi_a\}$ as the set of occupied orbitals,

$$\sum_{i,j=1}^{n_{occ}} \sum_{a,b=1}^{n_{virt}} R_{ij}^{ab}(\mathbf{q}) \Phi_{ij}^{ab}(\mathbf{r};\mathbf{q}). \tag{B.77}$$

If the EOM-CC wave function determinants instead used the set $\{\phi'_a\}$ as occupied orbitals, i.e.

$$\sum_{i,j=1}^{n_{occ}} \sum_{a',b'=1}^{n_{virt}} R_{ij}^{a'b'}(\mathbf{q}) \Phi_{ij}^{a'b'}(\mathbf{r};\mathbf{q}). \tag{B.78}$$

the corresponding coefficient matrix (for ease of notation, let $m = n_{virt}$ and $n = n_{occ}$)

$$\boldsymbol{R'} = \left[\begin{array}{ccc|ccc|c|ccc} R_{11}^{1'1'} & \dots & R_{n1}^{1'1'} & R_{12}^{1'1'} & \dots & R_{n2}^{1'1'} & \cdots & R_{1n}^{1'1'} & \dots & R_{nn}^{1'1'} \\ R_{11}^{2'1'} & \dots & R_{n1}^{2'1'} & R_{12}^{2'1'} & \dots & R_{n2}^{2'1'} & \cdots & R_{1n}^{2'1'} & \dots & R_{nn}^{2'1'} \\ \vdots & & \vdots & \vdots & & \vdots & & \vdots & & \vdots \\ R_{11}^{m'm'} & \dots & R_{n1}^{m'm'} & R_{12}^{m'm'} & \dots & R_{n2}^{m'm'} & \cdots & R_{1n}^{m'm'} & \dots & R_{nn}^{m'm'} \end{array}\right] \tag{B.79}$$

can be determined through the transformation

$$\boldsymbol{R}' = (\boldsymbol{T} \otimes \boldsymbol{T}) * \boldsymbol{R} \tag{B.80}$$

Proof. The proof follows in the same vein as that of Theorem B.2.5. Note that, from equation B.36, $a_a^\dagger = \sum_{c=1}^{n_{virt}} T_{ca} a_{a'}^\dagger$. □

Corollary B.2.7. *Let $\boldsymbol{T}_1$ and $\boldsymbol{T}_2$ be $n_{occ} \times n_{occ}$ and $n_{virt} \times n_{virt}$ rotation matrices, respectively, that rotate sets of orbitals according to equations B.41 and B.54, respectively. Let the matrix R contain the coefficients for the double-excitation portion of an EOM-CC wave function whose determinants Φ_{ij}^{ab} use $\{\phi_i\}$ as the set of occupied orbitals and $\{\phi_a\}$ as the set of virtual orbitals,*

$$\sum_{i,j=1}^{n_{occ}} \sum_{a,b=1}^{n_{virt}} R_{ij}^{ab}(\mathbf{q}) \Phi_{ij}^{ab}(\mathbf{r};\mathbf{q}). \tag{B.81}$$

If the EOM-CC wave function determinants instead used the set $\{\phi_i'\}$ as occupied orbitals and $\{\phi_a'\}$ as virtual orbitals, i.e.

$$\sum_{i',j'=1}^{n_{occ}} \sum_{a',b'=1}^{n_{virt}} R_{i'j'}^{a'b'}(\mathbf{q}) \Phi_{i'j'}^{a'b'}(\mathbf{r};\mathbf{q}). \tag{B.82}$$

the corresponding coefficient matrix (for ease of notation, let $m = n_{virt}$ and $n = n_{occ}$)

$$\boldsymbol{R}' = \left[\begin{array}{ccc|ccc|c|ccc} R_{1'1'}^{1'1'} & \dots & R_{n'1'}^{1'1'} & R_{1'2'}^{1'1'} & \dots & R_{n'2'}^{1'1'} & \cdots & R_{1'n'}^{1'1'} & \dots & R_{n'n'}^{1'1'} \\ R_{1'1'}^{2'1'} & \dots & R_{n'1'}^{2'1'} & R_{1'2'}^{2'1'} & \dots & R_{n'2'}^{2'1'} & \cdots & R_{1'n'}^{2'1'} & \dots & R_{n'n'}^{2'1'} \\ \vdots & & \vdots & \vdots & & \vdots & & \vdots & & \vdots \\ R_{1'1'}^{m'm'} & \dots & R_{n'1'}^{m'm'} & R_{1'2'}^{m'm'} & \dots & R_{n'2'}^{m'm'} & \cdots & R_{1'n'}^{m'm'} & \dots & R_{n'n'}^{m'm'} \end{array}\right] \tag{B.83}$$

can be determined through the transformation

$$\boldsymbol{R}' = (\boldsymbol{T}_2 \otimes \boldsymbol{T}_2) * \boldsymbol{R} * (\boldsymbol{T}_1 \otimes \boldsymbol{T}_1) \tag{B.84}$$

Bibliography

[1] Murat Aydin, Kristal R Verhulst, Eric S Saltzman, Mark O Battle, Stephen A Montzka, Donald R Blake, Qi Tang, and Michael J Prather. Recent decreases in fossil-fuel emissions of ethane and methane derived from firn air. *Nature*, 476(7359):198–201, 2011.

[2] Zhaojun Bai, James Demmel, Jack Dongarra, Axel Ruhe, and Henk van der Vorst. *Templates for the solution of algebraic eigenvalue problems: a practical guide*, volume 11. Siam, 2000.

[3] Satish Balay, Shrirang Abhyankar, Mark F. Adams, Jed Brown, Peter Brune, Kris Buschelman, Lisandro Dalcin, Victor Eijkhout, William D. Gropp, Dinesh Kaushik, Matthew G. Knepley, Lois Curfman McInnes, Karl Rupp, Barry F. Smith, Stefano Zampini, Hong Zhang, and Hong Zhang. PETSc Web page. http://www.mcs.anl.gov/petsc, 2016.

[4] Satish Balay, Shrirang Abhyankar, Mark F. Adams, Jed Brown, Peter Brune, Kris Buschelman, Lisandro Dalcin, Victor Eijkhout, William D. Gropp, Dinesh Kaushik, Matthew G. Knepley, Lois Curfman McInnes, Karl Rupp, Barry F. Smith, Stefano Zampini, Hong Zhang, and Hong Zhang. PETSc users manual. Technical Report ANL-95/11 - Revision 3.7, Argonne National Laboratory, 2016.

[5] Satish Balay, William D. Gropp, Lois Curfman McInnes, and Barry F. Smith. Efficient management of parallelism in object oriented numerical software libraries. In E. Arge, A. M. Bruaset, and H. P. Langtangen, editors, *Modern Software Tools in Scientific Computing*, pages 163–202. Birkhäuser Press, 1997.

[6] R Berger, C Fischer, and M Klessinger. Calculation of the vibronic fine structure in electronic spectra at higher temperatures. 1. benzene and pyrazine. *J. Phys. Chem. A*, 102(36):7157–7167, 1998.

[7] Camille Bilger, Paul Rimmer, and Ch Helling. Small hydrocarbon molecules in cloud-forming brown dwarf and giant gas planet atmospheres. *Monthly Notices of the Royal Astronomical Society*, page stt1378, 2013.

[8] Ch W Bock, Yu N Panchenko, SV Krasnoshchiokov, and VI Pupyshev. Structure and vibrational assignment of gouche-1, 3-butadiene. *Journal of molecular structure*, 129(1-2):57–67, 1985.

[9] Yannick J Bomble, Kurt W Sattelmeyer, John F Stanton, and Jürgen Gauss. On the vertical excitation energy of cyclopentadiene. *The Journal of chemical physics*, 121(11):5236–5240, 2004.

[10] ME Brown, KM Barkume, GA Blake, EL Schaller, DL Rabinowitz, HG Roe, and CA Trujillo. Methane and ethane on the bright kuiper belt object 2005 fy9. *The Astronomical Journal*, 133(1):284, 2006.

[11] James W Caldwell and Mark S Gordon. An approach to polyatomic franck-condon integrals: Application to the photoelectron spectrum of water. *J. Mol. Spectrosc.*, 84(2):503–519, 1980.

[12] W Caminati, G Grassi, and A Bauder. Microwave fourier transform spectrum of s-trans-1, 3-butadiene-1, 1-d 2. *Chemical physics letters*, 148(1):13–16, 1988.

[13] Robert J Cave and John F Stanton. Block diagonalization of the equation-of-motion coupled cluster effective hamiltonian: Treatment of diabatic potential constants and triple excitations. *The Journal of chemical physics*, 140(21):214112, 2014.

[14] Robert J. Cave and John F. Stanton. Block diagonalization of the equation-of-motion coupled cluster effective hamiltonian: Treatment of diabatic potential constants and triple excitations. *The Journal of Chemical Physics*, 140(21):–, 2014.

[15] LS Cederbaum, H Köppel, and W Domcke. Multimode vibronic coupling effects in molecules. *International Journal of Quantum Chemistry*, 20(S15):251–267, 1981.

[16] Richard R Chadwick, Daniel P Gerrity, and Bruce S Hudson. Resonance raman spectroscopy of butadiene: demonstration of a 2 1 a g state below the 1 1 b u v state. *Chemical physics letters*, 115(1):24–28, 1985.

[17] Richard R Chadwick, Marek Z Zgierski, and Bruce S Hudson. Resonance raman scattering of butadiene: Vibronic activity of a bu mode demonstrates the presence of a 1ag symmetry excited electronic state at low energy. *The Journal of chemical physics*, 95(10):7204–7211, 1991.

[18] Norman C Craig, Peter Groner, and Donald C McKean. Equilibrium structures for butadiene and ethylene: compelling evidence for π-electron delocalization in butadiene. *The Journal of Physical Chemistry A*, 110(23):7461–7469, 2006.

[19] Csaba Daday, Simon Smart, George H Booth, Ali Alavi, and Claudia Filippi. Full configuration interaction excitations of ethene and butadiene: Resolution of an ancient question. *Journal of chemical theory and computation*, 8(11):4441–4451, 2012.

[20] Germund Dahlquist, Åke Björk, and Ned Anderson. *Numerical methods*. Prentice-Hall, 1974.

[21] Michal Dallos and Hans Lischka. A systematic theoretical investigation of the lowest valence-and rydberg-excited singlet states of trans-butadiene. the character of the 11bu (v) state revisited. *Theoretical Chemistry Accounts*, 112(1):16–26, 2004.

[22] M Dierksen. Fcfast (ver. 1.0). *Universität Münster, Münster, Germany*, 2005.

[23] Marc Dierksen and Stefan Grimme. An efficient approach for the calculation of franck–condon integrals of large molecules. *The Journal of chemical physics*, 122(24):244101, 2005.

[24] H Ding, TW Schmidt, T Pino, AE Boguslavskiy, F Güthe, and JP Maier. Gas phase electronic spectra of the linear carbon chains hc2n¿ h nä3–6, 9. *Journal of Chemical Physics*, 119(2), 2003.

[25] JP Doering. Electron impact study of the energy levels of trans-1, 3-butadiene. *The Journal of Chemical Physics*, 70(8):3902–3909, 1979.

[26] JP Doering and Ruth McDiarmid. Electron impact study of the energy levels of trans-1, 3-butadiene: Ii. detailed analysis of valence and rydberg transitions. *The Journal of Chemical Physics*, 73(8):3617–3624, 1980.

[27] JP Doering and Ruth McDiarmid. 100 ev electron impact study of 1, 3-butadiene. *The Journal of Chemical Physics*, 75(5):2477–2478, 1981.

[28] EV Doktorov, IA Malkin, and VI Man'ko. Dynamical symmetry of vibronic transitions in polyatomic molecules and the franck-condon principle. *J. Mol. Spectrosc.*, 64(2):302–326, 1977.

[29] W Domcke, C Woywod, and M Stengle. Diabatic casscf orbitals and wavefunctions. *Chemical physics letters*, 226(3):257–262, 1994.

[30] Wolfgang Domcke and Clemens Woywod. Direct construction of diabatic states in the casscf approach. application to the conical intersection of the 1

a 2 and 1 b 1 excited states of ozone. *Chemical physics letters*, 216(3):362–368, 1993.

[31] F Duschinsky. The importance of the electron spectrum in multi atomic molecules. concerning the franck-condon principle. *Acta Physicochim. URSS*, 7:551–566, 1937.

[32] Shirin Faraji, Susana Gómez-Carrasco, and Horst Köppel. Multistate vibronic dynamics and multiple conical intersections. *Conical Intersections: Theory, Computation and Experiment*, pages 249–300, 2011.

[33] Thomas R Faulkner and FS Richardson. On the calculation of polyatomic franck–condon factors: Application to the 1a1g 1b2u absorption band of benzene. *J. Chem. Phys.*, 70(3):1201–1213, 1979.

[34] Wayne M Flicker, Oren A Mosher, and Aron Kuppermann. Electron-impact investigation of excited singlet states in 1, 3-butadiene. *Chemical Physics*, 30(3):307–314, 1978.

[35] Jan Fulara, Patrick Freivogel, Daniel Forney, and John P Maier. Electronic absorption spectra of linear carbon chains in neon matrices. iii. hc2n+ 1h. *The Journal of chemical physics*, 103(20):8805–8810, 1995.

[36] Yukio Furukawa, Hideo Takeuchi, Issei Harada, and Mitsuo Tasumi. Molecular force fields of s-trans-1, 3-butadiene and the second stable conformer. *Bulletin of the Chemical Society of Japan*, 56(2):392–399, 1983.

[37] EL Gibb, MJ Mumma, N Dello Russo, MA DiSanti, and K Magee-Sauer. Methane in oort cloud comets. *Icarus*, 165(2):391–406, 2003.

[38] Susan L Graham, Peter B Kessler, and Marshall K Mckusick. Gprof: A call graph execution profiler. In *ACM Sigplan Notices*, volume 17, pages 120–126. ACM, 1982.

[39] Daniel Gruner and Paul Brumer. Efficient evaluation of harmonic polyatomic franck-condon factors. *Chem. Phys. Lett.*, 138(4):310–314, 1987.

[40] Robert J Hargreaves, Eric Buzan, Michael Dulick, and Peter F Bernath. High-resolution absorption cross sections of c 2 h 6 at elevated temperatures. *Molecular Astrophysics*, 1:20–25, 2015.

[41] WILLIAM HAUGEN and MARIT TRiETTEBERG. The molecular structures of 1, 3-butadiene and l, 3, 5-ir< ms-hexatriene. *Acta Chem. Scand*, 20(6), 1966.

[42] Anirban Hazra and Marcel Nooijen. Derivation and efficient implementation of a recursion formula to calculate harmonic franck–condon factors for polyatomic molecules. *Int. J. Quantum Chem.*, 95(4-5):643–657, 2003.

[43] RJ Hemley, JI Dawson, and V Vaida. Franck–condon analysis of the 1 1a-g 1 1b+ u transition of 1, 3-butadiene from absorption and raman intensities. *The Journal of Chemical Physics*, 78(6):2915–2927, 1983.

[44] Bruce S Hudson and Bryan E Kohler. A low-lying weak transition in the polyene α, ω-diphenyloctatetraene. *Chemical Physics Letters*, 14(3):299–304, 1972.

[45] Bruce S Hudson and Bryan E Kohler. Polyene spectroscopy: The lowest energy excited singlet state of diphenyloctatetraene and other linear polyenes. *The Journal of Chemical Physics*, 59(9):4984–5002, 1973.

[46] Bruce S Hudson, Bryan E Kohler, and Klaus Schulten. Linear polyene electronic structure and potential surfaces. *Excited states*, 6(1), 1982.

[47] Takatoshi Ichino, Adam J Gianola, W Carl Lineberger, and John F Stanton. Nonadiabatic effects in the photoelectron spectrum of the pyrazolide-d3 anion: three-state interactions in the pyrazolyl-d3 radical. *The Journal of chemical physics*, 125(8):084312, 2006.

[48] H-C Jankowiak, JL Stuber, and R Berger. Vibronic transitions in large molecular systems: Rigorous prescreening conditions for franck-condon factors. *J. Chem. Phys.*, 127(23):234101, 2007.

[49] Philip M Johnson. The multiphoton ionization spectrum of trans-1, 3-butadiene. *The Journal of Chemical Physics*, 64(11):4638–4644, 1976.

[50] Mihály Kállay and Jürgen Gauss. Calculation of excited-state properties using general coupled-cluster and configuration-interaction models. *The Journal of chemical physics*, 121(19):9257–9269, 2004.

[51] Kerstin Klein, Etienne Garand, Takatoshi Ichino, Daniel M Neumark, Jürgen Gauss, and John F Stanton. Quantitative vibronic coupling calculations: the formyloxyl radical. *Theoretical Chemistry Accounts*, 129(3-5):527–543, 2011.

[52] Henrik Koch, Ove Christiansen, Poul Jo, Alfredo M Sanchez de Merás, Trygve Helgaker, et al. The cc3 model: An iterative coupled cluster approach including connected triples. *The Journal of chemical physics*, 106(5):1808–1818, 1997.

[53] H Köppel, W Domcke, and LS Cederbaum. Theory of vibronic coupling in linear molecules. *The Journal of Chemical Physics*, 74(5):2945–2968, 1981.

[54] H Köppel, W Domcke, and LS Cederbaum. Multimode molecular dynamics beyond the born-oppenheimer approximation. *Advances in chemical physics*, 57:59–246, 1984.

[55] H Köuppel, W Domcke, and LS Cederbaum. Multimode molecular dynamics beyond the born-oppenheimer approximation. *Advances in Chemical Physics, Volume 57*, pages 59–246, 1984.

[56] Karol Kowalski and Piotr Piecuch. Excited-state potential energy curves of ch+: a comparison of the eomccsdt and full eomccsdt results. *Chemical physics letters*, 347(1):237–246, 2001.

[57] Robert P Krawczyk, Karsten Malsch, Georg Hohlneicher, Ralph C Gillen, and Wolfgang Domcke. 11bu–21ag conical intersection in trans-butadiene:

ultrafast dynamics and optical spectra. *Chemical Physics Letters*, 320(5):535–541, 2000.

[58] Stanisław A Kucharski, Marta Włoch, Monika Musiał, and Rodney J Bartlett. Coupled-cluster theory for excited electronic states: The full equation-of-motion coupled-cluster single, double, and triple excitation method. *The Journal of Chemical Physics*, 115(18):8263–8266, 2001.

[59] KC Kulander. Generalization of the faulkner–richardson method for calculating polyatomic franck–condon factors. *J. Chem. Phys.*, 71(6):2736–2737, 1979.

[60] KARI Kveseth, RAGNHILD Seip, DENIS A Kohl, et al. Conformational analysis. the structure and torsional potential of 1, 3-butadiene as studied by gas electron diffraction. *Acta Chem. Scand. A*, 34:31–42, 1980.

[61] Cornelius Lanczos. *An iteration method for the solution of the eigenvalue problem of linear differential and integral operators*. United States Governm. Press Office, 1950.

[62] Kin Long Kelvin Lee, Scott M Rabidoux, and John F Stanton. Cation states of ethane: Heat calculations and vibronic simulations of the photoelectron spectrum of ethane. *The Journal of Physical Chemistry A*, 2016.

[63] Olli Lehtonen, Dage Sundholm, Robert Send, and Mikael P Johansson. Coupled-cluster and density functional theory studies of the electronic excitation spec-

tra of trans-1, 3-butadiene and trans-2-propeniminium. *The Journal of chemical physics*, 131(2):024301, 2009.

[64] DG Leopold, RD Pendley, JL Roebber, RJ Hemley, and V Vaida. Direct absorption spectroscopy of jet-cooled polyenes. ii. the 1 1b+ u 1 1a- g transitions of butadienes and hexatrienes. *The Journal of chemical physics*, 81(10):4218–4229, 1984.

[65] Marco Mastrogiuseppe, Valerio Poggiali, Alexander Hayes, Ralph Lorenz, Jonathan Lunine, Giovanni Picardi, Roberto Seu, Enrico Flamini, Giuseppe Mitri, Claudia Notarnicola, et al. The bathymetry of a titan sea. *Geophysical Research Letters*, 41(5):1432–1437, 2014.

[66] Ruth McDiarmid. On the ultraviolet spectrum of trans-1, 3-butadiene. *The Journal of Chemical Physics*, 64(2):514–521, 1976.

[67] Ruth McDiarmid. On the valence excited states of conjugated polyenes. *The Journal of Chemical Physics*, 79(1):9–11, 1983.

[68] Ruth McDiarmid. An experimental estimate of rydberg-valence mixing in conjugated dienes. *Chemical physics letters*, 188(5):423–426, 1992.

[69] RUTH McDIARMID. On the electronic spectra of small linear polyenes. *Advances in chemical physics*, 110:177–214, 1999.

[70] Ruth McDiarmid and John P Doering. On the valence excited states of conjugated dienes. *Chemical Physics Letters*, 88(6):602–606, 1982.

[71] C Alden Mead and Donald G Truhlar. Conditions for the definition of a strictly diabatic electronic basis for molecular systems. *The Journal of Chemical Physics*, 77(12):6090–6098, 1982.

[72] Oren A Mosher, Wayne M Flicker, and Aron Kuppermann. Electronic spectroscopy of s-trans 1, 3-butadiene by electron impact. *The Journal of Chemical Physics*, 59(12):6502–6511, 1973.

[73] VA Mozhayskiy and AI Krylov. ezspectrum, h ttp. *iopenshell. usc. edu/-downloads*.

[74] Jarek Nieplocha, Manojkumar Krishnan, Vinod Tipparaju, and Bruce Palmer. Global arrays user manual. *Pacific Northwest National Laboratory, Richland, WA:.–URL http://www. emsl. pnl. gov/docs/global*, 2007.

[75] Jarek Nieplocha, Bruce Palmer, Vinod Tipparaju, Manojkumar Krishnan, Harold Trease, and Edoardo Aprà. Advances, applications and performance of the global arrays shared memory programming toolkit. *International Journal of High Performance Computing Applications*, 20(2):203–231, 2006.

[76] Marcel Nooijen. First-principles simulation of the uv absorption spectrum of ketene. *International journal of quantum chemistry*, 95(6):768–783, 2003.

[77] Bojana Ostojić and Wolfgang Domcke. Ab initio investigation of the potential energy surfaces involved in the photophysics of s-trans-1, 3-butadiene. *Chemical Physics*, 269(1):1–10, 2001.

[78] David L Phillips, Marek Z Zgierski, and Anne B Myers. Resonance raman excitation profiles of 1, 3-butadiene in vapor and solution phases. *The Journal of Physical Chemistry*, 97(9):1800–1809, 1993.

[79] Scott M Rabidoux, Victor Eijkhout, and John F Stanton. Parallelization strategy for large-scale vibronic coupling calculations. *The Journal of Physical Chemistry A*, 118(51):12059–12068, 2014.

[80] Scott M Rabidoux, Victor Eijkhout, and John F Stanton. A highly-efficient implementation of the doktorov recurrence equations for franck–condon calculations. *Journal of chemical theory and computation*, 12(2):728–739, 2016.

[81] Scott Michael Rabidoux. *https://bitbucket.org/rabism7/fc_squared*. Accessed December 23, 2015.

[82] Tim Reddish, Barry Wallbank, and John Comer. Electron energy-loss studies of trans-1, 3-butadiene. *Chemical physics*, 108(1):159–165, 1986.

[83] ST Ridgway. Jupiter: Identification of ethane and acetylene. *The Astrophysical Journal*, 187:L41–L43, 1974.

[84] Walter Ritz. Über eine neue methode zur lösung gewisser variationsprobleme der mathematischen physik. *Journal für die reine und angewandte Mathematik*, 135:1–61, 1909.

[85] LJ Rothberg, DP Gerrity, and V Vaida. Electronic spectra of butadiene and its methyl derivatives: A multiphoton ionization study. *The Journal of Chemical Physics*, 73(11):5508–5513, 1980.

[86] Peder Thusgaard Ruhoff and Mark A Ratner. Algorithms for computing franck–condon overlap integrals. *Int. J. Quantum Chem.*, 77(1):383–392, 2000.

[87] Fabrizio Santoro, Roberto Improta, Alessandro Lami, Julien Bloino, and Vincenzo Barone. Effective method to compute franck-condon integrals for optical spectra of large molecules in solution. *J. Chem. Phys.*, 126(8):084509, 2007.

[88] Michael S Schuurman, Daniel E Weinberg, and David R Yarkony. On the simulation of photoelectron spectra in molecules with conical intersections and spin-orbit coupling: The vibronic spectrum of ch3s. *The Journal of chemical physics*, 127(10):104309, 2007.

[89] Michael S Schuurman and David R Yarkony. A method to reduce the size of the vibronic basis employed in the simulation of spectra using the multimode vibronic coupling approximation. *The Journal of chemical physics*, 128(4):044119, 2008.

[90] Michael S Schuurman, Richard A Young, and David R Yarkony. On the multimode quadratic vibronic coupling problem: An open-ended solution using a parallel lanczos algorithm. *Chemical Physics*, 347(1):57–64, 2008.

[91] Luis Serrano-Andrés, Manuela Merchán, Ignacio Nebot-Gil, Roland Lindh, and Björn O Roos. Towards an accurate molecular orbital theory for excited states: Ethene, butadiene, and hexatriene. *The Journal of chemical physics*, 98(4):3151–3162, 1993.

[92] TE Sharp and HM Rosenstock. Franckcondon factors for polyatomic molecules. *The Journal of Chemical Physics*, 41(11):3453–3463, 1964.

[93] Christopher S Simmons, Takatoshi Ichino, and John F Stanton. The $\nu 3$ fundamental in no3 has been seen near 1060 cm–1, albeit some time ago. *The Journal of Physical Chemistry Letters*, 3(15):1946–1950, 2012.

[94] JF Stanton, J Gauss, ME Harding, PG Szalay, AA Auer, RJ Bartlett, U Benedikt, C Berger, DE Bernholdt, YJ Bomble, et al. Cfour, a quantum chemical program package. *For the current version, see http://www. cfour. de*, 2009.

[95] John F Stanton. On the vibronic level structure in the no3 radical. i. the ground electronic state. *The Journal of chemical physics*, 126(13):134309, 2007.

[96] John F Stanton. Quantitative vibronic coupling calculations. the visible spectrum of propadienylidene. *Faraday discussions*, 150:331–343, 2011.

[97] John F Stanton and Rodney J Bartlett. The equation of motion coupled-cluster method. a systematic biorthogonal approach to molecular excitation energies, transition probabilities, and excited state properties. *The Journal of chemical physics*, 98(9):7029–7039, 1993.

[98] John F Stanton and Jurgen Gauss. Analytic second derivatives in high-order many-body perturbation and coupled-cluster theories: computational considerations and applications. *International Reviews in Physical Chemistry*, 19(1):61–95, 2000.

[99] Gary D Strahan and Bruce S Hudson. The vacuum ultraviolet excited electronic states of 1, 3-butadiene: Selective enhancement of vibrational modes in resonant raman transitions. *The Journal of chemical physics*, 99(8):5780–5789, 1993.

[100] P Swiderek, M Michaud, and L Sanche. Electron-energy-loss spectroscopy of condensed butadiene and cyclopentadiene: Vibrationally resolved excitation of the low-lying triplet states. *The Journal of chemical physics*, 98(11):8397–8405, 1993.

[101] V Vaida and GM McClelland. Electronic absorption spectroscopy of cooled supersonic expansions: dynamics of the 1 b 1u state of trans-butadiene. *Chemical Physics Letters*, 71(3):436–439, 1980.

[102] Veronica Vaida, Robert E Turner, John L Casey, and Steven D Colson. Multiphoton transitions in trans-butadiene observed by multiphoton ionization and thermal lensing spectroscopy. *Chemical Physics Letters*, 54(1):25–29, 1978.

[103] Mark A Watson and Garnet Kin-Lic Chan. Excited states of butadiene to chemical accuracy: Reconciling theory and experiment. *Journal of chemical theory and computation*, 8(11):4013–4018, 2012.

[104] John D Watts, Steven R Gwaltney, and Rodney J Bartlett. Coupled-cluster calculations of the excitation energies of ethylene, butadiene, and cyclopentadiene. *The Journal of chemical physics*, 105(16):6979–6988, 1996.

[105] Edgar Bright Wilson. *Molecular vibrations: the theory of infrared and Raman vibrational spectra*. Courier Dover Publications, 1955.

[106] David E Woon and Thom H Dunning Jr. Gaussian basis sets for use in correlated molecular calculations. iv. calculation of static electrical response properties. *The Journal of chemical physics*, 100(4):2975–2988, 1994.

[107] HJ Wörner, X Qian, and F Merkt. Jahn-teller effect in tetrahedral symmetry: Large-amplitude tunneling motion and rovibronic structure of ch4+ and cd4+. *The Journal of chemical physics*, 126(14):144305, 2007.

[108] HJ Wörner, R van der Veen, and F Merkt. Jahn-teller effect in the methane cation: Rovibronic structure and the geometric phase. *Physical review letters*, 97(17):173003, 2006.