Examining the Vibrational Couplings and Dynamics in Cyanide-Bridged Transition Metal Mixed Valence Complexes Using Ultrafast Nonlinear Infrared Spectroscopy

Karla M. Slenkamp

A dissertation

submitted in partial fulfillment of the

requirements for the degree of

Doctor of Philosophy

University of Washington

2015

Reading Committee:

Munira Khalil, Chair

Charles T. Campbell

Xiaosong Li

Program Authorized to Offer Degree:

Chemistry

University of Washington

Abstract

Examining the Vibrational Couplings and Dynamics in Cyanide-Bridged Transition Metal Mixed Valence Complexes Using Ultrafast Nonlinear Infrared Spectroscopy

Karla M. Slenkamp

Chair of the Supervisory Committee:
Associate Professor Munira Khalil
Chemistry

Charge transfer processes are essential to any number of functions that are necessary for everyday life. Photosynthesis, respiration, and obtaining energy from solar cells are just a small sample of functions that rely on moving electrons around. Cyanide-bridged mixed valence complexes serve as useful models to study the coupling between electronic and nuclear motions during charge transfer reactions by using the cyanide stretching (ν_{CN}) vibrations as probes. However, little is known about the ν_{CN} modes for these classes of molecules on the electronic ground state. Polarization-selective two-dimensional infrared (2D IR) spectroscopy was used to measure anharmonic couplings and angles between the transition dipole moments of the four ν_{CN} vibrations found in $[(NH_3)_5Ru^{III}NCFe^{II}(CN)_5]^-$ (FeRu) and $[(NC)_5Fe^{II}CNPt^{IV}(NH_3)_4NCFe^{II}(CN)_5]^{4-}$ (FePtFe) by fitting the experimental spectra. For this

study, FeRu was dissolved in both formamide and D_2O, and FePtFe was dissolved in D_2O. IR pump-probe and 2D IR spectroscopies were also used to study vibrational relaxation of the ν_{CN} modes in both complexes. The experimental 2D IR spectra of FeRu and FePtFe and their fits reveal a set of weakly coupled anharmonic ν_{CN} modes with the vibrational mode anharmonicities ranging from 14 to 28 cm^{-1} and the mixed-mode anharmonicities ranging from 2 to 14 cm^{-1}. Measurement of the relative transition dipole moments of the four ν_{CN} modes reveal that FeRu appears almost linear in solution when dissolved in formamide, but bent when dissolved in D_2O. Fits of FePtFe dissolved in D_2O reveal that the molecule also appears bent. IR pump-probe experiments reveal that the vibrational lifetime of the radial, axial, and trans ν_{CN} modes are ~2 times faster when FeRu is dissolved in D_2O versus formamide, and the lifetimes of the ν_{CN} modes of FePtFe are ~4 times faster than FeRu when both are dissolved in D_2O. Finally, the synthesis and basic characterization of new cyanide containing Ru^{II} polypyridine complexes are discussed. These complexes contain a metal-to-ligand charge transfer that relaxes into a longer lived triplet state. This longer lifetime will allow for a more in-depth analysis of vibrational modes on an electronic excited state as well as simplifying the analysis by reducing the number of ν_{CN} modes from the four seen in FeRu and FePtFe to one or two in the Ru polypyridine complexes.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

So many people have supported me during my almost six years at the University of Washington. The work that made it into this dissertation, and all the work that didn't, would not have been possible without them.

First I have to thank all of the members of the Khalil research group starting with Munira Khalil. I would like to thank her for allowing me to join her group and introducing me to the world of ultrafast spectroscopy. It was an intimidating world to walk into, but an exciting one as well. She always required excellence, and because of that, I can be proud of the work I completed in her group and the work I present in this dissertation.

I need to thank Jennifer Brookes and Anthony Reynolds. We joined the Khalil group together, and I'm not sure I would have made it to the end without them. They were always there when I had a 'stupid' question, and they never judged me…well not much anyway. Over the past six years, they have become members of my family, ones that I have grown with, ones that I have laughed, cried, and overcome difficulties with. They have made graduate school a more rewarding and enriching experience.

I also need to thank former group member Michael Lynch for sharing his experimental knowledge with me. We may not have gotten along in lab at all times, but Mike was still a great friend, and I learned so much from him. Ben Van Kuiken shared theoretical knowledge. Deciphering his answers could be a challenge at times, but he was always willing to hear my questions and answer more than the question I had asked. Two wonderful undergraduates, Laura Estergreen and Caitlin Bannan, synthesized most of the compounds I studied at UW. Thank you

to all other members of the Khalil group during my time at the University of Washington for being friends and valuable sources of information.

Outside of my group, there were numerous people within the chemistry department that helped me survive the trials of graduate school. Thank you to Glennis Rayermann and Joseph May for numerous Monday night nacho sessions. Thanks to Trevor James for being my bus buddy before we even officially met the very first day of orientation in September 2009. He made our longer than normal commutes more interesting. And thank you to other members of my incoming year for their support including Tom Porter, Michael Roberto, Sam Marionni, Sam Connelly, Megan Duda, Emilie Viglino, and Nathan Sylvain.

Any acknowledgements section would be incomplete without thanking the people in my life that had nothing to do with the University of Washington. I have to thank (and blame) my friend Sam Backus. He convinced me to apply to graduate school, and he has been listening to me complain ever since. He has heard every problem I've encountered the last six years, and I truly do owe my continuing sanity to him. I have to thank Kelsey Roe for also listening to me complain. And I have to thank Camille and Josh Lider, Bonnie McAllister, Nena Slocum, Bridgette Hannifin, Avery Stewart, and Rachel Unterreiner for reminding me the world at large doesn't revolve around chemistry, even if mine has for almost 10 years now.

Last but not least, I have to thank my family. I have a large extended family, and thanking them individually here would take more pages than anyone would be willing to read. But they have all provided me with support and love and an escape from academic life. In particular, I need to thank my grandparents for emotional support. I need to thank my Grandma for sitting down and being actively engaged whenever I was discussing my research. I need to thank my sister, Shelby, for making me food and occasionally helping me clean my home when life got too hectic, even if she

grumbled the whole time. And finally, I need to thank my parents, Robert and Beverly Slenkamp. They have been amazing! I don't have enough words to express my gratitude for all the ways they have supported me in reaching my goals, both emotionally and materially. I couldn't have done it without you. I love you!

DEDICATION

To my family…

every last one of you!

Chapter 1:

Introduction

1.1 Importance of Charge Transfer Reactions

Charge transfer processes are essential to any number of functions that are necessary for everyday life. Photosynthesis, respiration, and obtaining energy from solar cells are just a small sample of functions that rely on moving electrons around. However, directly studying a system like photosynthesis is difficult because of its complexity; there is more to photosynthesis than moving one electron around. Instead, mixed valence transition metal complexes are much simpler systems, but they still exhibit these electron transfer processes.[1,2] More famous examples of mixed valence compounds include Prussian Blue,[3,4] used as a deep blue dye, and the Creutz-Taube ion,[5,6] used primarily to study inner sphere electron transfer. Electron transfer can be initiated by photons with sufficient energy to reach the charge transfer bands. Cyanide-bridged mixed valence transition metal complexes in particular are a class of molecules that exhibit unique redox, spectroscopic, and charge transfer properties and are widely studied for applications in solar energy conversion, molecular electronics, or photocatalysis.[7-9] Choosing cyanide as a bridging ligand between metal centers moves the metal-to-ligand charge transfer (MLCT) bands to higher energy keeping them from obscuring the metal-to-metal charge transfer (MMCT) bands in the near-infrared to visible region of the spectrum.[10,11] Cyanide-bridged complexes also serve as useful models to study the coupling between electronic and nuclear motions during charge transfer reactions.[12-31] Photochemical reactions in solution at room temperature can occur in less than 100

millionth-of-a-billionth-of-a-second, or in less than 100 femtoseconds (10^{-15} s). This is on the order of molecular vibrations in solution which have frequencies in the range of 50 – 3500 cm^{-1} (1.5 – 105 THz) corresponding to periods of 10 – 680 fs. Vibrational relaxation also occurs on a femtosecond to picosecond (10^{-12} s) timescale. Many high-frequency vibrations (> 1500 cm^{-1}), cyanide included, can provide spatial information because the vibrations are relatively localized within one part of the molecule. This makes vibrations an ideal tool to study photoinduced charge transfer reactions.

The role high frequency vibrations play in electron transfer processes really started to interest groups in the 1980s.[32-36] Cyanide-bridged transition metal mixed valence dimer (two metal centers) systems of the form $[(NH_3)_5M^{III}NCM^{II}(CN)_5]^-$, where M = Fe, Os and Ru, have been widely studied since the late 1980s to follow the timescales of electron transfer kinetics, coherent wavepacket motion on the excited/ground electronic state, and vibrational excitation/relaxation of the cyanide stretching modes (ν_{CN}) following MMCT excitation.[15,16,18,20-22,24,25,36-40] Transition metal mixed valence trimer systems like $[(NC)_5M^{II}CNPt^{IV}(NH_3)_4NCM^{II}(CN)_5]^{4-}$, where M = Fe or Ru, have been studied for similar purposes since the early 1990s.[11,26,41,42] The IR spectroscopy of the bridging cyanide ligand and its ability to probe the electron delocalization and coupling between the metal centers in mixed valence complexes is of great interest. Detailing how inter- and intramolecular vibrational energy flow in ultrafast photoinduced electron transfer processes relates to molecular conformational changes and the efficacy of the charge transfer process is critical for understanding basic and applied phenomena in chemistry, biology, and materials science. Using these relatively simple cyanide-bridged transition metal complexes to study the interplay between electronic transitions and vibrational motions allows for a better understanding

of the fundamental process of charge transfer without becoming overwhelmed by all that might be happening in a larger molecule like a protein.

1.2 Introduction to FeRu and FePtFe

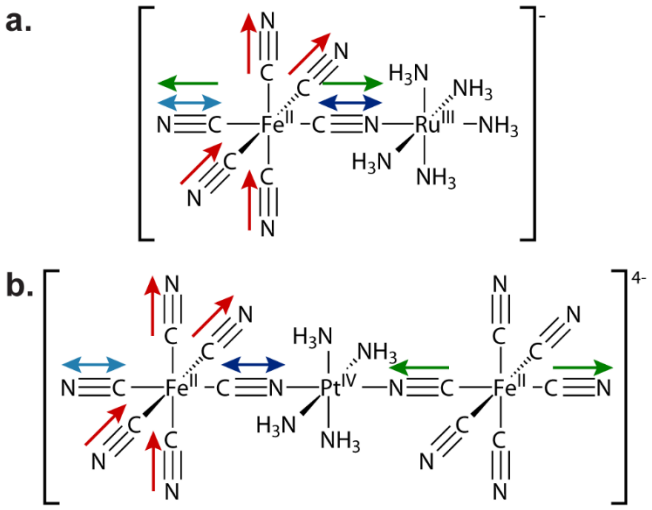


Figure 1.1: The molecular structures of (a) FeRu and (b) FePtFe. Colored arrows correspond to simplified depictions of four ν_{CN} modes. The transition dipole moments of the ν_{trans} (ν_t, blue), ν_{axial} (ν_a, green), and ν_{bridge} (ν_b, purple) modes lie along the MMCT axis and are perpendicular to the degenerate ν_{radial} (ν_r, red) mode.

Two commonly studied cyanide-bridged mixed valence dimer and trimer systems are $[(NH_3)_5Ru^{III}NCFe^{II}(CN)_5]^-$, or FeRu, and $[(NC)_5Fe^{II}CNPt^{IV}(NH_3)_4NCFe^{II}(CN)_5]^{4-}$, or FePtFe. Both molecules are shown as cartoons in Figure 1.1. The FeRu molecule undergoes a photoinduced MMCT, transferring an electron from the iron atom to the ruthenium atom, when in solution and excited by ~1000 nm light.[15,16]

$$[(NH_3)_5Ru^{III}NCFe^{II}(CN)_5]^- \xrightarrow{h\nu} [(NH_3)_5Ru^{II}NCFe^{III}(CN)_5]^-$$

This is followed almost immediately by back electron transfer (BET), which shifts the electron density back to the iron atom within ~100 fs. When photoexcited at 400 nm, FePtFe also undergoes MMCT moving an electron from one of the iron centers to the platinum atom, at which point the

electron is transferred back to the iron atom in ~100 fs with ~99% efficiency to form a "hot" electronic ground state.

$$[(NC)_5Fe^{II}CNPt^{IV}(NH_3)_4NCFe^{II}(CN)_5]^{4-} \xrightarrow{h\nu} [(NC)_5Fe^{III}CNPt^{III}(NH_3)_4NCFe^{II}(CN)_5]^{4-}$$

The other ~1% of the time, a second electron from the other iron atom is transferred to the platinum yielding $Fe^{III}Pt^{II}Fe^{III}$. This product immediately decomposes into $2[Fe^{III}(CN)_6]^{3-}$ and $[Pt^{II}(NH_3)_4]^{2+}$.[11,41] Spectral electrochemistry has been used to study the nature of the MMCT bands in both the dimer and trimer molecules.[11,21,26,40,43-49] This work depicted a simplified description of the energetics of the dimer and trimer molecules, where the potential energy is given by a two- or three-parabola potential energy surface in the Marcus-Hush picture, pictured in Figure 1.2 for both molecules. Resonance Raman data and time-dependent intensity analysis by several research

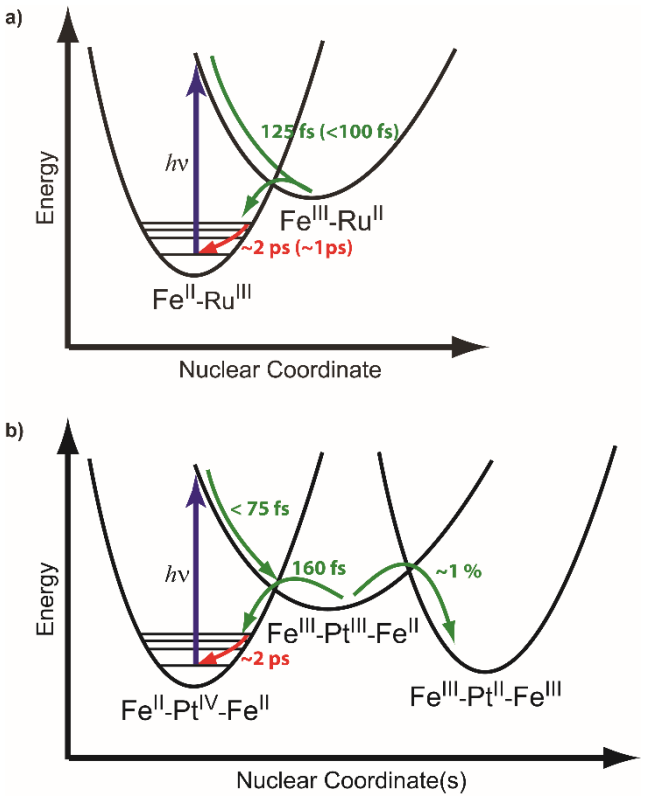


Figure 1.2: Potential energy surface showing MMCT and BET in FeRu (a) and FePtFe (b). Timescales for FeRu dissolved in D_2O (FA).[15,16] Figure adapted from Pfennig and Bocarsly.[41]

groups have shown that >40% of the inner sphere reorganization energy involves vibrational modes which lie along the intervalance charge transfer axis of the complex.[26,41,42,49,50] Static and time-resolved experiments have been analyzed to show that the majority of the reorganization energy arises from solvent effects in transition metal mixed valence complexes.[15,16,18] FeRu and FePtFe both have four ν_{CN} frequencies that are labeled ν_{trans} (ν_t), ν_{radial} (ν_r), ν_{axial} (ν_a), and ν_{bridge} (ν_b) (illustrated in Fig. 1.1a,b) of which three (ν_t, ν_a, and ν_b) lie approximately along the MMCT axis.[28,51]

Starting in the late 1980s, Barbara, Hupp, Walker, and Woodruff began studying mixed valence complexes using vibrational spectroscopy. Barbara and Hupp focused on Raman spectroscopy using it to back out parameters relevant to the electron transfer such as classical reorganization energy and electronic coupling.[16,37] By studying FeRu with picosecond time-resolved infrared spectroscopy, Walker and coworkers found excitation in ν_{CN} modes after BET in when the molecule was dissolved in both formamide (FA) and deuterated water (D_2O).[20] Three ν_{CN} modes were assigned which they called $\nu_{bridge\text{-}CN}$, $\nu_{cis\text{-}CN}$, and $\nu_{trans\text{-}CN}$. Walker and company's resonance Raman work on FeRu showed that the solvent controls the degree of vibronic coupling for the trans ν_{CN} mode in the MMCT transition.[21] They have also concluded that solvent induced changes in the structure of the molecule may result from the strong interaction between the solvent and $\nu_{trans\text{-}CN}$. In the similar $[(NH_3)_5Ru^{III}NCOs^{II}(CN)_5]^-$ system, they found that vibrational levels n = 0, n = 1, and n = 2 of cis and trans CN were populated after BET.[22] It has also been observed by Woodruff and coworkers that after BET in $[(NH_3)_5Ru^{III}NCRu^{II}(CN)_5]^-$ there is population in ν_{CN} modes in vibrationally excited states n = 17.[24] Around the same time, Endicott and coworkers studied a series of cyanide-bridged transition metal complexes. They found a simple vibronic coupling model could be used to understand the frequencies of the ν_{CN} within the complexes.[29-31]

As mentioned above Bocarsly and co-workers first synthesized a transition metal mixed valence trimer system of the form, $[(NC)_5M^{II}CNPt^{IV}(NH_3)_4NCM^{II}(CN)_5]^{4-}$, where M = Fe or Ru in 1990, and showed that it undergoes an ultrafast multielectron transfer process following MMCT excitation.[11,41] The multielectron transfer process discussed above made the trimer a potentially more interesting molecule than FeRu, but the small percentage that undergoes the second MMCT makes it very difficult to study that additional aspect.

Our group has obtained X-ray absorption spectroscopy results at the Ru L3-edge of FeRu.[52] These results highlighted the important role hydrogen-bonding plays in the solvent modulation of charge around transition metal ions in solution. We have also investigated the role of the ν_{CN} modes in the MMCT excitation, and subsequent BET, reaction of FePtFe dissolved in D_2O.[27,28,53]

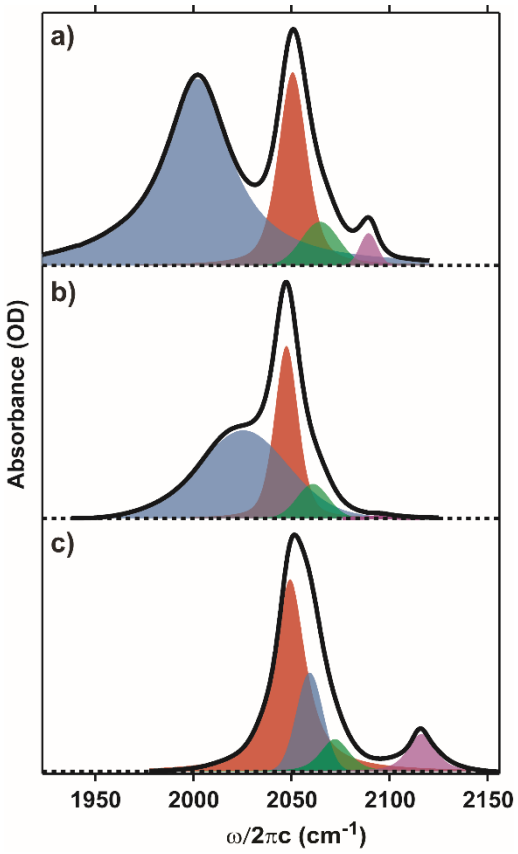


Figure 1.3: FTIR of FeRu in FA (a), FeRu in D_2O (b), and FePtFe in D_2O (c). Colors correspond to the colors of the modes seen in Figure 1.1.

The four ν_{CN} modes (see Figure 1.1b) are a multidimensional probe that can be used to study vibrational energy relaxation following the MMCT transition. Using sub-100 fs transient infrared spectroscopy, we found that BET to the ground state occurs in ~100 fs and it deposits greater than 6 quanta of vibrational energy into ν_b.[28] Using transient heterodyne-detected dispersed vibrational echo (t-HDVE), a fifth order nonlinear visible-infrared spectroscopy, the dynamics of the ν_{CN} modes can be monitored as a function of time after visible excitation (τ_{vis}).[53] In other words, the vibrational dynamics can be monitored as a photochemical reaction occurs.[27] This technique provided experimental evidence that ν_{CN} modes are both coherently and incoherently coupled to the charge transfer reaction. Vibrational relaxation timescales change as τ_{vis} increases moving the system back towards equilibrium, and a change in the frequency of a ~57 cm^{-1} oscillation indicates that the coupling between the probed ν_{CN} modes changes $\tau_{vis} < 500$ fs. The Fourier transform infrared (FTIR) spectra of the ν_{CN} modes (see Fig. 1.3a-c) of the bimetallic and trimetallic systems are strikingly different in terms of frequencies, amplitudes, and lineshapes. Why is this the case? What is the difference between the solvent or molecules that allows for this? Previous studies, both by our group and others, lack a detailed understanding of the electronic ground state in these systems.

These time-resolved and spectral studies have identified the importance of the solvent and vibrations in the MMCT process and measured the relevant rate constants. The goal of this dissertation is to understand how the solvent modulates the molecular geometry and the anharmonic couplings between the bridging and terminal cyanide stretching vibrations as well as the relaxation dynamics in cyanide-bridged bimetallic and trimetallic transition metal mixed valence complexes in the electronic ground state (without the MMCT). Molecular geometry and coupling affect how modes interact with each other as well as how modes interact with the rest of

their environment. Knowing the role these parameters play can have implications for the design of molecules or determining their spectroscopic properties. Vibrational relaxation dynamics reflect the differences in local environment and the interactions that occur between the solvent and the local vibrational probe. Changing dynamics between systems, or even different vibrational modes within a molecule, show differences in how energy is transferred from the vibrational mode to its environment. Ultrafast nonlinear infrared spectroscopy includes the most common techniques used to study vibrational coupling and relaxation in the condensed phase. We use infrared pump-probe and polarization-selective two-dimensional infrared spectroscopies to measure anharmonic couplings, angles between the transition dipole moments, and electronic ground state vibrational dynamics of the cyanide stretching vibrations of FeRu dissolved in D_2O and formamide (FA) and FePtFe dissolved in D_2O.

1.3 Ultrafast Nonlinear Vibrational Spectroscopy

One of the most comprehensive ways to gain information about a system's vibrational potential energy surface is two-dimensional infrared spectroscopy (2D IR). Linear Fourier transform spectroscopy (FTIR) can give you some information like peak positions and relative amplitudes, but it is insensitive to coupling and distributions underneath the peak envelope. Two-dimensional infrared spectroscopy allows us to spread the spectra out in two dimensions (ω_1 and ω_3) and monitor how the spectra change as a function of time (τ_2). It allows for the measurement of vibrational anharmonicities, vibrational relaxation dynamics, vibrational frequency-frequency correlation functions (FFCF), and coupling between vibrational modes as well as determining how much of a peak's width is due to homogeneous versus inhomogeneous broadening.

Figure 1.4 depicts an energy diagram for two coupled vibrational modes, ν_p and ν_q. Two coupled modes produce a six-level system consisting of a common ground state, two first-excited states,

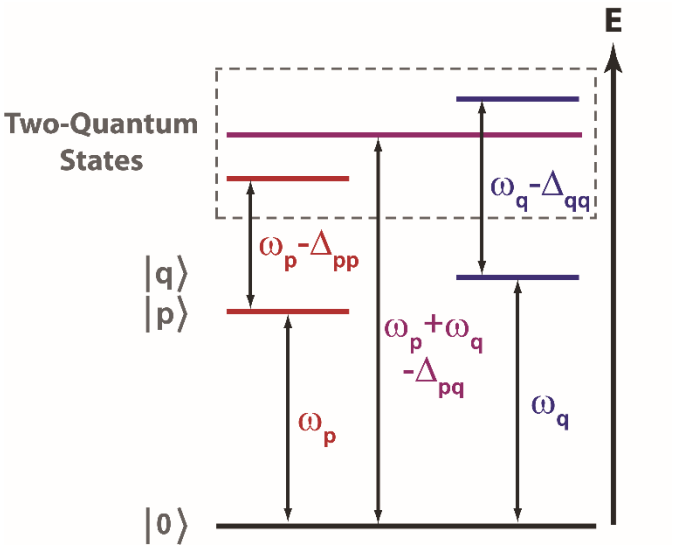


Figure 1.4: An example of an energy level diagram for a system of two coupled vibrations, modes *p* and *q*. Diagonal and mixed-mode anharmonicities are labeled.

two second-excited states (vibrational overtones), and a combination band that is a linear combination of both modes. There are three anharmonicities labeled in Figure 1.4 that relate the position of two-quantum states to the frequency of v = 0 → v = 1 transition. The two that show how far from harmonic each mode is are called diagonal anharmonicities and are labeled Δ_{pp} and Δ_{qq}. The last anharmonicity, Δ_{pq}, is the mixed-mode anharmonicity, and it is a measure of how strongly modes p and q are coupled. The larger the value of Δ_{pq}, the stronger the coupling.

Figure 1.5 shows a simulated 2D IR spectrum of a six-level system at a single τ_2 point as well as the simulated FTIR spectrum. The positive (red) peaks represent transitions to one quantum states and the negative (blue) peaks represent transitions to two-quantum states. The separations between the positive and negative peaks in each pair reflect the anharmonicities of the system. In a perfectly harmonic system, the anharmonicities would be zero and the positive and negative peaks would cancel out. Peaks along the diagonal (where $\omega_1 = \omega_3$) represent the v = 0 → v = 1 transition for each mode. The widths of the peaks along the diagonal tell us how inhomogeneous the environment for that mode is, while the anti-diagonal width tells us about homogeneous

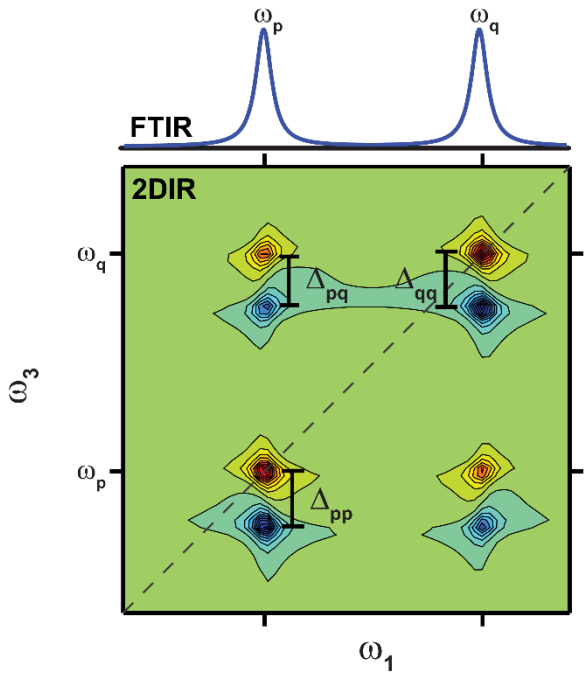


Figure 1.5: Simulated FTIR and 2D IR spectra for a system of two coupled vibrations, ν_p and ν_q, with homogeneous lineshapes. Diagonal and mixed-mode anharmonicities are labeled. The 2D IR spectrum was generated with code found in Appendix A.

broadening. In addition to the two sets of peaks along the diagonal, there are two additional sets of peaks seen in Figure 1.5 where $\omega_1 = \omega_p$ and $\omega_3 = \omega_q$, or vice versa. These are called crosspeaks, and their existence proves that the two modes are indeed coupled. While all of these parameters are useful to understand a system, if we want to learn about the system's dynamics, we need to look at a series of spectra at multiple τ_2 delays.

Figure 1.6 shows a series of 2D IR spectra as a function of τ_2. Something in the system is changing as τ_2 increases. The peaks "round-out", the cross-peaks appear to grow and/or beat in relation to the diagonal peaks, and the overall amplitude of the spectra decrease. The peaks round out as a result of spectral diffusion, and the rate at which this happens is proportional to the FFCF for that mode. Frequency-frequency timescales measure how long it takes for the vibrational mode

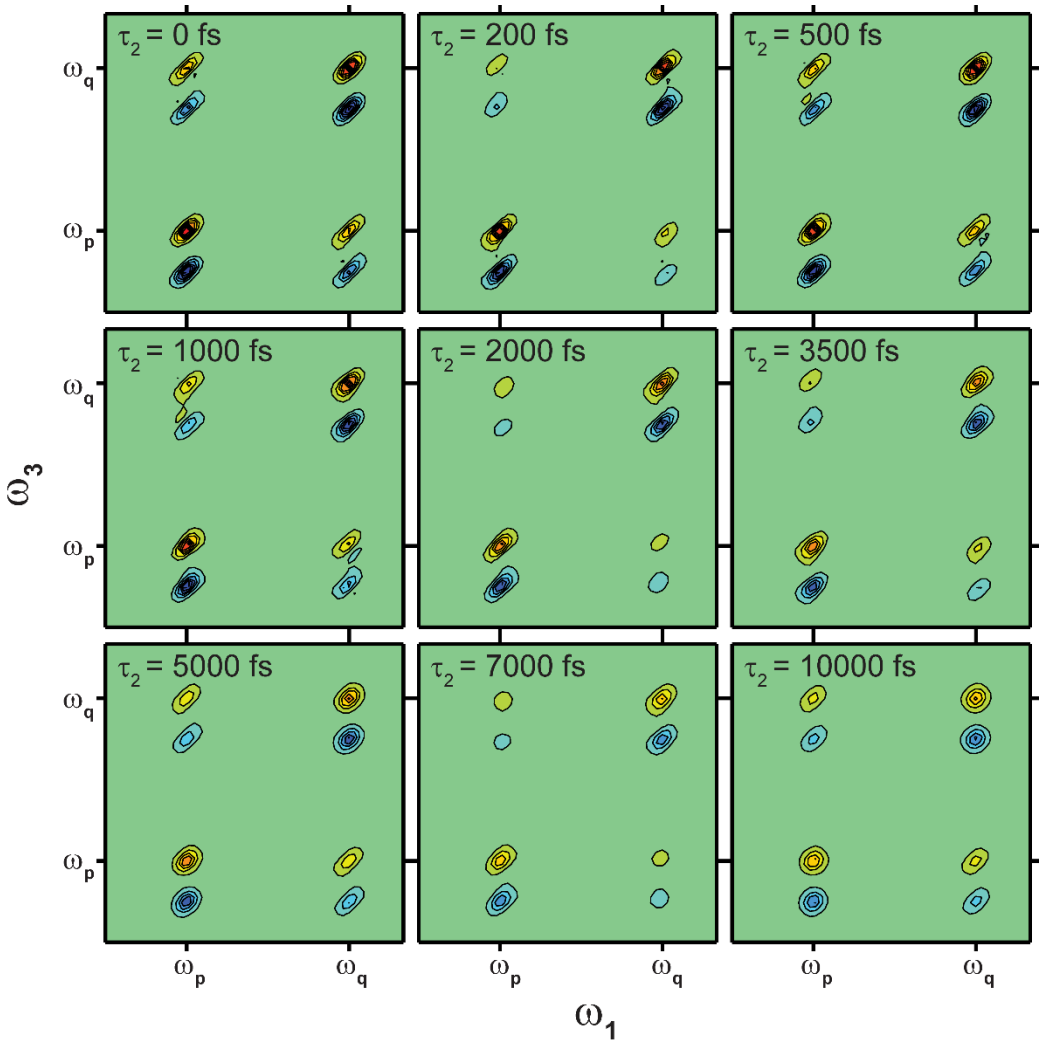


Figure 1.6: A series of simulated 2D IR spectra (code in Appendix A) for two coupled vibrations at increasing τ_2 delays. All spectra are plotted using the same contour levels and color scheme so amplitudes of peaks can be directly compared.

to sample all of its possible environments. Depending on the system being measured, FFCF timescales can be attributed to things like the breaking and forming of hydrogen bonds, or they can simply remark on the mode's environment; maybe the environment for this mode is more polar than for another mode. Growth in the cross-peaks as τ_2 increases measures population transfer between modes; energy starts out in one mode ($\omega_1 = \nu_p$) but ends up in another ($\omega_3 = \nu_q$). Following the rates of energy transfer gives important information about how energy is transferred within the molecule and the route it takes before escaping into the environment. The overall decrease in

amplitude for all parts of the spectrum as τ_2 increases is a result of population relaxation. As the modes relax back to their ground state, there is less and less signal to measure. Full 2D IR spectra take a significant amount of time to collect which limits the number of τ_2 points obtained during each experiment. Because of this, population relaxation measurements are usually performed using another ultrafast technique, IR pump-probe. This technique collapses a 2D IR spectrum onto the ω_3 axis, so all information can be collected in one dimension. An example of a series of IR pump-probe spectra for increasing τ_2 delays are shown in Figure 1.7. It is easy to see the decay

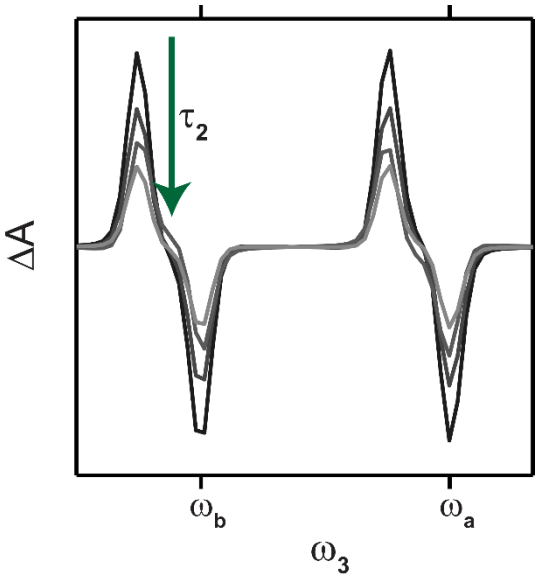


Figure 1.7: A series of IR pump-probe spectra for several τ_2 points for a system with two modes. Population relaxation dynamics can be measured by quantifying how fast each feature decays to zero. Negative going peaks represent v = 0 → v = 1 transitions, while positive going peaks represent v = 1 → v = 2 transitions.

due to population relaxation. These are much faster to collect working in lab so more τ_2 points can be measured giving a more detailed look at relaxation dynamics. All of the dynamics discussed in this paragraph are described in further detail in Chapter 3.

Finally, 2D IR can be used to calculate angles, θ_{pq}, between the transition dipole moments of coupled modes. Figure 1.8 shows 2D IR spectra at two different polarizations, where the first and second pulse have polarizations either parallel (ZZZZ) or rotated 90° (YYZZ) to the third pulse

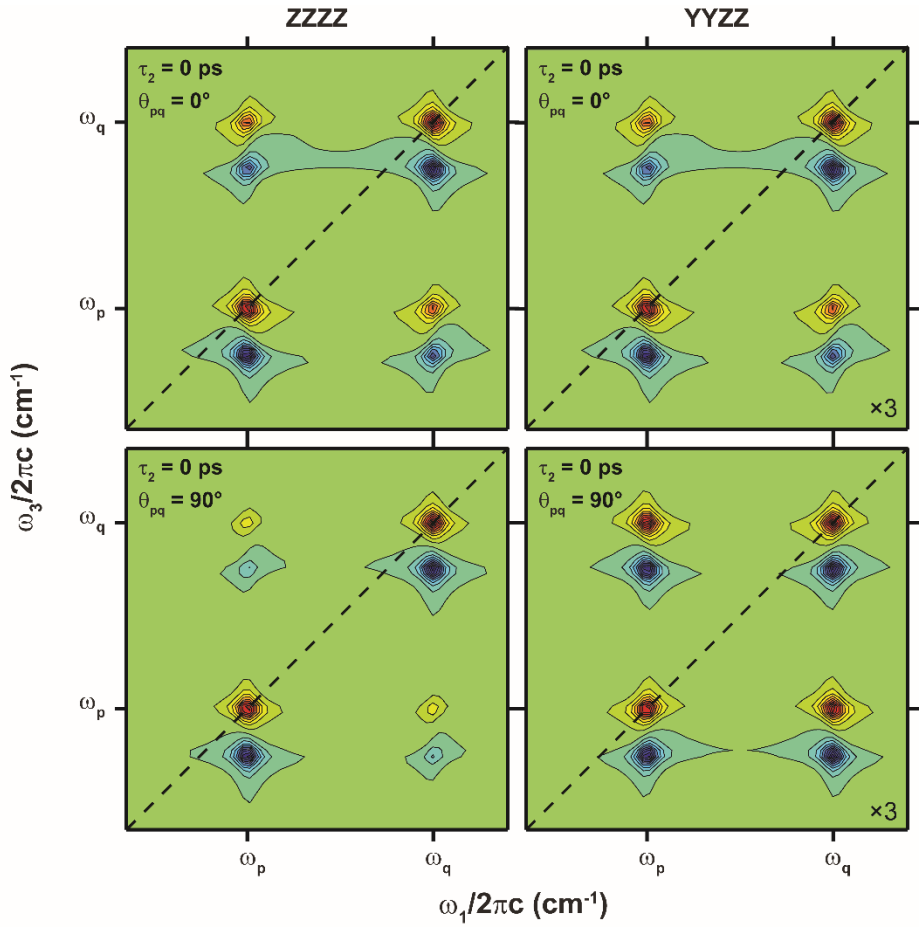


Figure 1.8: Simulated 2D IR spectra (code in Appendix A) for coupled vibrations with two different relative angles between the transition dipole moments of ν_p and ν_q. The relative angles between the dipoles can be determined by comparing the amplitudes of the cross-peaks in the two different geometries, ZZZZ and YYZZ. Note the amplitude correction in the corner of each YYZZ spectrum.

and signal field, and for two different θ_{pq}. In order to extract the angle, the ratio of the amplitudes between the same cross-peak in different polarizations must be compared. Because transition dipole moments at different orientations will react more strongly with light polarized in a similar orientation, the strength of the cross-peaks will change with different polarizations. This is seen in Figure 1.8, where the cross-peaks in the ZZZZ geometry are much weaker when $\theta_{pq} = 90°$ than when $\theta_{pq} = 0°$.

All of the 2D IR examples presented above were theoretical spectra generated from a 2D IR simulation code written by our group. If the peaks are well-separated and highly anharmonic like

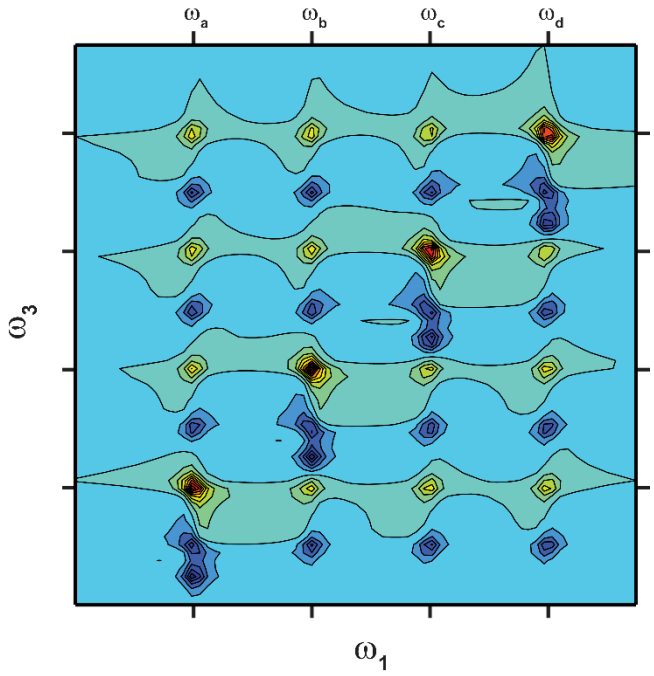


Figure 1.9: Simulated 2D IR spectrum (code in Appendix A) for a system of four coupled vibrations.

the examples here, all of this information can be read directly from experimental spectra as well. But when modes become overlapped, or they are only slightly anharmonic, or there are a larger number of modes, it becomes difficult to extract useful information from the 2D IR spectra. When this is the case, it becomes necessary to fit the spectra, which is the reason we developed the simulation code in the first place. In the case of FeRu and FePtFe, all three of these issues become a problem. To illustrate, Figure 1.9 shows a simulated spectrum of four coupled modes that are well-separated and highly anharmonic. This alone leads to 16 sets of peaks and a more confusing spectrum. However, as the FTIR of both FeRu and FePtFe show, the four ν_{CN} modes are not well separated, and as shown later, they are not highly anharmonic. This is the subject of Chapter 2.

1.4 Dissertation Outline

Chapter 2 looks at gaining a better understanding of the vibrational coupling between four cyanide stretching modes in the ground electronic state of FeRu dissolved in D_2O or FA and FePtFe

dissolved in D_2O using 2D IR spectroscopy. In order to obtain these couplings and other molecular properties, it was necessary to fit the 2D IR spectra. Chapter 3 also looks more closely at the ground electronic state of FeRu and FePtFe. Here, the focus is on using 2D IR and IR pump-probe spectroscopy to study vibrational relaxation dynamics within each molecule. Finally in Chapter 4, we move to new cyanide containing systems. This chapter focuses on developing syntheses and characterizing the vibrational spectra of three ruthenium (II) polypyridine cyanide systems for use in 2D IR experiments to characterize the cyanide stretching modes on the ground electronic state before using visible pump – IR probe experiments to study what role the high-frequency CN stretching modes play in a MLCT reaction.

REFERENCES

[1] Allen, G. C.; Hush, N. S., *Prog. Inorg. Chem.* **1967**; Vol. 8.

[2] Hush, N. S., *Prog. Inorg. Chem.* **1967**; Vol. 8.

[3] Robin, M. B. *Inorg. Chem.* **1962**, *1*, 337.

[4] Weidinger, D.; Brown, D. J.; Owrutsky, J. C. *J. Chem. Phys.* **2011**, *134*, 124510.

[5] Creutz, C.; Taube, H. *J. Am. Chem. Soc.* **1969**, *91*, 3988.

[6] Lu, H.; Petrov, V.; Hupp, J. T. *Chem. Phys. Lett.* **1995**, *235*, 521.

[7] Bernhardt, P.; Bozoglian, F.; Macpherson, B.; Martinez, M. *Coord. Chem. Rev.* **2005**, *249*, 1902.

[8] Ohba, M.; Ōkawa, H. *Coord. Chem. Rev.* **2000**, *198*, 313.

[9] Shatruk, M.; Avendano, C.; Dunbar, K. R., *Prog. Inorg. Chem.* **2009**; Vol. 56.

[10] Vogler, A.; Osman, A. H.; Kunkely, H. *Inorg. Chem.* **1987**, *26*, 2337.

[11] Zhou, M.; Pfennig, B. W.; Steiger, J.; Van Engen, D.; Bocarsly, A. B. *Inorg. Chem.* **1990**, *29*, 2456.

[12] Barbara, P. F.; Meyer, T. J.; Ratner, M. A. *J. Phys. Chem* **1996**, *100*, 13148.

[13] Hush, N. S. *Electrochim. Acta* **1968**, *13*, 1005.

[14] Son, D. H.; Kambhampati, P.; Kee, T. W.; Barbara, P. F. *J. Phys. Chem. A* **2002**, *106*, 4591.

[15] Reid, P. J.; Silva, C.; Barbara, P. F.; Karki, L.; Hupp, J. T. *J. Phys. Chem.* **1995**, *99*, 2609.

[16] Tominaga, K.; Kliner, D. A. V.; Johnson, A. E.; Levinger, N. E.; Barbara, P. F. *J. Chem. Phys.* **1993**, *98*, 1228.

[17] Walker, G. C.; Aakesson, E.; Johnson, A. E.; Levinger, N. E.; Barbara, P. F. *J. Phys. Chem.* **1992**, *96*, 3728.

[18] Kliner, D. A. V.; Tominaga, K.; Walker, G. C.; Barbara, P. F. *J. Am. Chem. Soc.* **1992**, *114*, 8323.

[19] Arnett, D. C.; Voehringer, P.; Scherer, N. F. *J. Am. Chem. Soc.* **1995**, *117*, 12262.

[20] Wang, C.; Mohney, B. K.; Akhremitchev, B. B.; Walker, G. C. *J. Phys. Chem. A* **2000**, *104*, 4314.

[21] Wang, C.; Mohney, B. K.; Williams, R. D.; Petrov, V.; Hupp, J. T.; Walker, G. C. *J. Am. Chem. Soc.* **1998**, *120*, 5848.

[22] Tivanski, A. V.; Wang, C.; Walker, G. C. *J. Phys. Chem. A* **2003**, *107*, 9051.

[23] Doorn, S. K.; Hupp, J. T. *J. Am. Chem. Soc.* **1989**, *111*, 1142.

[24] Doorn, S. K.; Stoutland, P. O.; Dyer, R. B.; Woodruff, W. H. *J. Am. Chem. Soc.* **1992**, *114*, 3133.

[25] Doorn, S. K.; Dyer, R. B.; Stoutland, P. O.; Woodruff, W. H. *J. Am. Chem. Soc.* **1993**, *115*.

[26] Watson, D. F.; Tan, H. S.; Schreiber, E.; Mordas, C. J.; Bocarsly, A. B. *J. Phys. Chem. A* **2004**, *108*, 3261.

[27] Lynch, M. S.; Slenkamp, K. M.; Khalil, M. *J. Chem. Phys.* **2012**, *136*, 241101.

[28] Lynch, M. S.; Van Kuiken, B. E.; Daifuku, S. L.; Khalil, M. *J. Phys. Chem. Lett.* **2011**, *2*, 2252.

[29] Macatangay, A. V.; Mazzetto, S. E.; Endicott, J. F. *Inorg. Chem.* **1999**, *38*, 5091.

[30] Watzky, M. A.; Endicott, J. F.; Song, X.; Lei, Y.; Macatangay, A. *Inorg. Chem.* **1996**, *35*, 3463.

[31] Watzky, M. A.; Macatangay, A. V.; Van Camp, R. A.; Mazzetto, S. E.; Song, X.; Endicott, J. F.; Buranda, T. *J. Phys. Chem. A* **1997**, *101*, 8441.

[32] Bixon, M.; Jortner, J. *Faraday Discuss.* **1982**, *74*, 17.

[33] Newton, M. D.; Sutin, N. *Annu. Rev. Phys. Chem.* **1984**, *35*, 437.

[34] Marcus, R. A.; Sutin, N. *Biochim. Biophys. Acta* **1985**, *811*, 265.

[35] Jortner, J.; Bixon, M. *J. Chem. Phys.* **1988**, *88*, 167.

[36] Barbara, P. F.; Walker, G. C.; Smith, T. P. *Science* **1992**, *256*, 975.

[37] Walker, G. C.; Barbara, P. F.; Doorn, S. K.; Dong, Y.; Hupp, J. T. *J. Phys. Chem.* **1991**, *95*, 5712.

[38] Doorn, S. K.; Hupp, J. T. *J. Am. Chem. Soc.* **1989**, *111*, 1942.

[39] Hupp, J. T.; Williams, R. D. *Acc. Chem. Res.* **2001**, *34*, 808.

[40] Vance, F. W.; Karki, L.; Reigle, J. K.; Hupp, J. T.; Ratner, M. A. *J. Phys. Chem. A* **1998**, *102*, 8320.

[41] Pfennig, B. W.; Bocarsly, A. B. *J. Phys. Chem.* **1992**, *96*, 226.

[42] Pfennig, B. W.; Wu, Y.; Kumble, R.; Spiro, T. G.; Bocarsly, A. B. *J. Phys. Chem.* **1996**, *100*, 5745.

[43] Vance, F. W.; Slone, R. V.; Stern, C. L.; Hupp, J. T. *Chem. Phys.* **2000**, *253*, 313.

[44] Todd, M. D.; Nitzan, A.; Ratner, M. A.; Hupp, J. T. *J. Photochem. Photobiol., A* **1994**, *82*, 87.

[45] Bublitz, G. U.; Laidlaw, W. M.; Denning, R. G.; Boxer, S. G. *J. Am. Chem. Soc.* **1998**, *120*, 6068.

[46] Karki, L.; Lu, H. P.; Hupp, J. T. *J. Phys. Chem.* **1996**, *100*, 15637.

[47] Karki, L.; Hupp, J. T. *J. Am. Chem. Soc.* **1997**, *119*, 4070.

[48] Wu, Y.; Pfennig, B. W.; Sharp, S. L.; Ludwig, D. R.; Warren, C. J.; Vicenzi, E. P.; Bocarsly, A. B. *Coord. Chem. Rev.* **1997**, *159*, 245.

[49] Watson, D. F.; Bocarsly, A. B. *Coord. Chem. Rev.* **2001**, *211*, 177.

[50] Pfennig, B. W.; Mordas, C. J.; McCloskey, A.; Lockard, J. V.; Salmon, P. M.; Cohen, J. L.; Watson, D. F.; Bocarsly, A. B. *Inorg. Chem.* **2002**, *41*, 4389.

[51] Slenkamp, K. M.; Lynch, M. S.; Van Kuiken, B. E.; Brookes, J. F.; Bannan, C. C.; Daifuku, S. L.; Khalil, M. *J. Chem. Phys.* **2014**, *140*, 084505.

[52] Van Kuiken, B. E.; Valiev, M.; Daifuku, S. L.; Bannan, C.; Strader, M. L.; Cho, H.; Huse, N.; Schoenlein, R. W.; Govind, N.; Khalil, M. *J. Phys. Chem. A* **2013**.

[53] Lynch, M. S.; Slenkamp, K. M.; Cheng, M.; Khalil, M. *J. Phys. Chem. A* **2012**, *116*, 7023.

Chapter 2:

Investigating Vibrational Anharmonic Couplings In Cyanide-Bridged Transition Metal Mixed Valence Complexes Using Two-Dimensional Infrared Spectroscopy

The work in this chapter was published in the following article:
Slenkamp, K. M.; Lynch, M. S.; Van Kuiken, B. E.; Brookes, J. F.; Bannan, C. C.; Daifuku, S. L.; and Khalil, M. "Investigating Vibrational Anharmonic Couplings in Cyanide-Bridged Transition Metal Mixed Valence Complexes Using Two-Dimensional Infrared Spectroscopy." *Journal of Chemical Physics* **2014**, *140*, 084505.

Using polarization-selective two-dimensional infrared (2D IR) spectroscopy, we measure anharmonic couplings and angles between the transition dipole moments of the four cyanide stretching (ν_{CN}) vibrations found in $[(NH_3)_5Ru^{III}NCFe^{II}(CN)_5]^-$ (FeRu) dissolved in D_2O and formamide and $[(NC)_5Fe^{II}CNPt^{IV}(NH_3)_4NCFe^{II}(CN)_5]^{4-}$ (FePtFe) dissolved in D_2O. These cyanide-bridged transition metal complexes serve as model systems for studying the role of high frequency vibrational modes in ultrafast photoinduced charge transfer reactions. Here, we focus on the spectroscopy of the ν_{CN} modes in the electronic ground state. The FTIR spectra of the ν_{CN} modes of the bimetallic and trimetallic systems are strikingly different in terms of frequencies, amplitudes, and lineshapes. The experimental 2D IR spectra of FeRu and FePtFe and their fits reveal a set of weakly coupled anharmonic ν_{CN} modes. The vibrational mode anharmonicities of the individual ν_{CN} modes range from 14 to 28 cm^{-1}. The mixed-mode anharmonicities range from 2 to 14 cm^{-1}. In general, the bridging ν_{CN} mode is most weakly coupled to the radial ν_{CN} mode, which involves the terminal CN ligands. Measurement of the relative transition dipole moments

of the four ν_{CN} modes reveal that the FeRu molecule is almost linear in solution when dissolved in formamide, but it assumes a bent geometry when dissolved in D_2O. The ν_{CN} modes are modelled as bilinearly coupled anharmonic oscillators with an average coupling constant of 6 cm^{-1}. This study elucidates the role of the solvent in modulating the molecular geometry and the anharmonic vibrational couplings between the ν_{CN} modes in cyanide-bridged transition metal mixed valence complexes.

2.1 INTRODUCTION

Cyanide-bridged transition metal mixed valence complexes are a class of molecules which exhibit unique redox, spectroscopic, and charge transfer properties by virtue of bearing oxidizing and reducing moieties. They are widely studied for their applications in magnetism and photochemical energy conversion.[1-3] Additionally, these complexes have served as model systems to probe the coupling between electronic and vibrational motions during ultrafast photoinduced electron transfer reactions using equilibrium and transient spectroscopic techniques.[4-24] The goal of this work is to understand how the solvent modulates the molecular geometry and the anharmonic couplings between the bridging and terminal cyanide stretching vibrations in cyanide-bridged bimetallic and trimetallic transition metal mixed valence complexes. In this study, we use polarization-selective two-dimensional infrared (2D IR) spectroscopy to measure the anharmonic couplings and the angles between the transition dipole moments in the ground electronic state of the cyanide stretching (ν_{CN}) vibrations of $[(NH_3)_5Ru^{III}NCFe^{II}(CN)_5]^-$ (FeRu) dissolved in D_2O and formamide (FA) and $[(NC)_5Fe^{II}CNPt^{IV}(NH_3)_4NCFe^{II}(CN)_5]^{4-}$ (FePtFe) dissolved in D_2O.

The molecular structures of the complexes and a simplified depiction of the ν_{CN} modes (ν_{trans}, ν_{radial}, ν_{axial}, and ν_{bridge}) are shown in Figure 2.1. A detailed description of the modes for FePtFe is provided in Figure 2.2 below. Our motivation to study the complexes shown in Figure 2.1 is based on previous IR and Raman spectroscopic studies of these molecules which have implicated some of the high frequency CN stretching vibrations in the ultrafast charge transfer processes. Transient IR studies of the FeRu molecule and its analogues find a high degree of vibrational excitation in the ν_{CN} modes upon ultrafast back electron transfer (BET).[14,15,18]. The role of the ν_{CN} modes in the metal-to-metal charge transfer (MMCT) excitation, and subsequent BET, of FePtFe dissolved in D_2O has also been recently studied using various nonlinear spectroscopies. [20,21,25] These studies have found greater than 6 quanta of vibrational energy deposited into the ν_b mode following BET and provided experimental evidence that ν_{CN} modes are both coherently and incoherently coupled to the charge transfer reaction. Resonance Raman data on FePtFe and time-dependent intensity analysis by several research groups have shown that >40% of the inner sphere reorganization

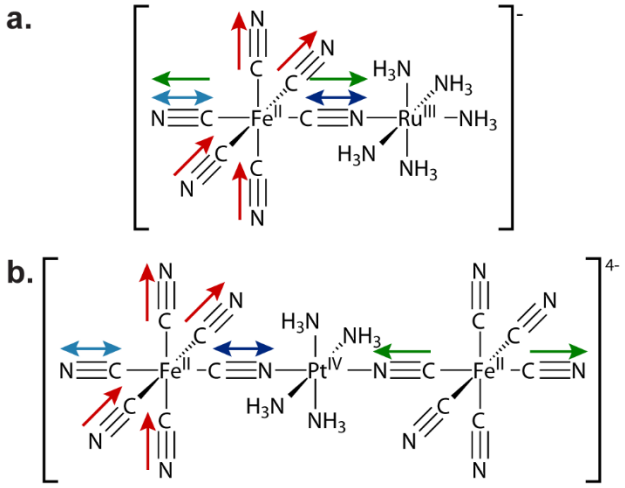


Figure 2.1: The molecular structures of (a) FeRu and (b) FePtFe. Colored arrows correspond to simplified depictions of four ν_{CN} modes. The transition dipole moments of the ν_{trans} (ν_t, blue), ν_{axial} (ν_a, green), and ν_{bridge} (ν_b, purple) modes lie along the MMCT axis and are perpendicular to the degenerate ν_{radial} (ν_r, red) mode (see Figure 2.2). The metal-metal bond distance for FePtFe [31] is 4.99 Å and ranges from 5.0 and 5.2 Å for FeRu. [9,14,32,33]

energy involves vibrational modes which lie along the intervalence charge transfer axis of the complex.[19,26-30]. In this study, we aim to measure the anharmonic couplings and angles between the transition dipole moment of the ν_{CN} modes lying parallel and perpendicular to the charge transfer axis in the electronic ground state of the complexes shown in Figure 2.1. Our results will help illuminate how the molecular structure and geometry of cyanide-bridged transition metal mixed valence complexes affects ultrafast charge and energy transfer pathways following photoexcitation.

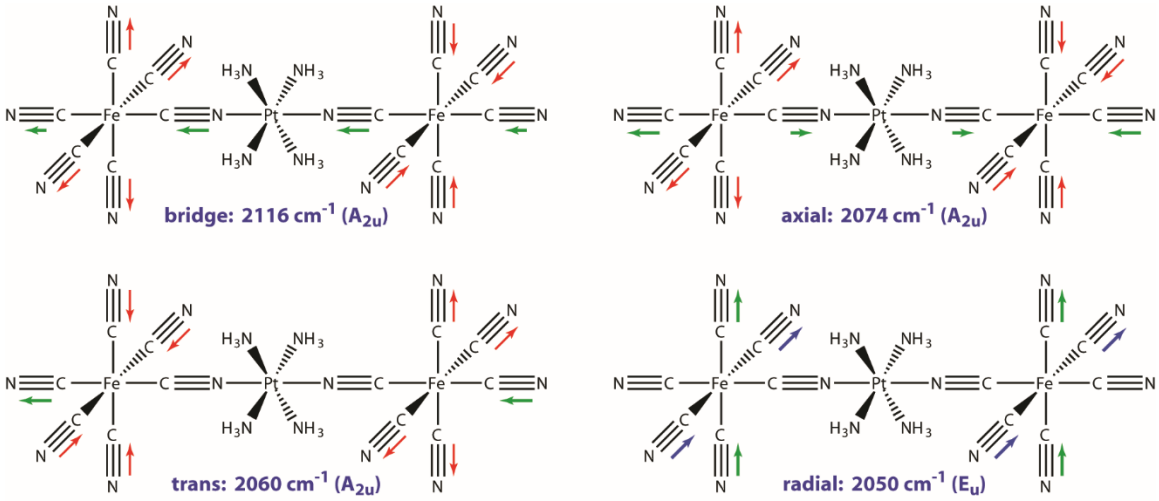


Figure 2.2: Cartoons representing the normal modes for the ν_{CN} vibrations ν_{trans}, ν_{radial}, ν_{axial}, and ν_{bridge}. Normal mode symmetries are based on the D_{4h} point group of the molecule. Red arrows represent symmetric displacements (not to scale). Similar modes are observed for the dimer RuFe.

The role of the solvent in modulating the vibronic coupling in transition metal complexes has been studied for the trans ν_{CN} mode in FeRu (see Figure 2.1a).[13] Recent X-ray absorption spectroscopy results at the Ru L_3-edge on FeRu have also shown the importance of hydrogen bonding effects of the solvent in modulating the charge on the transition metals.[34] We will use 2D IR spectroscopy to measure how the solvent modifies the molecular geometry and the vibrational couplings of the ν_{CN} modes in FeRu and relate our results to previous studies. The experiment will measure the angles between the ν_{CN} modes and their vibrational mode and mixed-mode

anharmonicities by fitting simulated 2D IR spectra to the experimental spectra of FeRu and FePtFe. Using these normal mode anharmonicities, we also calculate bilinear coupling constants using the local mode basis for all three samples.

The remainder of the Chapter is organized as follows. Section 2.2 describes the methods used to synthesize the molecules, collect the 2D IR spectra, and simulate and fit the 2D IR spectra. Section 2.3 is split into three subsections. Section 2.3.1 describes the FTIR fitting and results. Section 2.3.2 contains the results and discussion of the experimental and simulated 2D IR spectra including the values for the angles and anharmonicities for FeRu in FA, FeRu in D_2O, and FePtFe in D_2O. Section 2.3.3 describes the method used to determine bilinear coupling constants and includes the discussion of these coupling constants. The final section, Section 2.4, contains the summary of the results and our conclusions.

2.2 METHODS

Materials. Starting materials and solvents were purchased from Sigma Aldrich and used without further purification. Steady-state infrared spectra were collected with a JASCO FT/IR-4100 with a spectral resolution of 0.5 cm^{-1}. The solvents used in this study have absorption bands in the ν_{CN} region as shown in Figure 2.3. In order to remove the effect of the solvent before fitting the spectra shown in Figure 2.5, the solvent background was subtracted. For the case of FeRu dissolved in D_2O, an additional Voigt lineshape centered at 1979 cm^{-1} with a width of 64 cm^{-1} was subtracted from the FTIR spectrum to remove all contributions from the solvent. This subtraction has no effect on the fitting results tabulated in Table 2.1.

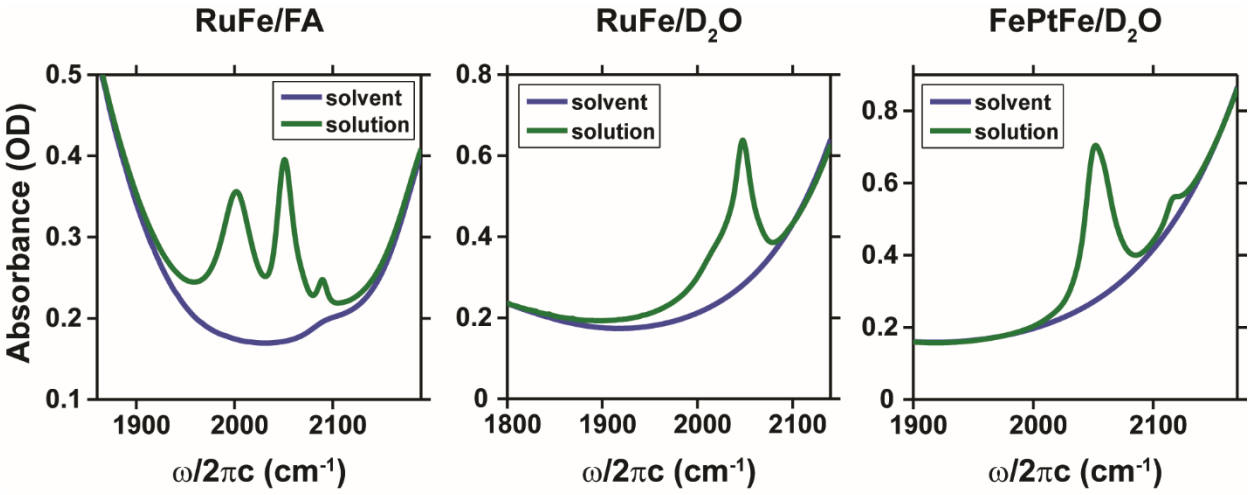


Figure 2.3: FTIR spectra before being solvent subtracted. The final experimental spectra presented in this paper in Figure 2.5 are the difference between the solution spectra (green) and the solvent spectra (blue).

The mixed valence dimer, FeRu, was synthesized according to literature methods with a modified purification procedure.[35] Briefly, an aqueous solution consisting of ~0.01 M $K_4[Fe(CN)_6]$ and ~0.01 M $[RuCl(NH_3)_5]Cl_2$ was heated at 60 °C for two hours. After removing solid impurities using vacuum filtration, methanol was added to the solution in a 2:1 volume ratio, and the solution was cooled overnight. The solid, green dimer was filtered out of solution, run through an ion-exchange column packed with Dowex, and then run through a size-exclusion column prepared with Bio-Gel P2 to remove starting materials. A saturated solution of FeRu was prepared in formamide (FA) with a maximum OD of 0.22 in the ν_{CN} region after solvent subtraction. A solution of FeRu was also prepared in deuterium oxide (D_2O) to a concentration of ~23 mM and maximum OD of 0.35 in the ν_{CN} region.

The mixed valence trimer, FePtFe, was synthesized following literature methods with additional purification steps.[19] Briefly, aqueous $K_3[Fe(CN)_6]$ and $[Pt(NH_3)_4](NO_3)_2$ were mixed at room temperature, and the resulting solution was filtered using gravity filtration and purified by ion-exchange and size-exclusion chromatography. A solution of FePtFe dissolved in D_2O was prepared to a concentration of ~13 mM and a maximum OD of 0.41 for use in experiments.

2D IR Experiments. The experimental set-up for performing 2D IR spectroscopy has been described recently.[25,36] The polarization-selective experiments described here are performed with ~80 fs mid-IR pulses centered at a wavelength of 4.9 μm with a spectral bandwidth greater than 250 cm^{-1}. Each arm of the interferometer has a tunable half-wave plate (PO-TWP-L2-25-IR, Alphalas) and a wire-grid polarizer (WP25H-Z, Thorlabs) to adjust beam polarization and input power. The input beams, arranged in a "boxcar" geometry, are focused onto the sample with a spot size ($1/e^2$) of ~150 μm. The pulse energy in each arm of the three input beams is 0.4 μJ/pulse (0.5 μJ/pulse) for the FeRu (FePtFe) samples. The sample was contained in a home built sample cell equipped with two 1 mm thick CaF_2 windows and a 50 μm Teflon spacer for the D_2O solutions or a 100 μm Teflon spacer for the FeRu in FA solution. The background free signal field is overlapped with the local oscillator beam on a 50:50 ZnSe beamsplitter for balanced heterodyne detection as described previously.[25]

2D IR spectra were collected in two different polarization geometries: parallel (ZZZZ) where all input beams and the signal field have the same polarization, and crossed (YYZZ) where input beams 1 and 2 have a polarization that is rotated 90° to the polarization of beam 3 and the signal field. The ratio between peak amplitudes in the ZZZZ and YYZZ polarization geometry contains information about the angles between vibrational dipole moments.[37,38] The IR coherence time delay (τ_1) between pulses 1 and 2 and the waiting time delay (τ_2) between pulses 2 and 3 are computer-controlled via Newport XMS50 linear stages. For FeRu dissolved in FA, the τ_1 delay was scanned from 0 to 3 ps in 3 fs steps for the rephasing (R) spectra and from 0 to 1.8 ps in 3 fs steps for the non-rephasing (NR) spectra at fixed τ_2 = 140 fs. For FeRu dissolved in D_2O, the τ_1 delay was scanned from 0 to 2.5 ps in 3 fs steps for the R spectra and from 0 to 1.5 ps in 3 fs steps for the NR spectra at fixed τ_2 = 150 fs. For FePtFe, τ_1 was scanned from 0 to 2.2 ps in 4 fs steps

for the R spectra and from 0 to 1.5 ps in 4 fs steps for the NR spectra at fixed τ_2 = 150 fs. The 2D IR data is collected at $\tau_2 \neq 0$ fs to reduce the contamination of the desired signal from additional signals present during the pulse-width of 80 fs and non-resonant signals from the sample cell windows. The data was zero padded to 2^{13} steps along τ_1 and then Fourier transformed to obtain the ω_1 axis. Rephasing and non-rephasing spectra were phased using a dispersed pump-probe trace at the same τ_2 and added to construct absorptive 2D IR spectra for each sample as described previously.[39] Each data point represents 1500 shots for the FeRu samples and 2000 shots for the FePtFe sample, and each spectrum was collected twice and averaged to obtain the experimental spectra presented here. Experimental spectral resolution along the ω_1 and ω_3 axes are 1.4 and ~2.8 cm^{-1} respectively for the FeRu samples and 1.4 and ~2.9 cm^{-1} for the FePtFe sample.

2D IR Spectral Fitting. One can simulate a 2D IR spectrum starting from the third order nonlinear signal that can be calculated using the nonlinear response formalism developed previously.[39-41] The signal is calculated in the time domain as a function of τ_1 and τ_3 for a specific τ_2 and Fourier transformed in both dimensions to obtain the ω_1 and ω_3 axes. Rephasing and non-rephasing spectra are calculated separately and then added in the frequency domain to obtain the absorptive spectrum. We assume a separation of time scales for vibrational and orientational dynamics when calculating the third order response functions. The rotational correlation function was written assuming a symmetric diffuser with an orientational diffusion constant of D_{or}.[42]

The frequency-frequency correlation functions (FFCFs) used to simulate the 2D spectra are of the form:

$$C_{pp}(t) = A_{pp}^2 \exp\left(\frac{-t}{\tau_{pp}}\right) + \frac{\delta(t)}{T_2^*} \tag{2.1}$$

$$C_{pq}(t) = A_{pp} A_{qq} \exp\left(\frac{-t}{\tau_{pq}}\right) + \frac{\delta(t)}{T_2^*} \tag{2.2}$$

Equation 2.1 is an autocorrelation function for mode p and Equation 2.2 is a cross correlation function between modes p and q. In the above expressions, A_{pp} and τ_{pp} are the amplitude and time scale of the Kubo model for mode p, τ_{pq} is the cross correlation time scale for modes p and q, $\delta(t)$ is the Dirac delta function, and T_2^* is the time scale that represents the fast dynamics in a system leading to the homogeneous linewidth of peaks. All other FFCFs can be approximated as combinations of the two equations shown above.[41] The following harmonic scaling relationships are used to limit the parameters necessary to simulate theoretical spectra:

$$\begin{aligned} &C_{2p,2p}(t) = 2C_{2p,p}(t) = 2C_{p,2p}(t) = 4C_{pp}(t); \\ &C_{p,pq}(t) = C_{pp}(t) + C_{pq}(t); C_{q,pq}(t) = C_{qq}(t) + C_{pq}(t); \\ &C_{pq,pq}(t) = C_{pp}(t) + C_{qq}(t) + 2C_{pq}(t) \end{aligned} \tag{2.3}$$

These relationships reduce the number of amplitude and timescale parameters needed for a system of four coupled vibrations from 60 to 14. We assume harmonic scaling for the magnitude of the vibrational transition dipole moment such that $|\mu_{12}| = \sqrt{2}|\mu_{10}|$ and that the transition dipole moment for the n = 1 → n = 2 transition is parallel to the transition dipole moment for the n = 0 → n = 1 transition where n is the vibrational energy level of the system.

To determine orientational response functions each transition dipole moment is projected onto Cartesian coordinates with the transition dipole moment of the ν_b mode acting as the arbitrary z-axis, the transition dipole moment of the ν_r mode confined to the xz-plane at an angle of θ_{rb} from the ν_b mode's transition dipole moment, and the other two transition dipole moments determined in relation to the ν_b and ν_r modes by the angles between each transition dipole moment. We calculate the contribution of each affected transition dipole moment to the orientational response function for each possible set of light-matter interactions using the following equations from Reference 37:

$$
\begin{aligned}
Y_{ZZZZ}^{aaaa} &= \frac{1}{9}c_1(\tau_1)c_1(\tau_3)\left(1+\frac{4}{5}c_2(\tau_2)\right),\\
Y_{YYZZ}^{aaaa} &= \frac{1}{9}c_1(\tau_1)c_1(\tau_3)\left(1-\frac{2}{5}c_2(\tau_2)\right),\\
Y_{ZZZZ}^{aabb} &= \frac{1}{9}c_1(\tau_1)c_1(\tau_3)\left(1-\frac{2}{5}c_2(\tau_2)\right),\\
Y_{YYZZ}^{aabb} &= \frac{1}{9}c_1(\tau_1)c_1(\tau_3)\left(1+\frac{1}{5}c_2(\tau_2)\right),\\
Y_{ZZZZ}^{abab} &= Y_{ZZZZ}^{abba} = \frac{1}{15}c_1(\tau_1)c_1(\tau_3)c_2(\tau_2),\\
Y_{YYZZ}^{abab} &= Y_{YYZZ}^{abba} = -\frac{1}{30}c_1(\tau_1)c_1(\tau_3)c_2(\tau_2),\\
&\text{where } c_J(\tau_2) = \exp(-J(J+1)D_{or}\tau_n)
\end{aligned}
\tag{2.4}
$$

Each Y from Equation 2.4 represents the tensor element of the orientational response function for the case of four field interactions. Here *a* represents transition dipole moments that are parallel to the first transition dipole excited, and *b* represents transition dipole moments that are orthogonal to *a*.

The goal of simulating the 2D IR spectra is to extract vibrational mode and mixed-mode anharmonicities and the angles between the transition dipole moments of the ν_{CN} modes for each molecule. For all three complexes, a 15 energy-level system is used consisting of a common ground state, four first-excited one-quantum states, six combination two-quantum states, and four second-excited two-quantum states. A total of 120 third order response functions have to be calculated for each absorptive spectrum. To obtain a best fit spectrum, we iteratively change the lineshape parameters, the position of the two-quantum energy levels, and the transition dipole moment directions. We also fit an amplitude correction factor to each YYZZ spectrum to account for any experimental differences between the input power of beams 1 and 2 during the collection of the YYZZ and the ZZZZ spectra. The differences in input power arise from polarization-selective responses of some of the optics in the beam path. We stress that since we obtain the

angles by fitting both the ZZZZ and YYZZ data simultaneously for the four ν_{CN} modes, the correction factor does not affect our conclusions. The lineshape parameters are only fit to get the correct amplitude for each peak to enable an accurate determination of the transition dipole moment directions and the angles between them. These lineshape parameters were used to simulate the FTIR spectra displayed in Figure 2.4. The final lineshape parameters do not necessarily reflect the actual FFCF amplitudes and timescales for each sample and therefore will not be discussed in this chapter. The fits were completed in MATLAB using a nonlinear least-squares fitting routine.

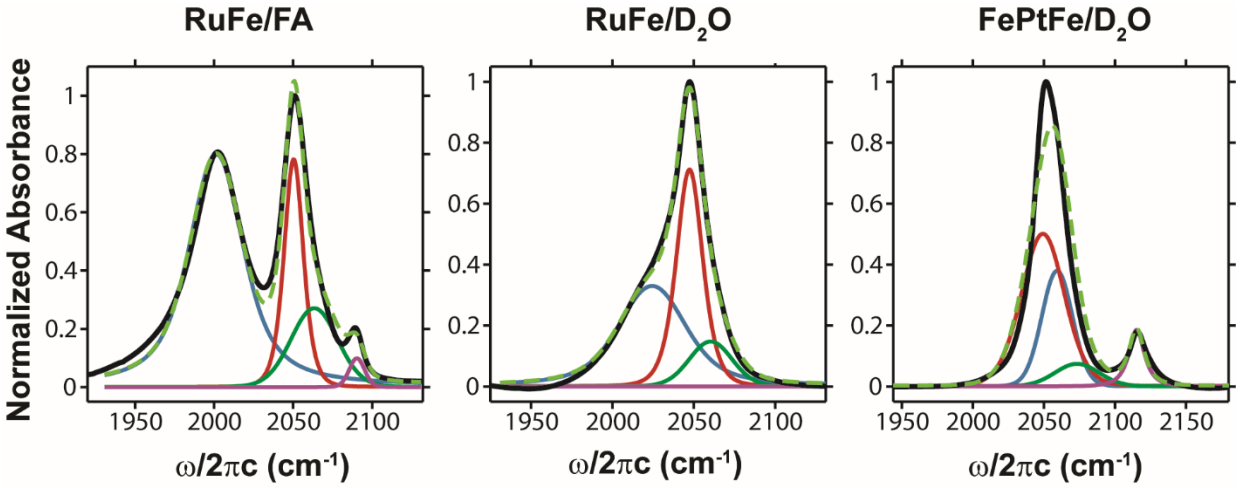


Figure 2.4: Experimental FTIR spectra (solid black) and spectra simulated (dashed green) using the lineshape parameters from our 2D fits presented in the captions for Figures 2.7, 2.8, and 2.10. The contribution to the simulated spectra from each mode is also shown: ν_t (blue), ν_r (red), ν_a (green), and ν_b (purple).

2.3 Results and Discussion

2.3.1 *FTIR Spectra and Assignment of the ν_{cn} Modes*

Figure 2.5 displays the solvent-subtracted FTIR spectra of the ν_{CN} region for FeRu dissolved in FA and D_2O and FePtFe dissolved in D_2O. Each FTIR spectrum was fit to four Voigt lineshapes for each ν_{CN} mode shown in Figure 2.1. Table 2.1 lists the center frequencies and spectral

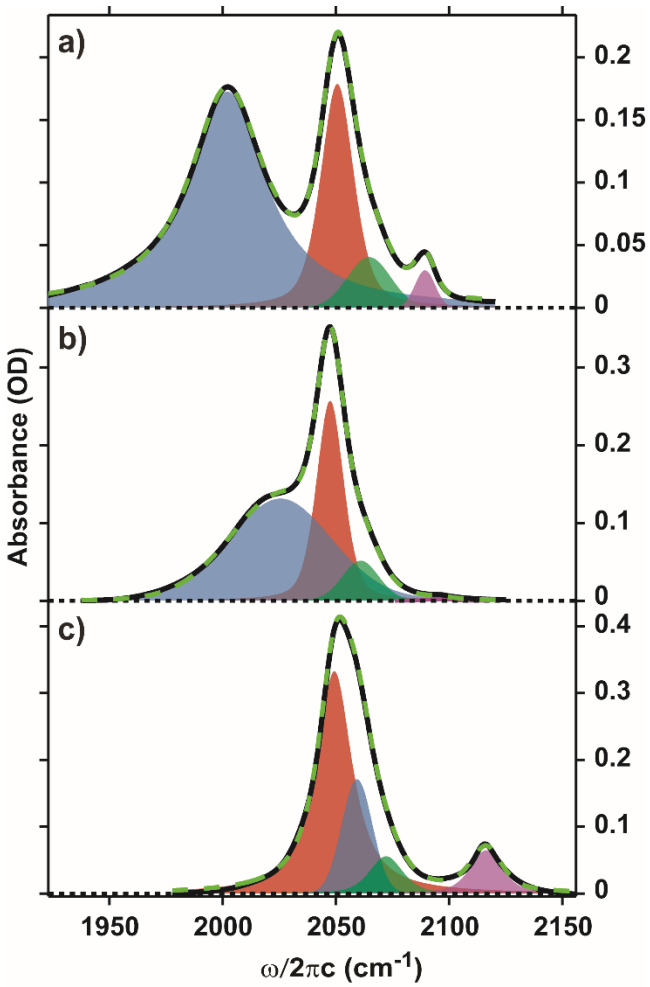


Figure 2.5: Solvent subtracted FTIR spectra (black line) of (a) FeRu in FA (b) FeRu in D_2O and (c) FePtFe in D_2O. Each FTIR spectrum was fit to four Voigt lineshapes, and each peak is color coded to match the four modes shown in Figure 2.1 (ν_t (blue), ν_a (green), ν_b (purple), and ν_r (red)). Parameters of each fit are listed in Table 2.1. The final total fit is shown in dashed green.

linewidths of those fits for each sample. The fits to the spectra reveal significant changes in positions for the four ν_{CN} modes between FeRu and FePtFe and as a function of solvent for FeRu. In addition to shifts in frequency, there is variation in the linewidths ($\Delta\nu_{FWHM}$) and transition dipole strengths ($\left|\mu_p\right|^2$) in the FTIR spectra of the four ν_{CN} modes shown in Figure 2.5. The linewidth of the ν_t mode is significantly larger in FeRu than in FePtFe. By contrast, the linewidths of the other three modes change by less than 3 cm^{-1} between all the samples. The relative transition dipole

strengths also exhibit a variation among the three samples. For FeRu in FA, the transition dipole moment strengths of all the ν_{CN} modes with respect to the ν_t mode are as follows: $|\mu_r| = 0.56|\mu_t|$, $|\mu_a| = 0.30|\mu_t|$, and $|\mu_b| = 0.22|\mu_t|$. When FeRu is dissolved in D_2O, the $|\mu_p|^2$ of all the ν_{CN} modes with respect to the ν_t mode are as follows: $|\mu_r| = 0.89|\mu_t|$, $|\mu_a| = 0.40|\mu_t|$, and $|\mu_b| = 0.12|\mu_t|$. For FePtFe in D_2O, the strongest peak is assigned to the radial mode and $|\mu_p|^2$ of all the ν_{CN} modes with respective to the ν_r mode are as follows: $|\mu_t| = 0.73|\mu_r|$, $|\mu_a| = 0.40|\mu_r|$, and $|\mu_b| = 0.42|\mu_r|$.

It is instructive to compare the FTIR spectra shown in Figure 2.5 with previously reported FTIR and Raman spectra of FeRu and FePtFe. Previous work by Walker and co-workers had assigned three high-frequency IR active ν_{CN} modes for FeRu, two of which are also Raman active. The IR active modes were labelled as $\nu_{brigde\text{-}CN}$, $\nu_{cis\text{-}CN}$, and $\nu_{trans\text{-}CN}$ and they map to the modes labelled ν_b, ν_r, and ν_t, respectively in Figure 2.1.[14] The previous study did not assign an axial mode in FeRu, but a fourth ν_{CN} mode was acknowledged in the FTIR spectrum of $[(NH_3)_5Ru^{III}NCOs^{II}(CN)_5]^-$ at approximately 2070 cm^{-1}.[15] We see clear evidence of the ν_a mode in the form of cross-peaks in 2D IR spectra (Figures 2.7-2.10) and in dispersed IR pump-probe data of the FeRu samples. We have measured the Raman spectrum of FeRu in FA showing that the ν_t mode is centered at 2000±2 cm^{-1}, and the ν_b mode is centered at 2088±2 cm^{-1}, similar to the IR spectrum. For FeRu in D_2O, the ν_t mode in the Raman spectrum is centered at 2022±4 cm^{-1}, and the ν_b mode is centered at 2099±2 cm^{-1}. Four high-frequency IR and Raman active modes are also observed for FePtFe. The previously reported Raman spectrum of FePtFe assigns the bridge mode at 2125 cm^{-1}, peaks at 2067 cm^{-1} and 2081 cm^{-1} are assigned to the ν_r mode, and the ν_a mode is observed at 2107 cm^{-1}.[27] The discrepancy between the Raman and IR frequencies of the observed cyanide stretching modes in FePtFe is attributed to the structural complexity of the trimer molecule.

Table 2.1: Best fit parameters of the FTIR spectra of FeRu and FePtFe. The error bars are reported at 95% confidence interval.

(cm^{-1})	ν_{trans}		ν_{radial}		ν_{axial}		ν_{bridge}	
	ν_{max}	$\Delta\nu_{FWHM}$	ν_{max}	$\Delta\nu_{FWHM}$	ν_{max}	$\Delta\nu_{FWHM}$	ν_{max}	$\Delta\nu_{FWHM}$
FeRu/FA	2002.0±0.1	20.5±0.1	2050.6±0.1	8.7±0.1	2065±1	11.2±0.9	2089.4±0.1	5.3±0.2
FeRu/D_2O	2026±1	26.8±0.8	2047.5±0.1	7.1±0.1	2060.9±0.6	9.5±0.6	2096±2	6±3
FePtFe/D_2O	2059.3±0.8	8±3	2049.3±0.5	9.4±0.5	2072±5	9±1	2116.3±0.2	8.2±0.3

The FTIR spectra in Figure 2.5 and the best fit results in Table 2.1 reveal that the largest variation in the spectral parameters (frequency, linewidth, and amplitude) occur for the ν_{CN} modes that lie along the charge transfer axis. These include the trans, axial, and bridge ν_{CN} modes. Our results are consistent with the Raman studies and transient IR studies which have found that the trans and the bridge ν_{CN} modes are coupled to the MMCT transition in these transition metal mixed valence complexes.[13-15,18,20,21] The 2D IR spectra of the ν_{CN} modes described in the next section will measure the anharmonic coupling between these modes and the relative angles between them.

2.3.2 *2D IR Spectra of the ν_{CN} Modes in Mixed Valence Transition Metal Complexes*

Two-dimensional infrared spectroscopy (2D IR) is a third order nonlinear technique that is sensitive to the transitions involving the ground (n = 0), first excited (n = 1), and second excited (n = 2) vibrational states. An example of a generalized fifteen vibrational energy level diagram contributing to a 2D IR spectrum of four coupled ν_{CN} modes is shown in Figure 2.6. Two-dimensional infrared spectra of coupled vibrational modes consist of 'diagonal' and 'cross' peaks. Diagonal peaks, centered at $\omega_1 = \omega_3$, occur when all three pulses interact with the same vibrational mode during τ_1 and τ_3. Cross peaks located at $\omega_1 = \nu_p$ and $\omega_3 = \nu_q$, only occur if modes *p* and *q* are anharmonically coupled and are found within the bandwidth of the IR pulses. There are also corresponding cross peaks between modes *p* and *q* across the diagonal at $\omega_1 = \nu_q$ and $\omega_3 = \nu_p$. Each

diagonal and cross peak consists of positive and negative spectral features. The positive features involve transitions between n = 0 and n = 1 states, while the negative features involve a transition to a two-quantum state. The diagonal peaks are separated along ω_3 by the vibrational mode anharmonicity ($\Delta_{p(q)}$), and the cross peaks are separated along ω_3 by the mixed-mode anharmonicity (Δ_{pq}). The mixed-mode anharmonicities measure the magnitude of the couplings between modes *p* and *q*. When the vibrational anharmonicities are larger than the linewidths of the vibrational modes under study, the positive and negative features are well-separated in the ω_3 dimension and the values of $\Delta_{p(q)}$ and Δ_{pq} can be read off the 2D IR spectrum. This is not the case for the ν_{CN} modes in both FeRu and FePtFe, making it necessary to fit the 2D IR spectrum to obtain accurate values for the anharmonicities.

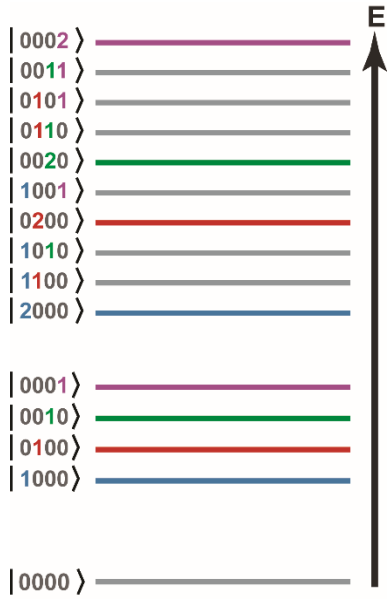


Figure 2.6: A generalized diagram of all the relevant vibrational states probed in the 2D IR experiment for four coupled vibrations. The naming scheme is $|n_t n_r n_a n_b\rangle$ where n_i refers to the vibrational quantum number of the trans, radial, axial, and bridge ν_{CN} modes.

The amplitudes of the diagonal peaks are proportional to $|\mu_p|^4$, and the cross peak amplitudes are proportional to $|\mu_p|^2 |\mu_q|^2$, where μ_p is the magnitude of the transition dipole moment of mode *p*. The amplitudes of the diagonal and cross peaks are also affected by the angle between the transition

dipole moments of the vibrations under study. The ratio of the amplitude of a cross peak in a 2D IR spectrum measured in the ZZZZ (parallel) and the YYZZ (cross) polarization geometries can be related to the relative angle between the transition dipole moments of the coupled vibrations. When the vibrations of interest are spectrally overlapped and the spectral linewidths are larger than the vibrational anharmonicities, it is not possible to obtain an accurate measurement of the angles between coupled vibrations by simply taking a ratio of their amplitudes under different polarization conditions. Attempting to do this for the 2D IR spectra shown in Figures 2.7, 2.8, and 2.10 resulted in imaginary values for several angles making it necessary to fit the spectra to obtain useful angles. We have used the nonlinear response formalism described in the Section 2.2 to fit the polarization selective 2D IR spectra of FeRu in D_2O and FA and FePtFe in D_2O. Table 2.2 lists the best fit parameters for the vibrational anharmonicities and angles between the four coupled ν_{CN} modes. Below we discuss the experimental 2D IR spectra and the best fit results.

Table 2.2: Best fit values for vibrational mode anharmonicities, mixed-mode anharmonicities, and angles between modes for all three samples. The error bars are estimated to be ±2 cm^{-1} and ±5°.

	FeRu/FA	FeRu/D_2O	FePtFe/D_2O
Δ_t (cm^{-1})	14	15	24
Δ_r (cm^{-1})	18	19	21
Δ_a (cm^{-1})	27	28	19
Δ_b (cm^{-1})	19	28	24
Δ_{tr} (cm^{-1})	2	14	2
Δ_{ta} (cm^{-1})	2	2	15
Δ_{tb} (cm^{-1})	4	6	9
Δ_{ra} (cm^{-1})	6	10	9
Δ_{rb} (cm^{-1})	5	9	3
Δ_{ab} (cm^{-1})	13	11	10
θ_{tr}	85°	71°	73°
θ_{ta}	2°	5°	9°
θ_{tb}	6°	14°	25°
θ_{ra}	87°	74°	68°
θ_{rb}	88°	59°	50°
θ_{ab}	6°	19°	18°

2D IR spectroscopy of FeRu in Formamide. The ZZZZ and YYZZ experimental spectra of FeRu in FA at τ_2 = 140 fs are shown in Figures 2.7a and 2.7b. The experimental spectra display three main peaks along the diagonal. The lowest energy diagonal peak involves the ν_t mode, the middle diagonal peak contains overlapping contributions from the ν_r and ν_a modes, and the highest energy peak results from exciting and probing the ν_b mode. The middle diagonal peak also has contributions from the cross peaks between the ν_r and the ν_a modes. There are three well resolved

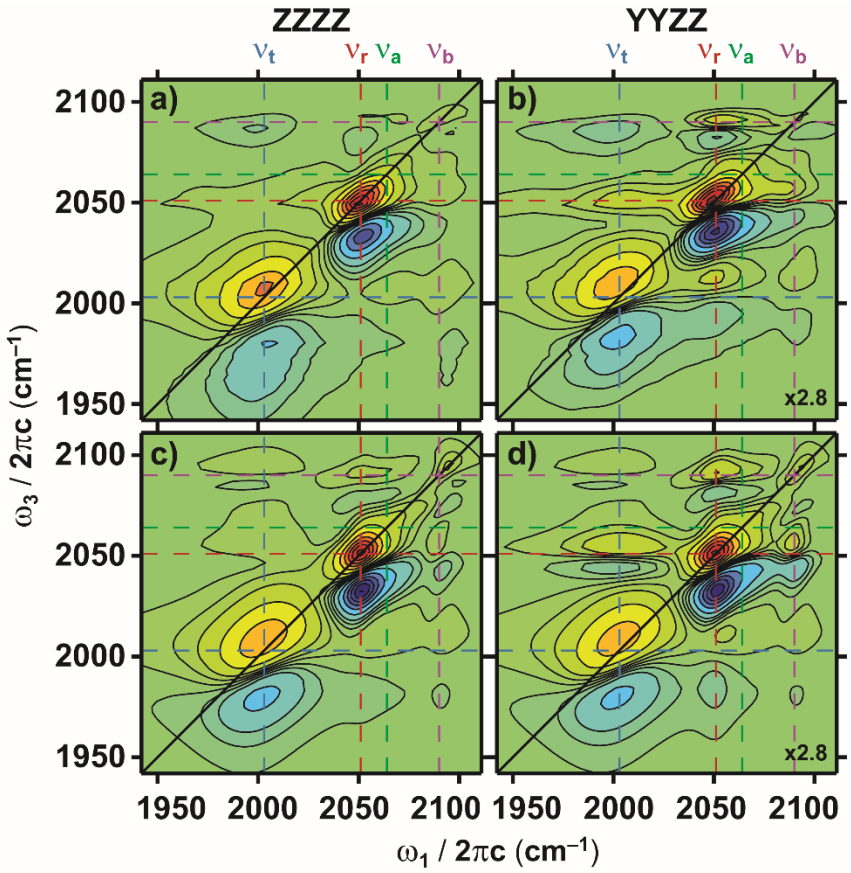


Figure 2.7: 2D IR spectra of FeRu dissolved in FA at τ_2 = 140 fs. (a) ZZZZ experiment, (b) YYZZ experiment, (c) ZZZZ simulated, and (d) YYZZ simulated. Each spectrum is plotted using contour lines at the following positions: ±0.018, ±0.055, ±0.1, ±0.2, ±0.35, ±0.5, ±0.65, ±0.8, and ±0.95. The YYZZ spectra are multiplied by the numbers in the lower right corners. The lineshape parameters used to simulate the spectra in (c) and (d) are μ_t = 1.1, μ_r = 0.60, μ_a = 0.45, μ_b = 0.30, A_{tt} = 2.0 ps^{-1}, A_{rr} = 1.6 ps^{-1}, A_{aa} = 2.9 ps^{-1}, A_{bb} = 1.0 ps^{-1}, τ_{tt} = 9.5 ps, τ_{rr} = 0.78 ps, τ_{aa} = 2.8 ps, τ_{bb} = 4.0 ps, τ_{tr} = 0.14 ps, τ_{ta} = 0.34 ps, τ_{tb} = 0.84 ps, τ_{ra} = 0.79 ps, τ_{rb} = 0.85 ps, τ_{ab} = 2.3 ps, T_{2t}^{*} = 0.21 ps, and T_2^{*} = 0.21 ns. For the ν_t mode, $|\mu_{12}| = 1.27|\mu_{01}|$.

cross peaks in the ZZZZ spectrum. Two cross peaks are located along $\omega_1 = \nu_b$, one between the ν_b and ν_t modes at $\omega_3 = \nu_t$ and one between the ν_b and ν_r modes at $\omega_3 = \nu_r$. Along $\omega_1 = \nu_r$, there is a cross peak between ν_r and ν_b at $\omega_3 = \nu_b$. Several more cross peaks are observed in the YYZZ spectrum (Fig. 2.7b). Along $\omega_1 = \nu_r$, there is a cross peak between ν_r and ν_t at $\omega_3 = \nu_t$, and along $\omega_1 = \nu_t$, there is the corresponding cross peak between ν_r and ν_t at $\omega_3 = \nu_r$. The negative feature of the cross peak at $\omega_1 = \nu_t$, $\omega_3 = \nu_r$ is overlapped with the much stronger positive feature of the ν_t diagonal peak and is not visible on the spectrum. Along $\omega_1 = \nu_t$, the negative feature of a cross peak between ν_t and ν_b at $\omega_3 = \nu_b$ can be seen. The cross peaks that are visible in both polarization geometries are stronger in the YYZZ spectrum relative to the diagonal peaks in comparison to the ZZZZ spectrum indicating that the angles between those coupled modes are closer to 90° than 0°. We note that the cross peaks involving the bridge mode have a higher intensity than the diagonal peak for the bridge mode. This arises from the fact that the dipole moments for ν_t, ν_r, and ν_a are larger than the dipole for ν_b so $|\mu_p|^2 |\mu_b|^2$ for the cross peaks is greater than $|\mu_b|^4$ for the diagonal peak (where p = t, r, a).

As observed in the FTIR spectrum of FeRu in FA, the four cyanide stretches are overlapping with each other and this makes it challenging to assign all the peaks and read off the anharmonicities from the experimental 2D spectra. For example, the negative peak below the ν_b diagonal peak does not represent the vibrational mode anharmonicity of ν_b. Instead it includes contributions from several combination bands. The indent in the contours of the cross peak at $\omega_1 = \nu_b$, $\omega_3 = \nu_a$ likely represents the position of the negative diagonal peak of ν_b.

The ZZZZ and YYZZ 2D IR spectra simulated from the best fit parameters obtained using the methodology described in Section 2.2 are shown in Figure 2.7c and 2.7d. The fit spectra capture all the features of the experimental data in terms of spectral positions, amplitudes, and lineshapes.

Like in the experimental spectra, we see two main peaks along the diagonal with a weak diagonal bridge peak. Differences between the experimental and fit spectra are observed at $\omega_1 = \nu_t$ and $\omega_3 = \nu_b$, where the positive portion of the ν_t and ν_b cross peak is visible in the fit spectra and the negative portion of the cross peak has less amplitude in the fit. However, we point out that the differences among those contour lines is only ~4%. The slight differences in amplitudes and lineshapes of the negative diagonal and cross-peak features between the experiment and simulation are attributed to the breakdown of the harmonic scaling approximation for the FFCFs, transition moment dipole strengths, and relative angles between the $n = 0 \rightarrow n = 1$ and $n = 1 \rightarrow n = 2$ transitions among all the ν_{CN} modes.

The vibrational anharmonicities and angles between the transition dipole moments determined from the fit are shown in column 1 of Table 2.2. The individual vibrational mode anharmonicities range from 14 cm^{-1} for ν_t to 27 cm^{-1} for ν_a, and the mixed-mode anharmonicities range from 2 cm^{-1} for Δ_{tr} and Δ_{ta} to 13 cm^{-1} for Δ_{ab}. The 13 cm^{-1} value for Δ_{ab} is significantly larger than the other mixed-mode anharmonicities indicating that those two modes are strongly coupled. For a perfectly linear molecule (as FeRu is depicted in Figure 2.1a), we would expect all of the angles between the transition dipoles of the four ν_{CN} modes to be either 0° or 90°. The results from the spectral fitting find that the angles between the transition dipole of the radial mode and the other three modes is ~ 90° and the transition dipoles of the modes lying along the charge transfer axis are approximately parallel to one another. The polarization-selective 2D IR spectra and their fits reveal that the FeRu molecule dissolved in FA exists in a linear geometry, and the four ν_{CN} modes are weakly coupled to each other.

2D IR spectroscopy of FeRu dissolved in D_2O. The *ZZZZ* and YYZZ 2D IR experimental spectra of FeRu in D_2O at $\tau_2 = 150$ fs are shown in Figure 2.8a and 2.8b. In contrast to the 2D IR

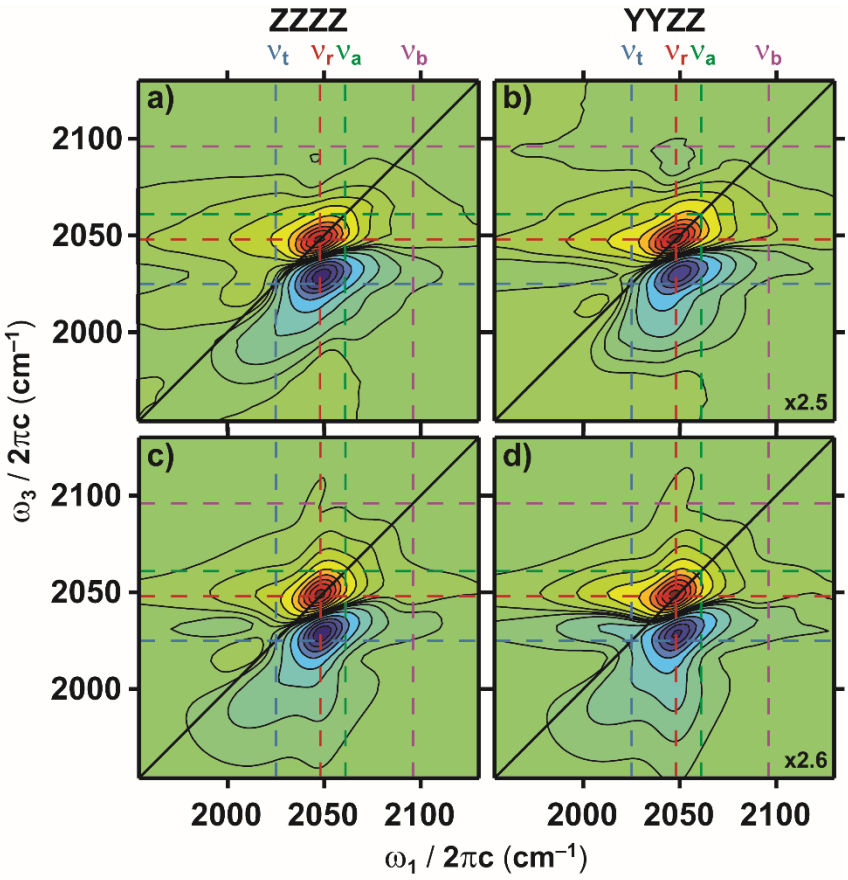


Figure 2.8: 2D IR spectra of FeRu dissolved in D_2O at $\tau_2 = 150$ fs. (a) ZZZZ experiment, (b) YYZZ experiment, (c) ZZZZ simulated, and (d) YYZZ simulated. Each spectrum is plotted using contour lines at the following positions: ±0.01, ±0.04, ±0.1, ±0.2, ±0.35, ±0.5, ±0.65, ±0.8, and ±0.95. The YYZZ spectra are multiplied by the numbers in the lower right corners. The lineshape parameters used to simulate the spectra in (c) and (d) are $\mu_t = 1$, $\mu_r = 0.923$, $\mu_a = 0.458$, $\mu_b = 0.163$, $A_{tt} = 2.9$ ps^{-1}, $A_{rr} = 2.1$ ps^{-1}, $A_{aa} = 2.6$ ps^{-1}, $A_{bb} = 2.8$ ps^{-1}, $\tau_{tt} = 4.9$ ps, $\tau_{rr} = 0.61$ ps, $\tau_{aa} = 2.9$ ps, $\tau_{bb} = 1.3$ ps, $\tau_{tr} = 0.090$ ps, $\tau_{ta} = 0.79$ ps, $\tau_{tb} = 1.3$ ps, $\tau_{ra} = 0.62$ ps, $\tau_{rb} = 1.6$ ps, $\tau_{ab} = 1.4$ ps, $T_{2t}^* = 0.24$ ps, and $T_2^* = 3.0$ ns.

spectra of FeRu in FA where there were three visible diagonal peaks, there is only one obvious diagonal peak here. This diagonal peak mainly represents the overlapped contributions from ν_r and ν_a. In the ZZZZ spectrum, there is also another positive feature at approximately $\omega_1 = \omega_3 = \nu_t$, that represents some of the contribution from the diagonal peak of ν_t. There is no spectral amplitude in the region where ν_b should have a diagonal peak. Given the relative transition dipole moment strengths of the bridge and radial modes, we estimate that the magnitude of the ν_b diagonal peak would be ~3000 times smaller than that of the ν_r diagonal peak. The relatively weak transition

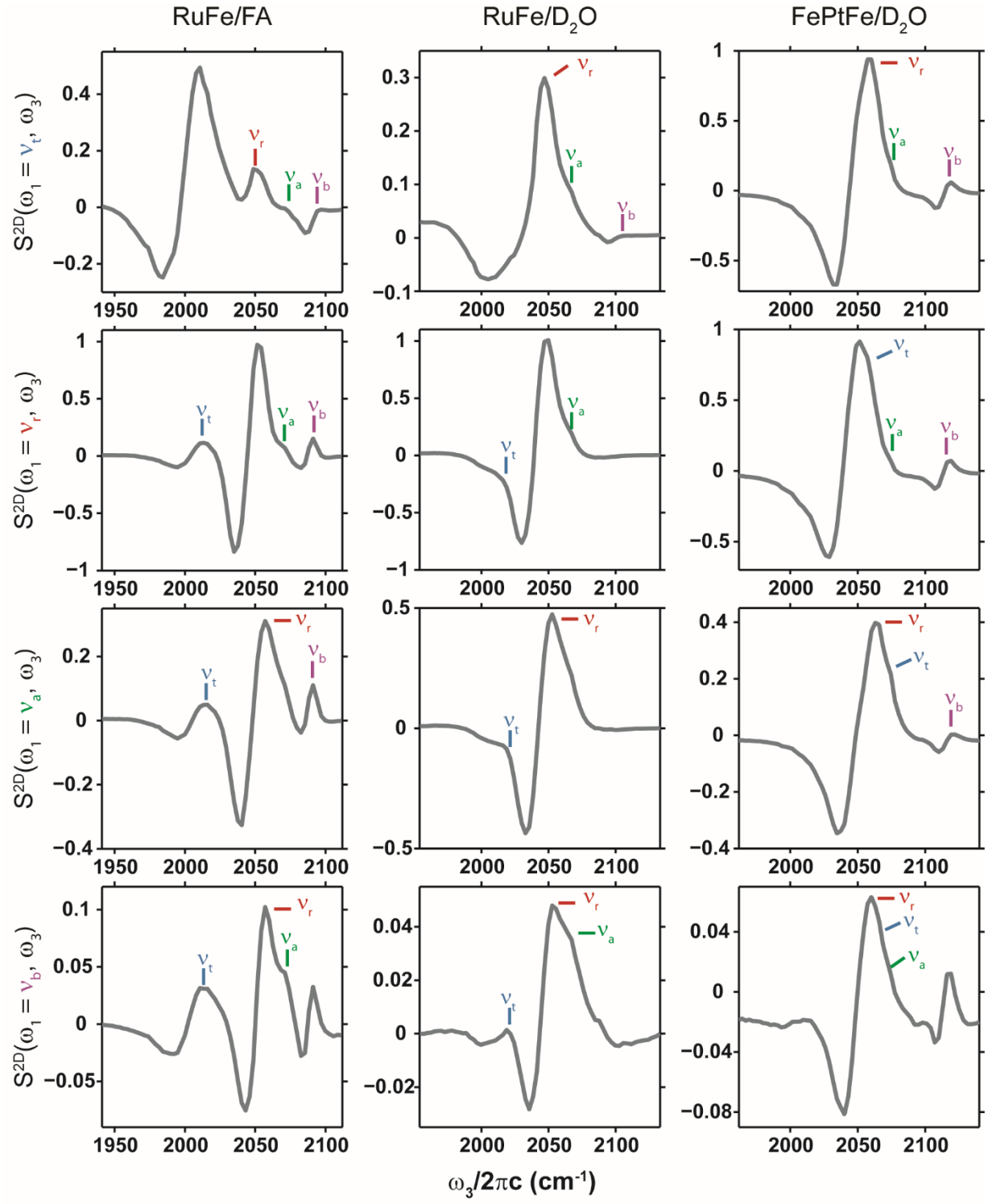


Figure 2.9: Slices of the YYZZ experimental 2D spectra (normalized) at the ω_1 frequency for each of the four modes: ν_t (blue), ν_r (red), ν_a (green), and ν_b (purple). Cross peaks are marked with the corresponding mode in ω_3.

dipole moment strength of the bridge mode results in a lack of cross peaks in the 2D IR spectra.

We estimate that the cross peak between ν_r and ν_b would have an amplitude ~50 times smaller than the ν_r diagonal peak. Some of the cross peaks are more easily observed when taking slices of the 2D spectra. Slices at the ω_1 frequency of each mode for each system are shown in Figure 2.9. The cross peaks observed when looking at the 2D spectra are spectrally overlapped with the diagonal peaks. The wing of the 2D IR lineshape at $\omega_3 = \nu_r$ seen in both polarizations is a cross peak between ν_r and ν_t. It is much stronger in the YYZZ spectrum relative to the diagonal peak in comparison to the ZZZZ spectrum, and the negative feature is not visible in the ZZZZ spectrum. The shoulder at $\omega_1 = \nu_r$ off of the negative ν_r diagonal peak to lower energy seen in the YYZZ spectrum is the negative feature of the corresponding cross peak between ν_r and ν_t.

The ZZZZ and YYZZ spectra of the fit are shown in Figure 2.8c and 2.8d. The fit has successfully captured all of the features of the experimental spectra. The most obvious differences between the experimental and fit spectra appear in the amplitude of the negative peaks due to our assumption of harmonic scaling for the magnitude of the transition dipole moments. The anharmonicities and angles obtained from this fit are in the second column of Table 2.2. The vibrational mode anharmonicities range from 15 cm^{-1} for the ν_t mode to 28 cm^{-1} for the ν_a and ν_b modes. The mixed-mode anharmonicity between the ν_t and ν_r modes is 14 cm^{-1} which is the same as Δ_t. The other mixed-mode anharmonicities range from 2 cm^{-1} for Δ_{ta} to 11 cm^{-1} for Δ_{ab}. The results from the spectral fitting find that the angles between the transition dipoles of the radial mode and the other three modes are in the range of 60°-75°. The relative angles between the trans, axial, and bridge modes are in the range of 5°-20°. The polarization-selective 2D IR spectra and their fits reveal that the FeRu molecule dissolved in D_2O exists in a bent geometry, and the four ν_{CN} modes are weakly coupled to each other.

These results highlight the importance and strong effect of solute-solvent interactions in modulating the molecular geometry and in turn the couplings between the four coupled ν_{CN} modes. The large variation in the angles between the transition dipoles of the four ν_{CN} modes measured for FeRu dissolved in FA and D_2O demonstrates how the solute-solvent interactions affect the geometry of solute molecules. When FeRu is dissolved in FA, the measured angles between the four ν_{CN} modes reveal that the molecule is almost perfectly linear. However, when FeRu is dissolved in D_2O, a bent structure is preferred. While the diagonal anharmonicities for the ν_t, ν_r, and ν_a modes are within error of each other, the anharmonicity for the ν_b mode increases by 9 cm^{-1} from FeRu in FA to FeRu in D_2O. Except for Δ_{tr}, all of the mixed-mode anharmonicities are also within error of each other. The value for Δ_{tr} increases from 2 cm^{-1} for FeRu in FA to 14 cm^{-1} for FeRu in D_2O indicating the ν_t and ν_r modes are more strongly coupled in D_2O than in FA. The increased coupling between the trans and radial modes could result from the bent molecular geometry of FeRu in D_2O. As discussed earlier, previous Resonance Raman studies of FeRu in FA and D_2O have noted that the solvent determines which of the four ν_{CN} modes couple to the non-radiative back electron transfer process in this molecule.[13] The increase in the coupling between the trans and radial modes when FeRu is dissolved in FA and D_2O respectively provides evidence on how solvents can be used to manipulate vibrational energy relaxation and charge transfer pathways in complex molecular systems used for photochemical energy conversion.

2D IR spectroscopy of FePtFe dissolved in D_2O. The ZZZZ and YYZZ 2D IR experimental spectra of FePtFe in D_2O at τ_2 = 150 fs are shown in Figure 2.10a and 2.10b. In contrast to FeRu in D_2O where there was only one visible diagonal peak, there are two diagonal peaks noticeable in both spectra for FePtFe. The more intense diagonal peak contains contributions from the ν_r, ν_t, and ν_a modes. The much smaller peak is the ν_b diagonal peak. Vibrational mode anharmonicities

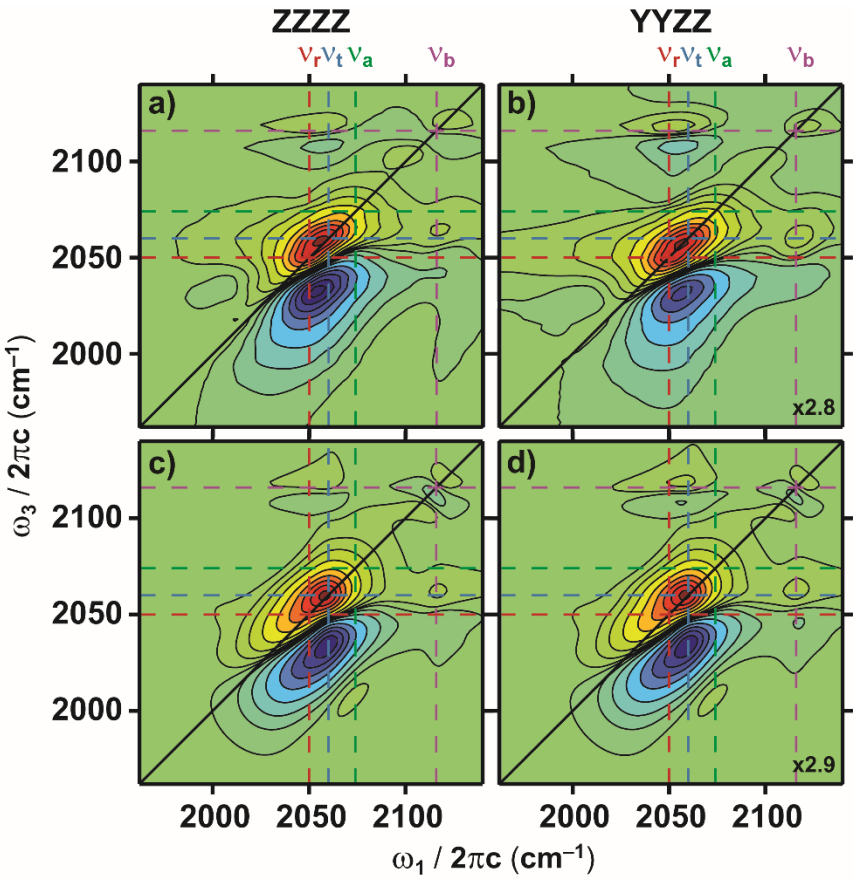


Figure 2.10: 2D IR spectra of FePtFe dissolved in D_2O at $\tau_2 = 150$ fs. (a) ZZZZ experiment, (b) YYZZ experiment, (c) ZZZZ simulated, and (d) YYZZ simulated. Each spectrum is plotted using contour lines at the following positions: ±0.01, ±0.045, ±0.1, ±0.2, ±0.35, ±0.5, ±0.65, ±0.8, and ±0.95. The YYZZ spectra are multiplied by the numbers in the lower right corners. The lineshape parameters used to simulate the spectra in (c) and (d) are $\mu_t = 0.69$, $\mu_r = 1$, $\mu_a = 0.34$, $\mu_b = 0.28$, $A_{tt} = 2.1$ ps^{-1}, $A_{rr} = 3.2$ ps^{-1}, $A_{aa} = 8.0$ ps^{-1}, $A_{bb} = 2.2$ ps^{-1}, $\tau_{tt} = 5.4$ ps, $\tau_{rr} = 1.0$ ps, $\tau_{aa} = 5.0$ ps, $\tau_{bb} = 0.27$ ps, $\tau_{tr} = 0.20$ ps, $\tau_{ta} = 1.3$ ps, $\tau_{tb} = 0.20$ ps, $\tau_{ra} = 1.1$ ps, $\tau_{rb} = 0.50$ ps, $\tau_{ab} = 1.0$ ps, and $T_2^* = 0.39$ ns.

for the ν_r, ν_t, and ν_a modes do not appear to be significantly different since the separation between the positive and negative features of the central diagonal peak does not seem to change much across its range of ω_1 frequencies. The central diagonal peak also seems to be fairly elongated along the diagonal indicating a large inhomogeneous component to its lineshape, whereas the ν_b diagonal peak seems to be fairly well rounded indicating a more homogeneous lineshape. This is in contrast to the lineshape for ν_b in FeRu in FA, where ν_b does not have much anti-diagonal width.

There are only two observable cross peaks for FePtFe that can be seen in the spectra at both polarization geometries. One cross peak is along $\omega_1 \approx \nu_r$ at $\omega_3 = \nu_b$ and the other is $\omega_1 = \nu_b$ at $\omega_3 \approx \nu_r$. Both represent the cross peak between the ν_b mode and each of the other three modes ν_r, ν_t, and ν_a. The cross peaks are once again much stronger in the YYZZ spectrum relative to the diagonal peaks than they are in the ZZZZ spectrum.

The ZZZZ and YYZZ spectra of the fit for FePtFe are shown in Figure 2.10c and 2.10d. We see that the fit reproduces all the spectral features seen in the experimental spectra. As with FeRu in D_2O, we see that most of the negative features have too much amplitude, proving that harmonic scaling is not an accurate assumption for these molecules dissolved in D_2O. The experimental spectra have what appears to be another small positive and negative set of peaks centered at $\omega_1 = \sim 1990$ cm^{-1} and $\omega_3 = \nu_r$, and the fit does not reproduce this. The peaks could be from some slight contamination of starting material or from a product of sample degradation.

The vibrational anharmonicities and angles obtained from this fit are shown in the third column of Table 2.2. The individual vibrational mode anharmonicities are all around 20 cm^{-1} with the smallest being 19 cm^{-1} for the ν_a mode and the largest at 24 cm^{-1} for both the ν_t and ν_b modes, which is a much smaller spread than the values obtained for FeRu in D_2O. The vibrational anharmonicities for the ν_t, ν_r, and ν_a modes listed in Table 2.2 are within experimental error to those previously reported.[21] The mixed-mode anharmonicities range from 2 to 15 cm^{-1}. The results from the spectral fitting find that the angles between the transition dipoles of the radial mode and the other three modes are in the range of 50°-73°. The relative angles between the trans, axial, and bridge modes are in the range of 9°-25°. The polarization-selective 2D IR spectra and their fits reveal that the FePtFe molecule dissolved in D_2O exists in a bent geometry, and the four ν_{CN} modes are weakly coupled to each other.

Comparing our fitting results for FeRu and FePtFe both dissolved in D_2O, we find that there are a lot of similarities. The vibrational mode anharmonicities for the ν_r and ν_b modes are the same within error, as are the mixed-mode anharmonicities for the ν_t and ν_b, ν_r and ν_a, and ν_a and ν_b modes. However, some of the diagonal vibrational mode anharmonicities are very different. For example, Δ_t shifts up 11 cm^{-1} and Δ_a shifts down 9 cm^{-1} from FeRu to FePtFe. The mixed-mode anharmonicities Δ_{tr}, Δ_{ta}, and Δ_{rb} also shift significantly between molecules with Δ_{tr} and Δ_{rb} larger in FeRu than in FePtFe.

Previous work on CN stretches find vibrational mode anharmonicities that range from about 15 to 36 cm^{-1}.[15,43-47] Anharmonicities for Fe-CN and Os-CN span from ~15 cm^{-1} to ~25 cm^{-1}. These values agree well with the vibrational mode anharmonicities presented here for Fe-CN systems which range from 14 to 28 cm^{-1}. Walker has reported anharmonicities for the $\nu_{trans\text{-}CN}$ and $\nu_{cis\text{-}CN}$ modes of the $[(NH_3)_5Ru^{III}NCOs^{II}(CN)_5]^-$ dimer in FA and D_2O.[15] The $\nu_{cis\text{-}CN}$ mode corresponds to our ν_r mode, and their reported value of ~20 cm^{-1} matches well with the 18 – 21 cm^{-1} we report for the ν_r in all three samples. However, the $\nu_{trans\text{-}CN}$ mode anharmonicity of ~25 cm^{-1}, which corresponds to our ν_t mode, only matches the ν_t anharmonicity of 24 cm^{-1} for the FePtFe system and is significantly larger than the ν_t anharmonicities for both FeRu systems.

Femtosecond transient IR spectroscopy on FePtFe dissolved in D_2O has measured rapid dissipation of vibrational energy along the axis of the Fe-Pt-Fe bond in the ground electronic state following MMCT excitation and subsequent ultrafast BET.[21] The directionality of the rapid vibrational energy relaxation can be explained by the average anharmonic coupling of ~11 cm^{-1} between the trans, bridge, and axial modes lying along the charge transfer axis. In comparison, the average anharmonic coupling is ~5 cm^{-1} between the radial mode (whose transition dipole moment does not lie along the charge transfer axis) and the rest of the ν_{CN} modes. These results

suggest that appropriate placement of ligand architectures in charge transfer complexes can be used to direct charge and energy migration.

2.3.3 *Extracting Anharmonic Couplings in the Site Basis Representation*

The data in Table 2.2 suggests that we can effectively describe our system of four ν_{CN} modes as weakly coupled anharmonic oscillators where the Hamiltonian is described in the site basis by Equation 2.5:[48]

$$\begin{aligned} H = & \sum_{i=t',r',a',b'} (\varepsilon_i m_i^\dagger m_i - \frac{\Delta_i}{2} m_i^\dagger m_i^\dagger m_i m_i) \\ & + \beta_{t'r'} \left(m_{t'}^\dagger m_{r'} + m_{r'}^\dagger m_{t'} \right) + \beta_{t'a'} \left(m_{t'}^\dagger m_{a'} + m_{a'}^\dagger m_{t'} \right) \\ & + \beta_{t'b'} \left(m_{t'}^\dagger m_{b'} + m_{b'}^\dagger m_{t'} \right) + \beta_{r'a'} \left(m_{r'}^\dagger m_{a'} + m_{a'}^\dagger m_{r'} \right) \\ & + \beta_{r'b'} \left(m_{r'}^\dagger m_{b'} + m_{b'}^\dagger m_{r'} \right) + \beta_{a'b'} \left(m_{a'}^\dagger m_{b'} + m_{a'}^\dagger m_{b'} \right) \end{aligned} \tag{2.5}$$

Our choice of using four individual site energies to represent each of the four vibrational states observed experimentally is dictated by the molecular structure and the photochemistry of these molecules which strongly suggest that the five terminal CN ligands are non-equivalent. In the above expression, $m_i^\dagger$ and m_i are the creation and annihilation operators of each vibrational mode respectively, ε_i is the site energy of each mode, Δ_i is the vibrational mode anharmonicity, and β_{ij} is the bilinear coupling constant between modes *i* and *j*. Our third order nonlinear experiment is capable of looking at interactions up to the two-exciton state. Here, the ν_{CN} modes are assumed to be weakly coupled, weakly anharmonic oscillators. This is a reasonable assumption as it was observed above that the modes are indeed coupled and that the magnitudes of the anharmonicities are on the order of only ~1% of the fundamental frequencies.

In order to determine the coupling constants, the above Hamiltonian is diagonalized and the first fifteen values of the trace are taken. These correspond to the zero, one, and two-quantum states in

the normal mode basis. These values are compared to the energy levels we determined from the experimental spectra. This transformation process and comparison is repeated iteratively to minimize the difference between the experimentally determined normal mode anharmonicities and frequencies and the calculated anharmonicities and frequencies. The site energies and coupling constants from these best fits are reported in Table 2.3.

Table 2.3: Best fit values for the local mode site energies, vibrational mode anharmonicities, and bilinear coupling constants for all three systems. The error bars are estimated to be ±2 cm^{-1} for the site energies and ±1 cm^{-1} for the anharmonicities and coupling constants.

(cm^{-1})	FeRu/FA	FeRu/D_2O	FePtFe/D_2O
$\varepsilon_{t'}$	2004	2032	2062
$\varepsilon_{r'}$	2052	2044	2051
$\varepsilon_{a'}$	2062	2057	2071
$\varepsilon_{b'}$	2087	2091	2113
$\Delta_{t'}$	18	21	22
$\Delta_{r'}$	21	21	22
$\Delta_{a'}$	20	21	26
$\Delta_{b'}$	21	21	25
$\beta_{t'r'}$	6	11	3
$\beta_{t'a'}$	5	6	6
$\beta_{t'b'}$	6	10	8
$\beta_{r'a'}$	3	2	4
$\beta_{r'b'}$	4	5	4
$\beta_{a'b'}$	6	9	7

The site energies do not seem to shift far from their normal mode positions with the largest change being a 6 cm^{-1} shift for the ν_t mode in FeRu in D_2O which had the largest changes for all modes overall. We also see that all the ν_{CN} modes have similar local mode vibrational anharmonicities. They range from 18 cm^{-1} for the $\nu_{t'}$ mode in FeRu in FA to 26 cm^{-1} for the $\nu_{a'}$

mode in FePtFe. The values of the coupling constants indicate that all of the ν_{CN} modes are weakly coupled to each other in the site basis showing that the vibrations are fairly localized on the CN ligands. The strongest coupling in the local mode basis set seems to occur between the $\nu_{t'}$ and $\nu_{b'}$ modes and the $\nu_{a'}$ and $\nu_{b'}$ modes. We also note that the $\nu_{r'}$ mode does not couple strongly with the $\nu_{a'}$ or $\nu_{b'}$ modes.

The site Hamiltonian calculated using Equation 2.5 agrees with previous transient IR work and resonant Raman work on FeRu and FePtFe which have revealed that the vibrational modes lying parallel to the charge transfer axis are crucial players in the photochemistry of these systems following MMCT excitation.[20,21] We have previously reported that the energy splitting between the trans and bridge modes in FePtFe decreases by about ~7 cm^{-1} in the ground electronic state following 400 nm MMCT excitation and back–electron transfer.[20] To see the effect that MMCT excitation has on the bilinear coupling of the trans and bridge modes in the local site basis, we move the frequency of the ν_t mode up by 7 cm^{-1} and rerun the fitting routine described above making the assumption that the other modes and their anharmonicities have not changed. We find that in the electronic ground state the bilinear coupling constant between the bridge and trans mode, in the site basis is 8 cm^{-1}, but after photoexcitation, $\beta_{t'b'}$ becomes 7 cm^{-1}. This indicates that the two modes are less strongly coupled immediately following photoexcitation. The slight change in coupling could arise from differences in molecular structure and/or differences in anharmonic vibrational couplings to the solvent bath which could include other intramolecular and intermolecular vibrational modes.

2.4 SUMMARY

This study elucidates the role of the solvent in modulating the molecular structure and vibrational anharmonic couplings between the high frequency CN stretching modes in the ground

electronic state of bimetallic and trimetallic transition metal mixed valence complexes. We measure anharmonic couplings and angles between transition moment dipoles of the ν_{CN} modes in FeRu in FA and D_2O and FePtFe in D_2O by simulating and fitting experimental 2D IR spectra in two different polarization geometries. We stress that the accurate determination of these parameters would not have been possible without rigorously fitting the spectra due to overlapping vibrational modes and large linewidths relative to the vibrational anharmonicities. The values for anharmonic couplings reveal that all four modes are weakly coupled with each other in FeRu and FePtFe. The radial mode seems to be the least coupled to the other three vibrational modes which is attributed to the fact that it does not lie along the main MMCT axis. Previous results have shown that the modes lying along the charge transfer axis are involved in the ultrafast charge and energy transfer following MMCT excitation. The weak coupling of the radial mode to the trans, axial, and bridge modes explains why it is the least coupled to the charge transfer excitation. The measured angles reveal that the solvent plays a large role in determining the molecular geometry of the mixed valence complexes. FeRu in FA appears almost perfectly linear while both FeRu in D_2O and FePtFe in D_2O appear bent. The modulation of the molecular geometry and vibrational anharmonic couplings due to specific solute-solvent interactions in the ground electronic state help explain why the coupling of high frequency CN modes to the MMCT transition is a strong function of the solvent environment. These findings hold implications for designing new cyanide-bridged transition metal complexes and polymers and for effectively simulating the spectroscopic properties of these solvated molecules.

REFERENCES

[1] Bernhardt, P. V.; Bozoglian, F.; Macpherson, B. P.; Martinez, M. *Coord. Chem. Rev.* **2005**, *249*, 1902.

[2] Ohba, M.; Okawa, H. *Coord. Chem. Rev.* **2000**, *198*, 313.

[3] Shatruk, M.; Avendano, C.; Dunbar, K. R. *Prog. Inorg. Chem.* **2009**, *56*, 155.

[4] Barbara, P. F.; Meyer, T. J.; Ratner, M. A. *J. Phys. Chem.* **1996**, *100*, 13148.

[5] Bixon, M.; Jortner, J. In *Electron Transfer-from Isolated Molecules to Biomolecules, Pt 1*; Jortner, J., Bixon, M., Eds. 1999; Vol. 106, p 35.

[6] Hush, N. S. *Electrochim. Acta* **1968**, *13*, 1005.

[7] Son, D. H.; Kambhampati, P.; Kee, T. W.; Barbara, P. F. *J. Phys. Chem. A* **2002**, *106*, 4591.

[8] Reid, P. J.; Silva, C.; Barbara, P. F.; Karki, L.; Hupp, J. T. *J. Phys. Chem.* **1995**, *99*, 2609.

[9] Tominaga, K.; Kliner, D. A. V.; Johnson, A. E.; Levinger, N. E.; Barbara, P. F. *J. Chem. Phys.* **1993**, *98*, 1228.

[10] Walker, G. C.; Aakesson, E.; Johnson, A. E.; Levinger, N. E.; Barbara, P. F. *J. Phys. Chem.* **1992**, *96*, 3728.

[11] Kliner, D. A. V.; Tominaga, K.; Walker, G.; Barbara, P. *J. Am. Chem. Soc.* **1992**, *114*, 8323.

[12] Arnett, D. C.; Voehringer, P.; Scherer, N. F. *J. Am. Chem. Soc.* **1995**, *117*, 12262.

[13] Wang, C.; Mohney, B. K.; Williams, R. D.; Petrov, V.; Hupp, J. T.; Walker, G. C. *J. Am. Chem. Soc.* **1998**, *120*, 5848.

[14] Wang, C.; Mohney, B. K.; Akhremitchev, B. B.; Walker, G. C. *J. Phys. Chem. A* **2000**, *104*, 4314.

[15] Tivansky, A.; Wang, C.; Walker, G. *J. Phys. Chem. A* **2003**, *107*, 9051.

[16] Doorn, S. K.; Hupp, J. T. *J. Am. Chem. Soc.* **1989**, *111*, 1142.

[17] Doorn, S. K.; Stoutland, P. O.; Dyer, R. B.; Woodruff, W. H. *J. Am. Chem. Soc.* **1992**, *114*, 3133.

[18] Doorn, S. K.; Dyer, R. B.; Stoutland, P. O.; Woodruff, W. H. *J. Am. Chem. Soc.* **1993**, *115*, 6398.

[19] Watson, D. F.; Tan, H. S.; Schreiber, E.; Mordas, C. J.; Bocarsly, A. B. *J. Phys. Chem. A* **2004**, *108*, 3261.

[20] Lynch, M. S.; Slenkamp, K. M.; Khalil, M. *J. Chem. Phys.* **2012**, *136*, 241101.

[21] Lynch, M. S.; Van Kuiken, B. E.; Daifuku, S. L.; Khalil, M. *J. Phys. Chem. Lett.* **2011**, *2*, 2252.

[22] Macatangay, A.; Mazzetto, S.; Endicott, J. *Inorg. Chem.* **1999**, *38*, 5091.

[23] Watzky, M.; Macatangay, A.; VanCamp, R.; Mazzetto, S.; Song, X.; Endicott, J.; Buranda, T. *J. Phys. Chem. A* **1997**, *101*, 8441.

[24] Watzky, M. A.; Endicott, J. F.; Song, X.; Lei, Y.; Macatangay, A. *Inorg. Chem.* **1996**, *35*, 3463.

[25] Lynch, M. S.; Slenkamp, K. M.; Cheng, M.; Khalil, M. *J. Phys. Chem. A* **2012**, *116*, 7023.

[26] Pfennig, B. W.; Bocarsly, A. B. *J. Phys. Chem.* **1992**, *96*, 226.

[27] Pfennig, B. W.; Wu, Y.; Kumble, R.; Spiro, T.; Bocarsly, A. B. *J. Phys. Chem.* **1996**, *100*, 5745.

[28] Hennessy, M. H.; Soos, Z. G.; Watson, D. F.; Bocarsly, A. B. *J. Phys. Chem. B* **2000**, *104*, 10909.

[29] Pfennig Brian, W.; Mordas Carolyn, J.; McCloskey, A.; Lockard Jenny, V.; Salmon Patty, M.; Cohen Jamie, L.; Watson David, F.; Bocarsly Andrew, B. *Inorg. Chem.* **2002**, *41*, 4389.

[30] Watson, D. F.; Bocarsly, A. B. *Coord. Chem. Rev.* **2001**, *211*, 177.

[31] Zhou, M. S.; Pfennig, B. W.; Steiger, J.; Vanengen, D.; Bocarsly, A. B. *Inorg. Chem.* **1990**, *29*, 2456.

[32] Vance, F. W.; Slone, R. V.; Stern, C. L.; Hupp, J. T. *Chem. Phys.* **2000**, *253*, 313.

[33] Burewicz, A.; Haim, A. *Inorg. Chem.* **1988**, *27*, 1611.

[34] Van Kuiken, B. E.; Valiev, M.; Daifuku, S. L.; Bannan, C.; Strader, M. L.; Cho, H.; Huse, N.; Schoenlein, R. W.; Govind, N.; Khalil, M. *J. Phys. Chem. A* **2013**, *117*, 4444.

[35] Vogler, A.; Kisslinger, J. *J. Am. Chem. Soc.* **1982**, *104*, 2311.

[36] Brookes, J. F.; Slenkamp, K. M.; Lynch, M. S.; Khalil, M. *J. Phys. Chem. A* **2013**, *117*, 6234.

[37] Golonzka, O.; Tokmakoff, A. *J. Chem. Phys.* **2001**, *115*, 297.

[38] Hochstrasser, R. M. *Chem. Phys.* **2001**, *266*, 273.

[39] Khalil, M.; Demirdöven, N.; Tokmakoff, A. *J. Phys. Chem. A* **2003**, *107*, 5258.

[40] Mukamel, S. *Principles of Nonlinear Optical Spectroscopy*; Oxford University Press: New York, 1995.

[41] Sung, J.; Silbey, R. J. *J. Chem. Phys.* **2001**, *115*, 9266.

[42] Berne, B. J. P. R. *Dynamic light scattering : with applications to chemistry, biology, and physics*; Dover Publications: Mineola, N.Y., 2000.

[43] Sando, G. M.; Zhong, Q.; Owrutsky, J. C. *J. Chem. Phys.* **2004**, *121*, 2158.

[44] Ohta, K.; Maekawa, H.; Tominaga, K. *J. Phys. Chem. A* **2004**, *108*, 1333.

[45] Urbanek, D. C.; Vorobyev, D. Y.; Serrano, A. L.; Gai, F.; Hochstrasser, R. M. *J. Phys. Chem. Lett.* **2010**, *1*, 3311.

[46] Kim, Y. S.; Hochstrasser, R. M. *Proc. Natl. Acad. Sci. U.S.A.* **2005**, *102*, 11185.

[47] Ha, J. H.; Lee, K. K.; Park, K. H.; Choi, J. H.; Jeon, S. J.; Cho, M. *J. Chem. Phys.* **2009**, *130*, 204509.

[48] Woutersen, S.; Hamm, P. *J. Phys.: Condens. Matter* **2002**, *14*, R1035.

Chapter 3:

Investigating Vibrational Relaxation In Cyanide-Bridged Transition Metal Mixed Valence Complexes Using Two-Dimensional Infrared And Infrared Pump-Probe Spectroscopies

The work presented in this chapter is in preparation for publication:
Slenkamp, K. M.; Lynch, M. S.; Brookes, J. F.; Bannan, C. C.; Daifuku, S. L.; and Khalil, M. "Investigating Vibrational Relaxation in Cyanide-Bridged Transition Metal Mixed-Valence Complexes Using Two-Dimensional Infrared and Infrared Pump-Probe Spectroscopies." *Journal of Physical Chemistry B* **2015**, *In preparation*.

Using polarization-selective two-dimensional infrared (2D IR) and infrared pump-probe spectroscopies, we study vibrational relaxation of the four cyanide stretching (ν_{CN}) vibrations found in $[(NH_3)_5Ru^{III}NCFe^{II}(CN)_5]^-$ (FeRu) dissolved in D_2O or formamide and $[(NC)_5Fe^{II}CNPt^{IV}(NH_3)_4NCFe^{II}(CN)_5]^{4-}$ (FePtFe) dissolved in D_2O. These cyanide-bridged transition metal complexes serve as models for understanding the role high frequency vibrational modes play in metal-to-metal charge transfers over a bridging ligand, but there is currently little information about their electronic ground state. IR pump-probe experiments reveal that the vibrational lifetimes of the radial, axial, and trans ν_{CN} modes are ~2 times faster when FeRu is dissolved in D_2O versus formamide. They also reveal that the vibrational lifetimes of the ν_{CN} modes of FePtFe in D_2O are almost four times as long as for FeRu in D_2O indicating that the lifetimes are not strongly dependent on solvent, higher molecular complexity, or larger overall charge as these factors would all result in faster relaxation times for FePtFe. Combined with mode-specific relaxation dynamics measured from the 2D IR experiments, IR pump-probe experiments

also reveal that intramolecular vibrational relaxation is occurring in all three systems on ~1 ps timescale. Center line slope (CLS) dynamics, which have been shown to be a measure of the frequency-frequency correlation function, reveal that the radial, axial, and trans ν_{CN} modes exhibit a ~3 ps timescale that the bridge ν_{CN} mode does not display. This timescale is attributed to the forming and breaking of hydrogen bonds between each mode and the solvent. In contrast to previous work done by our group, there does not appear to be a large solvent dependence for vibrational relaxation dynamics in these systems.

3.1 INTRODUCTION

Vibrational relaxation dynamics reflect differences in the local environment and therefore depend on specific interactions between solvent molecules and the local vibrational probe. Changing dynamics between systems, or even different vibrational modes spanning different regions in a molecule, highlight how excess vibrational energy is transferred from a vibrational mode to its environment. Energy dissipation has implications for chemical reactivity and the mechanisms by which those reactions occur. In polyatomic molecules in solution, this energy dissipation can occur by intramolecular vibrational relaxation (IVR), where the energy is transferred to other modes within the same molecule, or by direct relaxation to solvent modes. Rates of relaxation can be affected by factors that include the frequency of the vibrational mode, the size, structure, and charge of the solute molecule, and similar features of the solvent. Ultrafast spectroscopy is the most common technique used to study vibrational relaxation in the condensed phase.

Cyanide-bridged transition metal mixed valence complexes serve as model systems to study charge transfer reactions. High-frequency vibrational modes in these systems function as a probe of the interaction between vibrational and electronic motions following photoinduced electron

transfer reactions. Previous studies of cyanide-bridged mixed valence complexes have shown high degrees of excitation, in some cases greater than 6 quanta, in high frequency CN stretching (ν_{CN}) vibrations following back electron transfer subsequent to the metal-to-metal charge transfer (MMCT) reaction,[1-6] but little is known of vibrational dynamics on the electronic ground state. The goal of this chapter is to determine what role solvent and other molecular properties play in the vibrational relaxation of cyanide-bridged bimetallic and trimetallic transition metal mixed valence complexes. Infrared pump-probe and two-dimensional infrared spectroscopies are used to measure ground state vibrational dynamics of the cyanide stretching vibrations of $[(NH_3)_5Ru^{III}NCFe^{II}(CN)_5]^-$ (FeRu) dissolved in deuterium oxide (D_2O) or formamide (FA) and $[(NC)_5Fe^{II}CNPt^{IV}(NH_3)_4NCFe^{II}(CN)_5]^{4-}$ (FePtFe) dissolved in D_2O. FeRu and FePtFe both have four cyanide stretching frequencies that are labelled ν_{trans} (ν_t), ν_{radial} (ν_r), ν_{axial} (ν_a), and ν_{bridge} (ν_b) (illustrated in Fig. 3.1a,b) of which three (ν_t, ν_a, and ν_b) lie approximately along the MMCT axis.[7]

The FTIR spectra of the ν_{CN} modes (see Fig. 3.1c-e) of the bimetallic and trimetallic systems are strikingly different in terms of frequencies, amplitudes, and lineshapes. The four mode frequencies for the FeRu/FA system are centered at 2002, 2051, 2065, and 2089 cm^{-1} for the ν_t, ν_r, ν_a, and ν_b modes respectively. For the FeRu/D_2O system, they are centered at 2026, 2048, 2061, and 2096 cm^{-1}, and for FePtFe, they are centered at 2059, 2049, 2072, and 2116 cm^{-1} for the ν_t, ν_r, ν_a, and ν_b modes respectively. The higher oxidation state of Pt(IV) compared to Ru(III) explains why the frequency of the bridging mode is farther to the blue for FePtFe than for FeRu. The larger positive charge pulls more electron density out of the cyanide antibonding (π^*) orbital and increases the strength of the CN bond thus increasing its vibrational frequency.[8-10] If the electron withdrawing effect carries along the MMCT axis, the strength of the bond could also increase in the trans and

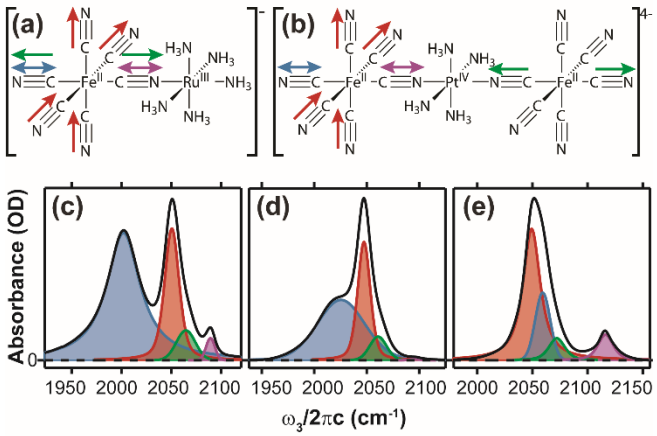


Figure 3.1: (a) FeRu and (b) FePtFe with their ν_{CN} modes illustrated by colored arrows ν_{trans} (ν_t, blue), ν_{radial} (ν_r, red), ν_{axial} (ν_a, green), and ν_{bridge} (ν_b, purple). Solvent subtracted FTIR spectra (black line) of (c) FeRu in FA (d) FeRu in D_2O and (e) FePtFe in D_2O. Each FTIR spectrum was fit to four Voigt lineshapes,[7] and each peak is color coded to match the four modes shown in panels (a) and (b).

axial cyanide ligands explaining why the ν_t mode in FePtFe is at much higher frequencies than it is in FeRu and why the frequency ordering of the four ν_{CN} modes change between systems.

Previous work by our group has shown the experimental 2D IR spectra of FeRu and FePtFe and their fits reveal a set of weakly coupled anharmonic ν_{CN} modes. The vibrational mode anharmonicities of the individual ν_{CN} modes range from 14 to 28 cm^{-1}, and the mixed-mode anharmonicities range from 2 to 14 cm^{-1}. In general, the bridging ν_{CN} mode is most weakly coupled to the radial ν_{CN} mode, which involves the terminal CN ligands approximately orthogonal to the main MMCT axis. Measurement of the relative transition dipole moments of the four ν_{CN} modes reveal that the angles in FeRu in FA are significantly different than both FeRu in D_2O and FePtFe in D_2O suggesting structure might be a function of solvent environment.[7] Here we will determine if the solvent plays a similar role in the vibrational dynamics in these systems.

The remainder of this chapter is organized as follows. Section 3.2 describes the methods used to prepare samples and collect both pump-probe and 2D IR spectra. Section 3.3 is split into two subsections. Section 3.3.1 describes the IR pump-probe fitting and results. Section 3.3.2 contains

the results of the 2D IR experiments including spectral diffusion and mode-specific dynamics. Section 3.4 contains the discussion of our results and is split into two subsections. Section 3.4.1 discusses vibrational relaxation and Section 3.4.2 discusses spectral diffusion dynamics. The final section, Section 3.5, contains the summary of the results and our conclusions.

3.2 METHODS

Materials. Starting materials and solvents were purchased from Sigma Aldrich and used without further purification. The mixed-valence complexes, FeRu and FePtFe, were synthesized as described previously.[7,11-13] A saturated solution of FeRu was prepared in formamide (FA) with a maximum optical density (OD) of 0.22 in the ν_{CN} region after solvent subtraction. A solution of FeRu was also prepared in D_2O to a concentration of ~25 mM and maximum OD of 0.36 in the ν_{CN} region. A solution of FePtFe dissolved in D_2O was prepared to a concentration of ~15 mM and a maximum OD of 0.45 for use in experiments.

2D IR Experiments. The experimental parameters and layout of the 2D IR experiments have been described in detail in Ref. 7. 2D IR spectra of FeRu in FA were collected at the following τ_2 delays (waiting times): 0.07, 0.14, 0.225, 0.3, 0.425, 0.475, 0.625, 1, 2.5, 5, 7.5, 10, 15, and 20 ps. 2D IR spectra of FeRu/D_2O were collected at τ_2 delays of 0.09, 0.15, 0.225, 0.3, 0.375, 0.65, 1, 3, 5, 7.5, 10, and 15 ps, and 2D IR spectra of FePtFe dissolved in D_2O were taken at τ_2 delays of 0.15, 0.34, 0.38, 1.36, 2.04, 5, 7.5, 10, and 15 ps. For each sample, 2D IR spectra were collected in both parallel $\left(S_{\parallel}\right)$ and crossed $\left(S_{\perp}\right)$ polarization geometries. Magic angle spectra were constructed using the relationship: $S_M = \frac{1}{3}\left(S_{\parallel} + 2S_{\perp}\right)$.

IR Pump-Probe Experiments. The generation and manipulation of the mid-IR light pulses used in the IR pump-probe experiments was similar to the 2D IR experiments and has been

described earlier. The pump beam (chopped at 500 Hz) and the probe beam were spatially and temporally overlapped in the sample to generate the pump-probe signal. An additional mid-IR pulse was also sent through the sample to act as a reference beam. The input pump and probe beams had energies of ~0.5 and ~0.3 μJ/pulse for all three samples. All input beams were focused onto the sample with a spot size ($1/e^2$) of ~150 μm. The pump-probe signal and the reference beam were vertically offset before being dispersed by a spectrometer (Triax 190, Horiba Jobin Yvon) and detected on the upper and lower stripes of a 2×64 element mercury cadmium telluride array detector (IR0144, Infrared Systems Development).

Pump-probe spectra for the FeRu samples were collected at magic angle where the polarization of the probe beam was rotated 54.7° relative to the polarization of the pump beam to eliminate contributions from rotational dynamics. Collected data were generated by dividing both the pump on and pump off signals by their respective reference pulses and then subtracting the pump on signal from the pump off signal. Dividing each signal pulse by the reference helped mitigate the effects of shot-to-shot and long term noise fluctuations in the mid-IR field. The reference beam was set to come about 150 ps before the pump beam to ensure the sample had relaxed back to the ground state. For FeRu dissolved in FA and D_2O, the time delay between the pump and probe pulses was scanned from -150 to150 fs in 10 fs steps, 150 to 2500 fs in 25 fs steps, 2.5 to 5 ps in 50 fs steps, 5 to 8 ps in 250 fs steps, 8 to 20 ps in 1 ps steps, 20 to 40 ps in 4 ps steps, and 40 to 100 ps in 10 ps steps. Each data point represents 1500 shots, and each spectrum was collected 100 times and averaged to obtain the experimental spectra presented here. The experimental spectral resolution is ~2.8 cm^{-1} for the FeRu samples.

Spectra for FePtFe were collected in both parallel $\left(S_{\parallel}\right)$ and crossed $\left(S_{\perp}\right)$ polarization geometries and no reference beam was used for shot-to-shot normalization of the data. For FePtFe

dissolved in D_2O, the time delay between the pump and probe pulses was scanned from -1.5 to 3.5 ps in 25 fs steps, 3.5 to 6 ps in 250 fs steps, 6 to 20 ps in 1 ps steps, and 20 to 100 ps in 4 ps steps. Each data point represents 2000 shots, and each spectrum was collected 25 times and averaged to obtain the experimental spectra presented here. Magic angle spectra were constructed. Experimental spectral resolution is ~4.4 cm^{-1} for the FePtFe sample.

3.3 RESULTS

Table 3.1: Parameters from the exponential fits to experimental pump-probe data. Error bars represent 95% confidence intervals.

	Mode	Freq (cm^{-1})	t_1 (ps) (A_1)	t_2 (ps) (A_2)	t_3 (ps) (A_3)	t_{rise} (ps) (A_{rise})	t_{osc} (ps) (A_{osc})	ω_{osc} (cm^{-1})
FeRu/FA	Trans	2004	0.31±0.02 (-0.52)	0.99±0.04 (-0.44)	10.±1 (-0.04)			
	Radial	2050	0.64±0.04 (-0.18)	10.8±0.1 (-0.77)			0.2±0.1 (-0.05)	50±10
	Axial	2064	9.32±0.09 (-0.97)				0.7±0.3 (-0.03)	90.±4
	Bridge	2091	12.4±0.5 (-0.51)			1.4±0.2 (-0.14)	0.30±0.02 (0.35)	84±1
FeRu/D_2O	Trans	2028	0.42±0.03 (0.15)	5.81±0.05 (0.85)				
	Radial	2047	0.72±0.03 (-0.24)	6.51±0.09 (-0.76)				
	Axial	2064	5.25±0.06 (-1)					
	Bridge	2093	0.41±0.06 (1)					
FePtFe/D_2O	Trans	2060	0.85±0.07 (-0.12)	17.2±0.2 (-0.88)				
	Radial	2050	0.5±0.1 (-0.13)	19.4±0.2 (-0.76)			0.34±0.06 (-0.11)	33±4
	Axial	2073	16.1±0.3 (-0.71)			1.11±0.09 (-0.20)	0.57±0.08 (-0.09)	52±2
	Bridge	2115	18±1 (-0.26)			0.82±0.08 (-0.23)	0.46±0.02 (-0.51)	53.6±0.6

3.3.1 *IR Pump-Probe*

The IR pump-probe spectra of FeRu/FA, FeRu/D_2O, and FePtFe/D_2O are shown in Figure 3.2. The time-dependence of the fundamental ν_{CN} frequencies of each sample are shown on semi-log plots. The data was fit to the following expression:

$$\Delta A(\tau_2,\omega_3)=\left(\sum_i A_i \exp\left(-\tau_2/t_i\right)\right)+B\left(1-\exp\left(-\tau_2/t_{rise}\right)\right)+C\exp\left(-\tau_2/t_{osc}\right)\cos\left(2\pi c\omega_{osc}\tau_2+\phi\right) \quad (3.1)$$

where $A_i(B)$ and t_i are the amplitude and timescale of each decay (rise) respectively, ω_{osc} is the frequency of any oscillation that is present, and ϕ is the phase of the oscillation. Because of large nonresonant signal, data begins at 200 fs and was only fit from this time forward. The best fit parameters are listed in Table 3.1.

FeRu/FA. Figure 3.2a displays τ_2 dependent dispersed pump-probe traces of FeRu/FA in the right panel. The data reveal that the negative bleach signals ($0\rightarrow1$ transitions) of the ν_r (2051 cm^{-1}) and ν_a (2065 cm^{-1}) modes are spectrally overlapped and the weaker axial mode appears as a shoulder to the blue side of the radial mode. It is also evident from Figure 3.2a that the ν_t mode (2004 cm^{-1}) decays at a much faster rate than the other three modes. The positive absorption signals represent the $1\rightarrow2$ transitions to the various overtones and combination bands from the first excited vibrational states. We are not analyzing the $1\rightarrow2$ transitions due to the fact that they have highly overlapped contributions from both combination bands and overtones and therefore do not represent dynamics of any transition exclusively. The vibrational relaxation dynamics of the four fundamental transitions are plotted in the left panel of Figure 3.2a and the data is fit using Equation 3.1. The traces in Figure 3.2b and the best fit parameters listed in Table 3.1 reveal that the four fundamental ν_{CN} modes display different vibrational relaxation dynamics. The ν_t mode

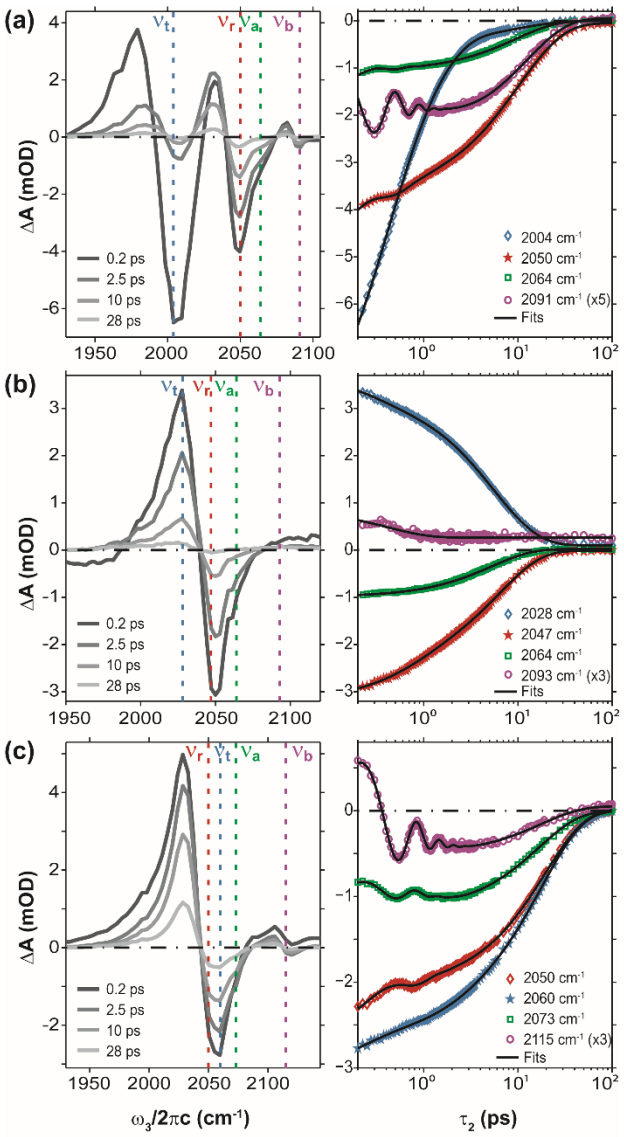


Figure 3.2: The two panels of (a) represent spectra (left) of FeRu in FA at several τ_2 delays and time traces (right) of the same signal for select frequencies shown as dashed lines in the left panel. Note that the time axis follows a logarithmic scale. (b) and (c) display the same information for FeRu in D_2O and FePtFe in D_2O, respectively, as described in (a). Note that the bridge trace for each sample is multiplied by a static factor to make it easier to compare to the other modes.

decays on two main timescales: 0.31 and 0.99 ps. We attribute the faster timescale to intramolecular vibrational relaxation (IVR) processes where the vibrational energy leaks into other high-frequency ν_{CN} modes or low-frequency vibrational modes along the metal-ligand backbone. The 0.99 ps decay timescale is attributed to vibrational energy relaxation to the solvent from the

excited ν_t mode. The fit of the ν_r mode reveals three timescales that consist of two exponential decays and a decaying cosine. We attribute the 10.8 ps decay to the vibrational population lifetime of the radial mode. The faster 0.64 ps decay is attributed to cross-population relaxation processes to other intramolecular vibrational modes. The oscillation at ~50 cm^{-1} observed in the relaxation dynamics of the ν_r mode results from the coherent superposition of two coupled ν_{CN} modes. The amplitude of the oscillatory component is proportional to the anharmonic coupling and the transition dipole moment between coupled modes. The frequency difference between the radial and trans modes and between the radial and bridge modes is ~50 and ~40 cm^{-1} respectively. We have previously measured similar anharmonic couplings and angles between the radial and trans modes and between radial and bridge modes suggesting that the oscillation arises from both of these coherent contributions. In addition, clear cross-peaks between the ν_t and ν_r and ν_t and ν_b modes can be seen in Figure 3.3a at $\omega_3 = \nu_t$. The fit of the ν_a mode (2064 cm^{-1}) shows two timescales consisting of an exponential decay and a very weak oscillatory component. The 9.32 ps timescale is the vibrational lifetime of the axial mode while the faster component is attributed to IVR processes. The weak oscillatory component at ~90 cm^{-1} likely results from spectral overlap with combination bands. The trace of the ν_b mode (2091 cm^{-1}) displays an exponential decay, a growth, and a decaying oscillation at ~84 cm^{-1}. The oscillation matches the difference in frequency between ν_t and ν_b indicating a superposition between the two modes which have parallel transition dipole moments. The decay of 12.4 ps is the vibrational lifetime of the bridge mode, and the growth at 1.4 ps is attributed to vibrational energy relaxation from the other three ν_{CN} modes. In summary, we note that the radial, axial, and bridge modes of FeRu/FA have vibrational lifetimes ranging from 9-12 ps. The trans mode is an outlier with a vibrational lifetime of ~1 ps. The lifetimes of the coherent oscillations found in the ν_r, ν_a, and ν_b are less than 1 ps.

FeRu/D_2O. Figure 3.2b shows the dispersed pump-probe spectra of the FeRu/D_2O sample as a function of ω_3 and τ_2, and the fit values for the fundamental traces can be seen in the FeRu/D_2O section of Table 3.1. We note that the ν_t (2028 cm^{-1}), ν_r (2047 cm^{-1}), and ν_a (2064 cm^{-1}) modes are spectrally overlapped and the ν_b mode (2093 cm^{-1}) is extremely weak. In contrast to the pump-probe data for FeRu in FA, the traces of FeRu in D_2O show no rises or oscillations. The fit of the ν_r mode has two timescales, where the short timescale is attributed to IVR processes, and the 6.51 ps timescale is attributed to vibrational energy relaxation to the solvent. The ν_a mode only decays on a timescale of 5.25 ps, which represents its vibrational population lifetime. The fit of the ν_t mode consists of two exponential decays that have amplitude factors of opposite signs to the rest of the fundamental transitions. The fundamental transition for the ν_t mode is overlapped with overtones of other modes as well as contributions from combination bands which results in the positive signal. This makes it difficult to attribute a vibrational population lifetime to the trans mode. Given that the vibrational lifetimes of the ν_r and ν_a modes are approximately half of their measured values in the FeRu/FA sample, we tentatively assign the 0.42 ps decay to be the vibrational lifetime of the ν_t mode. Since the ν_b mode region of the spectrum has such a low signal level, fit values for that trace likely have no physical meaning. In summary, we note that the radial and axial modes of FeRu/D_2O have lifetimes of ~6 ps, and the trans mode is again an outlier at ~400 fs. These timescales are twice as fast as those seen in FeRu/FA.

FePtFe/D_2O. Figure 3.2c shows τ_2 dependent dispersed pump-probe spectra of FePtFe and ω_3 dependent vibrational relaxation dynamics. The ν_r (2050 cm^{-1}), ν_t (2060 cm^{-1}), and ν_a (2073 cm^{-1}) modes are spectrally overlapped, but their individual contributions can be distinguished as shoulders in the strong bleach contribution in the dispersed pump-probe spectra. Similar to the FeRu sample dissolved in FA, we see that three of the modes in FePtFe have oscillations, but here

two modes have visible rises. From the fit values seen in Table 3.1, we can see that all four modes share a similar long time constant of ~18 ps representing vibrational population lifetimes for each mode. The ν_t mode does not display a coherent oscillation, but the fit shows a second decay in addition to its vibrational lifetime which is attributed to relaxation processes to other intramolecular vibrational modes. The fit for the ν_r mode also shows two exponential decays representing IVR relaxation and vibrational population relaxation to the solvent. Additionally, the ν_r mode displays an oscillation with a frequency of 33 cm^{-1}. Like seen in the FeRu/FA sample, this oscillation probably arises from contributions of both a ν_r/ν_t superposition, where the difference in frequency is 10 cm^{-1}, and a ν_r/ν_b superposition, where the difference in frequency is 67 cm^{-1}. A clear cross-peak can be seen between ν_r and ν_b in Figure 3.3c at $\omega_3 = \nu_r$ while the cross-peaks between ν_t and ν_r are not as easily distinguished due to overlap with the ν_t and ν_r diagonal peaks. The fit for the ν_a mode shows one exponential decay representing the vibrational lifetime, a rise, and a decaying oscillation. The rise of ~1.1 ps is attributed to energy relaxation from the other three ν_{CN} modes into the axial mode. A similar rise of ~800 fs is seen in the fit for the ν_b mode (2115 cm^{-1}) which also shows one exponential decay and a coherent oscillation. The oscillation frequency of ~54 cm^{-1} is approximately the split in frequency between the ν_t and ν_b modes indicating a superposition between the two modes like that seen in the bridge mode of FeRu/FA. In summary, we note that all four ν_{CN} modes of FePtFe/D_2O have similar vibrational lifetimes ranging from 16-19 ps. This is in contrast to FeRu where both samples had significantly shorter lifetimes for the trans mode. The lifetimes of the coherent oscillations found in the ν_r, ν_a, and ν_b modes of FePtFe are ~500 fs.

3.3.2 *2D IR Relaxation Experiment*

FeRu/FA. Figure 3.3 displays representative crossed polarization 2D IR plots as a function of τ_2 for all three samples. Note that contour levels do not follow a linear scale. The 2D IR data for FeRu in FA is shown in Figure 3.3a. There are three main features along the diagonal with positive (0→1 transitions) and negative (1→2 transitions) features. The spectral feature at lowest energy represents the ν_t mode, the feature in the middle of the spectrum represents overlapping contributions from the ν_r and the ν_a modes as well as their cross-peaks, and the one at highest energy represents the ν_b mode. It can be seen that the trans mode decays away relatively quickly compared to the other features. This is due to the faster vibrational relaxation time for the ν_t mode compared to the other ν_{CN} modes (see Table 3.1). The time-evolving 2D lineshapes and amplitudes measure the spectral diffusion and vibrational relaxation dynamics of the ν_{CN} modes in FeRu/FA. The 2D spectral feature of the ν_t mode has the largest anti-diagonal width and maintains its spectral elongation until after the time its amplitude has fully decayed ($\tau_2 \approx 5$ ps). The 2D peak representing both the radial and axial modes is elongated along the diagonal at τ_2 = 70 fs and its spectral diffusion dynamics are complete by ~5 ps. In contrast to the ν_r and ν_a diagonal peaks, the 2D peak of the bridge mode does not appear to have any elongation along the diagonal axis at early waiting times. The diagonal and cross-peaks are amplitude modulated as a function of τ_2 in a 2D IR spectrum of coupled vibrational modes. Oscillatory behavior is clearly seen for the ν_b mode in the 2D IR spectra where the diagonal peak has more amplitude at 300 fs than it does at 70 fs. The beats are also seen in the pump-probe data (Fig. 3.2a) which shows a clear oscillation at a frequency of 84 cm^{-1}. Recall that the dispersed pump-probe data at a particular τ_2 delay is the projection of the 2D IR spectrum along ω_3. Evidence of intramolecular vibrational energy redistribution among the ν_{CN} modes of FeRu/FA can be observed by the growth of three cross

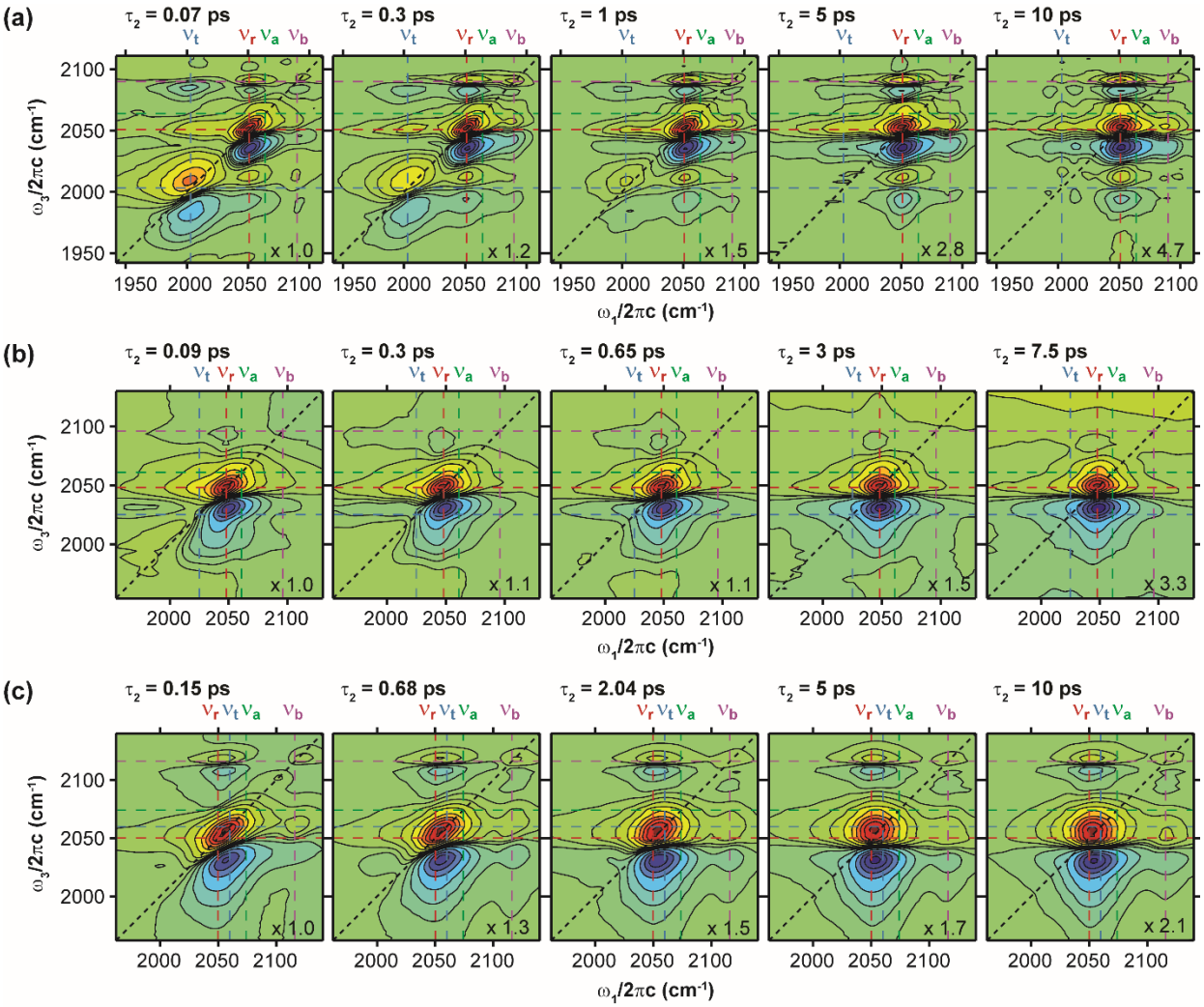


Figure 3.3: Crossed polarization 2D IR spectra at selected τ_2 delays for (a) FeRu in FA, (b) FeRu in D_2O, and (c) FePtFe. All spectra have been normalized to the highest point at each τ_2 delay, and the relative factors are shown in the bottom right corner. Each FeRu/FA spectrum is plotted using contour lines at the following positions: ±0.018, ±0.055, ±0.1, ±0.2, ±0.35, ±0.5, ±0.65, ±0.8, and ±0.95. Each FeRu/D_2O spectrum is plotted using contour lines at the following positions: ±0.01, ±0.04, ±0.1, ±0.2, ±0.35, ±0.5, ±0.65, ±0.8, and ±0.95. Each FePtFe spectrum is plotted using contour lines at the following positions: ±0.01, ±0.045, ±0.1, ±0.2, ±0.35, ±0.5, ±0.65, ±0.8, and ±0.95.

peaks at ($\omega_1 = \nu_t$, $\omega_3 = \nu_b$), ($\omega_1 = \nu_r$, $\omega_3 = \nu_b$) and ($\omega_1 = \nu_b$, $\omega_3 = \nu_r$ /ν_a) relative to the radial diagonal peak. The other two obvious cross-peaks at ($\omega_1 = \nu_r$, $\omega_3 = \nu_t$) and ($\omega_1 = \nu_t$, $\omega_3 = \nu_r$) appear to decay at similar rates to the ν_r and ν_a modes.

FeRu/D_2O. The 2D IR spectra for FeRu in D_2O are shown in Figure 3.3b and look very different from the 2D IR spectra of FeRu in FA. The FTIR spectrum of FeRu/D_2O shows significant overlap

of the ν_t, ν_r, and ν_a modes and very weak amplitude in the ν_b mode. The resultant 2D spectra of FeRu in D_2O show one main peak that represents overlapping contributions from the ν_r and ν_a modes as well as their cross peaks. At the lower left-hand corner of the central peak, there is a small feature that quickly disappears indicating the ν_t mode whose vibrational lifetime is believed to be ~400 fs (Table 3.1). A bridge diagonal peak is not distinguishable in any of the spectra due to the ν_b mode's very small transition dipole moment ($|\mu_r| = 0.89|\mu_t|$, $|\mu_a| = 0.40|\mu_t|$, and $|\mu_b| = 0.12|\mu_t|$) as diagonal peak amplitudes scale with μ_p^4. The central diagonal peak loses its spectral correlation and appears fully rounded by ~5 ps. We note that it is difficult to disentangle the relative homogenous and inhomogeneous contributions from the radial and axial cross peak contributions. There are two cross peaks that show up as wings to the lower and higher energy sides of the central peak ($\omega_1 = \nu_t/\nu_b$, $\omega_3 = \nu_r$). The 2D spectra reveal a spectral feature growing in at higher ω_3 frequencies for $\tau_2 > 3$ ps. We attribute this to a solvent feature as there is a large D_2O band centered at ~2500 cm^{-1}. Temperature dependent FTIR spectra of D_2O show a growth in the transmission from ~2100 to 2150 cm^{-1} as the temperature increases. In the 2D spectra, we see that the amplitude in the same spectral region increases.

FePtFe/D$_2$O. The 2D IR spectra for FePtFe are shown in Figure 3.3c. Here we see two main diagonal features, the central one representing overlapping contributions from the ν_r, ν_t, and ν_a modes as well as their respective cross peaks, and the higher energy peak representing the ν_b mode. The central feature starts off elongated along the diagonal with similar homogeneous width for all three modes (see 2D IR plot at $\tau_2 = 0.15$ ps) and the spectral diffusion dynamics are complete by 5 ps. Like seen in the 2D spectra for FeRu/FA, the 2D spectral feature of the bridge mode does not appear to have any elongation along the diagonal at early τ_2 delays. The diagonal bridge peak exhibits amplitude beating with a time-period of 620 fs consistent with the beats seen in the

dispersed IR pump-probe data (Fig. 3.2c). We see that the obvious cross peaks, ($\omega_1 \approx \nu_r$, $\omega_3 = \nu_b$) and ($\omega_1 = \nu_b$, $\omega_3 \approx \nu_t$) grow in relative to the diagonal features and are most visible in the $\tau_2 = 10$ ps 2D IR spectrum. Below we will discuss the spectral diffusion and vibrational relaxation dynamics measured in the 2D IR spectra.

3.3.2.1 *Spectral Diffusion*

Table 3.2: Exponential fit values to CLS traces. Error bars represent 90% confidence intervals.

	Mode	t_1 (ps) (A_1)	t_2 (ps) (A_2)	ω_{osc} (cm^{-1})[a]	A_∞
FeRu/FA	Trans	4±2 (0.31)			0.07±0.05
	Radial	2±1 (0.23)	20±10 (0.23)		
	Bridge	20±40 (0.21)	0.30 (0.3)	84	
FeRu/D_2O	Radial	3±1 (0.26)			0.09±0.03
	Axial	4±4 (0.09)	0.6±0.7 (0.12)		
FePtFe	Radial	3±3 (0.83)			0.1±0.3
	Axial	2±1 (0.19)			0.04±0.02
	Bridge	11±9 (0.16)	0.46 (0.7)	54	

[a]Oscillation frequencies are held constant at their respective pump-probe oscillation frequency.

Transition frequency fluctuations arise from the relative movements of molecules' positions and orientations. Vibrational correlation functions can be used to characterize these fluctuations, and as such, these correlation functions are a measure of the system-bath interaction for each vibration. This gives us a tool to look at solvation dynamics. Timescales for spectral diffusion, a measure of the frequency-frequency correlation function (FFCF), can vary greatly between similar systems. There are various metrics available to extract FFCF timescales, but here we use the center line

slope (CLS) method to measure spectral diffusion of the CN stretches.[14] CLS is one way to quantify how elongated the peak is along the diagonal at a particular τ_2 delay. To accomplish this, several ω_1 frequencies centered on the fundamental frequency of the vibrational mode are chosen, and the ω_3 frequency at which maximum amplitude is reached is recorded for each ω_1 frequency. These points determine a line on an ω_1 versus ω_3 plot, and the slope of that line is the CLS. This slope is calculated for each τ_2, and the resulting decay is fit to the following expression (results in Table 3.2) to find timescales that are proportional to the FFCF:

$$CLS(\tau_2;\omega_3) = \sum_i A_i \exp\left(\frac{-\tau_2}{t_i}\right) + A_{osc} \exp\left(\frac{-\tau_2}{t_{osc}}\right)\cos\left(2\pi c\omega_{osc}\tau_2 + \phi\right) + A_\infty \qquad (3.2)$$

where A and t are the amplitude and timescale of each decay respectively, ω_{osc} is the frequency of the oscillation, and ϕ is the phase of the oscillation. The term A_∞ represents a static component, dynamics that do not occur within the timescale of this experiment. The cosine is only included in the fit if there is an oscillatory component present, and the t_{osc} and ω_{osc} variables are held constant at the values determined in the IR pump-probe fits at that ω_3. Other oscillations that have appeared in peak shift measurements, another metric of the FFCF, have been attributed to underdamped hydrogen bond oscillation in HOD/D_2O,[15] anharmonic coupling of a high frequency NH stretching vibration to low frequency interdimer modes that modulate the NH hydrogen-bonding strength,[16] and coupling of the OH stretch to underdamped intermolecular coordinates HOD in D_2O.[17]

Figure 3.4 shows the CLS decays determined from the parallel polarization 2D IR spectra for selected modes from each system. The ν_{CN} modes for which CLS analysis was preformed were chosen carefully to minimize contributions from other close lying modes. Large error bars are due

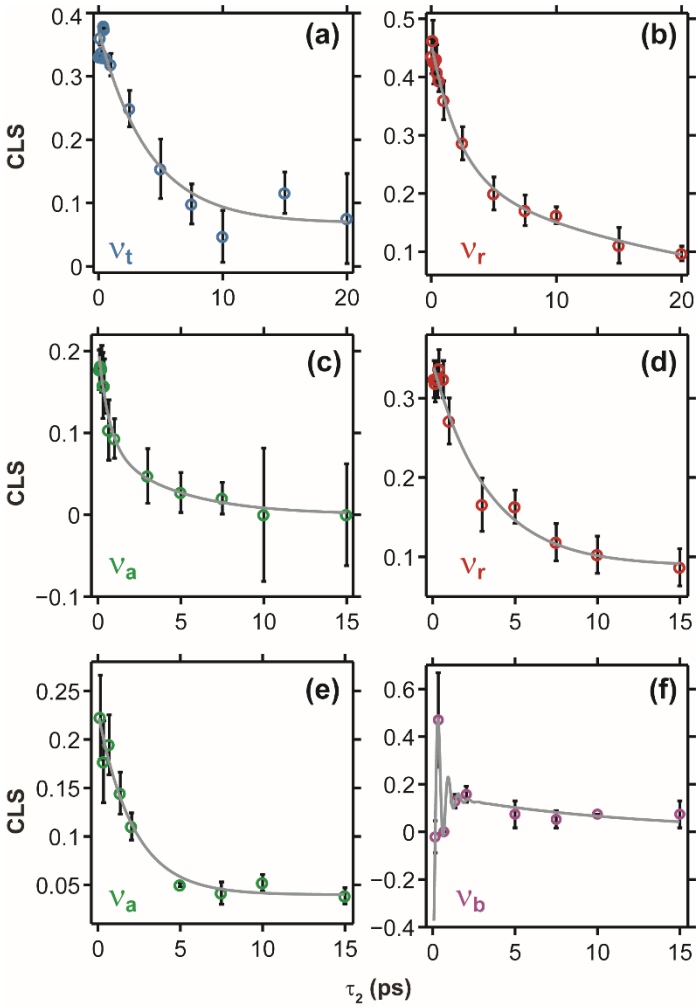


Figure 3.4: Plots of the center line slope (CLS) measuring spectral diffusion as a function of τ_2 for selected modes from each sample. Error bars represent 90% confidence intervals on the CLS value for each τ_2 point. (a) and (b) are the ν_t and ν_r modes, respectively, for FeRu/FA. (c) and (d) are the ν_a and ν_r modes, respectively, for FeRu/D_2O. (e) and (f) are the ν_a and ν_b modes, respectively, for FePtFe.

to the limited number of τ_2 points available for each sample. The CLS of the ν_r mode is the only one that can be compared between all three samples. The two CLS traces from the D_2O samples of both FeRu and FePtFe, have a short time constant around 3 ps and an offset that represents dynamics too long to be measured with the time resolution of this experiment. The FeRu/FA ν_r trace has two time constants and no offset. It still has a short time constant at 2 ps that is close to the D_2O samples, but its long time constant is ~20 ps. For the ν_a mode, only the two D_2O samples

can be compared. FeRu/D_2O has two timescales present in its fit: the shortest decay constant measured at ~600 fs and a longer one at ~4 ps that is close to the ~3 ps decays seen in the ν_r mode. The FePtFe ν_a trace has one time constant at ~2 ps and an offset. With the ν_b mode traces, we cannot compare across molecule or solvent due to low signal in the bridge region of the 2D spectra for FeRu/D_2O (Fig. 3.3b). The fits for the ν_b traces for the FeRu/FA and FePtFe samples reveal a matching decay time constant of ~20 ps fit to them while keeping the decay for the beat constant. This long decay also matches the long time constant seen in the ν_r mode of FeRu/FA. Only the trans mode of the FeRu/FA sample was fully distinguishable from the other modes due to a lack of overlapping spectral features. This only allowed a determination of a CLS trace for one ν_t mode. Its one decay constant of ~4 ps is close to the ~3 ps time constant we see in all CLS fits that do not contain an oscillation.

3.3.2.2 *Mode-Specific Vibrational Relaxation Dynamics*

To look at the dynamics of specific modes, a small area of each 2D spectrum was chosen for each peak and all signal intensity within that area was summed. These volumes were then plotted as a function of τ_2 to see peak specific relaxation. Each of these decays was then fit to a sum of exponentials like the formula used to fit the pump-probe traces. Because of the low number of τ_2 points available, these fits have very large uncertainties and only a few fits are shown in Figure 3.5 for illustration. These fits are discussed in a qualitative manner below.

Several illuminating time-dependent cross-peak volume plots for FeRu in FA show growths in addition to their decays (Fig. 3.4, left panel). These rises indicate population transfer between modes on the order of one picosecond. Most of these cross-peaks also have multiple decay timescales with the longest timescales on the order of the vibrational lifetimes of the ν_{CN} modes. The mode-specific dynamics of FeRu in D_2O do not contain as many growths as the FA sample,

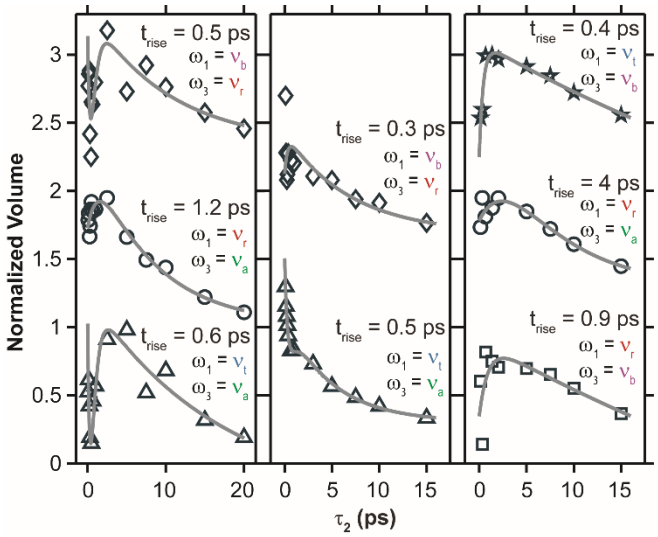


Figure 3.5: The summed signal intensity over a small area of each 2D IR spectrum centered on the peaks indicated by the ω_1 and ω_3 frequencies is shown for each trace for the FeRu/FA samples (left), FeRu/D_2O sample (middle), and FePtFe sample (right) as a function of τ_2 delay.

but the ones that are observed are also on ~1 ps timescales. Data for FePtFe is similar to that seen for FeRu in FA with multiple rises on the order of 1 ps, several decays, and long timescales on the order of the vibrational lifetimes for the ν_{CN} modes. All of the short decay timescales seen in the pump-probe match the ~1 ps rises seen in the 2D volume mode-specific relaxation dynamics giving more strength to the argument that IVR dynamics are happening in all three systems on this timescale.

3.4 Discussion

Our results show the vibrational relaxation in the dephasing dynamics are mode dependent in the solvated molecules. Vibrational lifetimes change significantly between modes within the same system and between the same modes in different systems. Long spectral diffusion timescales also show large variations between the same mode in different systems and different modes within the same system. In this section, we will put this work in the context of other work done with CN stretches.

3.4.1 *Vibrational Relaxation Comparison*

The IR pump-probe experiments reveal that the vibrational lifetimes of the radial and axial modes are ~2 times faster when FeRu is dissolved in D_2O versus FA. This is likely due to the much larger overlap between the CN and D_2O IR bands than exists between the CN and FA bands. The D_2O band centered at ~2500 cm^{-1} is much broader (~500 cm^{-1}) and more intense than the closest FA band (centered at ~2210 cm^{-1} with a width of ~80 cm^{-1}) to the CN stretches. This is particularly obvious in the 2D spectra for the FeRu/D_2O sample where a large solvent response grows in at high energies along the ω_3 axis. A greater overlap would lead to more efficient energy transfer between the solute and solvent resulting in faster relaxation times. The vibrational lifetime of the trans mode is also likely faster when FeRu is dissolved in D_2O versus FA, but it is hard to say by how much. A faster relaxation time for the ν_t mode in D_2O versus FA was also observed in a similar molecule, $[(NH_3)_5Ru^{III}NCOs^{II}(CN)_5]^-$ after optical excitation and back electron transfer.[4] This was attributed to better overlap between the ν_t band and solvent bands in D_2O compared with FA. Shorter vibrational lifetimes for the ν_t mode in FeRu indicate the ν_t mode has stronger interactions with the solvent than the other three modes. Because the ν_t mode is on the end of the molecule, it is more exposed to the solvent than the other modes.

Due to the larger size and increased complexity of FePtFe, it is expected that the vibrational relaxation for the ν_{CN} modes would be faster than for FeRu. Greater complexity increases the number of IVR channels available for the ν_{CN} modes to dump their energy into leading to more efficient energy transport out of the ν_{CN} modes. FePtFe also has a larger charge than FeRu. Previous work with other ions in solution has shown that they relax more quickly with increasing charge, likely due to the increased strength of interaction with the solvent.[18] In addition to these molecular attributes, most of the FePtFe ν_{CN} modes have better overlap with D_2O IR bands than

the ν_{CN} modes of FeRu. Again, this should increase the efficiency with which energy is funneled out of the ν_{CN} modes and into the solvent. Given these factors, it is surprising to see that the vibrational lifetimes for most FePtFe ν_{CN} modes are almost twice as long as the lifetimes for FeRu/FA and four times as long as the lifetimes for FeRu/D_2O. The ν_t mode certainly decays significantly slower in FePtFe than in either FeRu sample. The fact that the ν_t mode in FePtFe does not have a shorter lifetime than the other modes would indicate it is not as strongly coupled to the solvent as it is in FeRu. Weaker coupling of the ν_{CN} modes to the solvent could also explain the longer timescales seen in FePtFe in general. The larger charge on the Pt(IV) compared to the Ru(III) explains why the frequencies of the trans and bridge modes are farther to the blue for FePtFe than for FeRu. The larger positive charge pulls more electron density out of the cyanide antibonding orbital increasing the strength of that bond and increasing its vibrational frequency. This effect might also explain why the trans mode acts so differently in FePtFe than for the two FeRu systems. If the reduced metal-ligand interaction results in weaker coupling to other carbon-metal modes or other low frequency modes along the MMCT backbone, this would limit interactions with the solvent for FePtFe in general helping explain why the lifetimes observed are so much longer than those measured for the FeRu systems.

Vibrational lifetimes ranging from 2.3 to 38.2 ps were observed for OCN^-, SCN^-, and $SeCN^-$ in D_2O, a lifetime of 170 ps was measured for the ν_{CN} mode in $FeNO(CN)_5^{2-}$, and a lifetime of 4.2 ps was observed for a cyanide probe of the villin HP35 protein.[18-20] Other simple metal-cyanide compounds have long vibrational lifetimes that range from 8.0 to 170 ps when dissolved in D_2O.[21] All of the lifetimes observed in the current study fall within this range with the possible exception of the ~400 fs lifetime measured for the trans mode in FeRu/D_2O. Vibrational lifetimes for ν_{CN} modes in FePtFe, all of which are ~18 ps, closely match the lifetime of 24 ps reported for

$Fe^{II}(CN)_6^{4-}$ in D_2O.[19] However, the lifetimes for FeRu in D_2O and FA of ~6 ps and ~10 ps, respectively, much more closely match the lifetimes reported for $Fe^{III}(CN)_6^{3-}$ in D_2O and FA of 8 ps and 10 ps, respectively.[19] This comparison would indicate that the oxidation state of the iron in FeRu might be +3 instead of the +2 reported or at least more delocalized than assumed. However, work by multiple groups has shown that the extra electron is fairly localized on the Ru and not on the Fe.[22,23] In a study of single metal-cyanides of the form $M(CN)_x^{y-}$ dissolved in D_2O, it was found that the vibrational lifetimes for the ν_{CN} mode was split into two camps where the first group have long vibrational energy relaxation times of >100 ps, and the second group have times of <35 ps. The first group contains $Pt(CN)_4^{2-}$, a square planar molecule, while the second includes $Ru(CN)_6^{4-}$, $Fe(CN)_6^{2-}$, and $Fe(CN)_6^{3-}$, all octahedral molecules.[21] While the oxidation state and molecular configuration of the platinum investigated in this study is different than the state of the platinum in FePtFe, the large discrepancy in relaxation rates between the Pt and the Fe containing species despite the species similar charges might provide a clue as to why we observe longer lifetimes for the ν_{CN} modes in FePtFe than in the ν_{CN} modes in FeRu.

3.4.2 *Spectral Diffusion Comparison*

Due to the limited number of points available to fit and the large error bars that result for the CLS timescales measured, the comparisons that can be made between the samples studied here and other systems are limited. The approximately 3 ps timescale seen repeatedly in the CLS measurements can be compared to similar timescales of 1-2 ps seen in the frequency-frequency correlation functions for several compounds: a 1.5 ps spectral diffusion constant for the ν_{CN} mode in $Fe^{II}(CN)_6^{4-}$ in D_2O,[24] a ~2 ps timescale for ν_{NO} mode $FeNO(CN)_5^{2-}$ in several hydrogen bonding solvents,[25] and a 1.6 ps timescale for the cyanide probe of a villin HP35 protein when water molecules were allowed into the pocket.[20] In all of these cases, the timescale was compared to the

characteristic timescale for the forming and breaking of hydrogen bonds.[17,26,27] In light of this, the lack of a ~3 ps timescale for the bridge mode CLS is not surprising. The bridging cyanide is not capable of forming hydrogen bonds because the nitrogen is already bound to the ruthenium instead of being free to interact to with the solvent like it is for the other modes. Comparing the ν_r traces between all three samples, we see there is a dependence on solvent in that both D_2O samples only have one decay and an offset whereas the FA sample has a similar short decay and a longer decay with no offset. We have previously shown that angles between transition dipole moments might imply that the D_2O samples are bent in solution, while the FA sample is linear.[7] This would suggest that, at least for this mode, spectral diffusion rates could rely on the structure of the molecule. But we can see a difference between the molecules by comparing the ν_a traces between FeRu and FePtFe in D_2O. The FePtFe sample only shows the ~1 ps decay and an offset. In contrast, the FeRu sample shows a shorter timescale, the ~1 ps decay, and no offset.

3.5 Conclusion

Despite previous work showing correlations between solvent and molecular parameters completed by our group, this study fails to find a correlation between the solvent and electronic ground state vibrational relaxation dynamics of the cyanide stretching vibrations of bimetallic and trimetallic transition metal mixed-valence complexes. We measure vibrational lifetimes, IVR dynamics, and spectral diffusion dynamics using infrared pump-probe and two-dimensional infrared spectroscopies. The values for vibration lifetimes reveal the ~18 ps lifetimes for the ν_{CN} modes of FePtFe dissolved in D_2O are approximately four times longer than the modes of FeRu dissolved in D_2O and two times longer than the modes in FeRu dissolved in FA. The difference in lifetimes between the two D_2O systems indicates that vibrational lifetimes are not strongly dependent on solvent, higher molecular complexity, or larger overall charge of the molecule all of

which would result in faster relaxation times for FePtFe. Vibrational lifetimes for the trans mode in both FeRu system are significantly faster than the other three modes, which is not the case for FePtFe. This is an indication that FePtFe has weaker solute-solvent interactions since the trans mode is the most accessible to the solvent. Mode specific dynamics from both the pump-probe and 2D IR experiments reveal that IVR is occurring in all three systems on the order of 1 ps. This confirms previous findings that the ν_{CN} modes are coupled, but again show no obvious solvent dependence. Spectral diffusion dynamics reveal that the radial, axial, and trans ν_{CN} modes show a ~3 ps timescale that is indicative of the forming and breaking of hydrogen bonds between the solvent and each ν_{CN} mode that is not apparent in the bridge mode. The lack of a relationship between solvent and vibrational dynamics timescales in these systems points to a need to consider the specific metals used in the backbone of future cyanide-bridged transition metal systems, not just the ligands or the solvent used.

3.6 Additional 2D IR Spectra for FeRu

As mentioned in Section 3.2, 2D IR spectra were collected at more τ_2 points than are shown in Figure 3.3. The additional spectra for FeRu in both FA and D2O are shown below in Figures 3.6 - 3.9. Each figure shows all time points collected for each sample in either the parallel (ZZZZ) or crossed (YYZZ) polarizations. They are plotted on the same respective contour levels as listed in the caption for Figure 3.3.

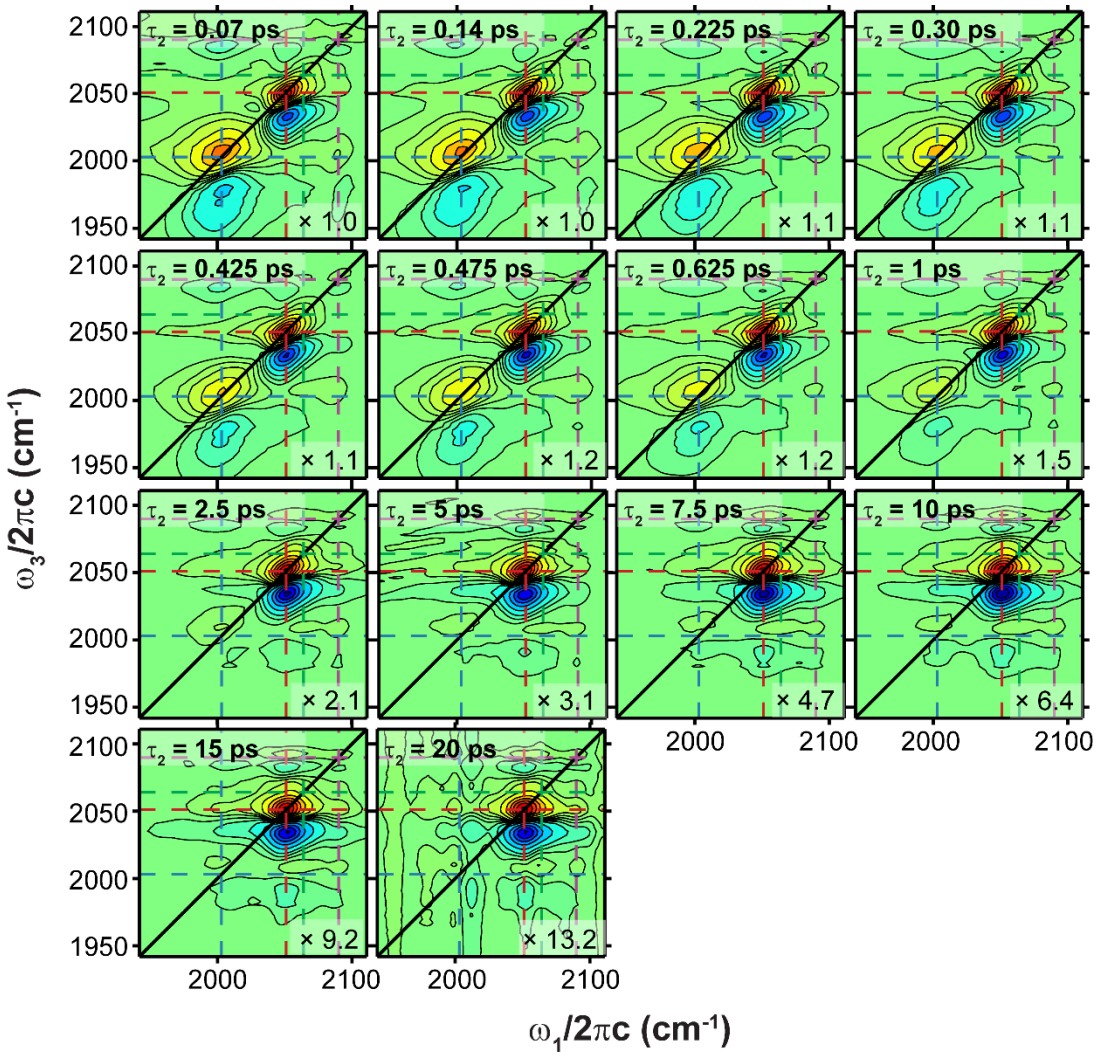


Figure 3.6: Parallel polarization 2D IR spectra for FeRu dissolved in FA for all time points collected. Each spectra has been normalized, and the relative amplitudes of each spectra are shown in the lower right corner. The frequencies of each ν_{CN} mode are shown by dashed lines and are color-coded to the modes shown in Figure 3.1.

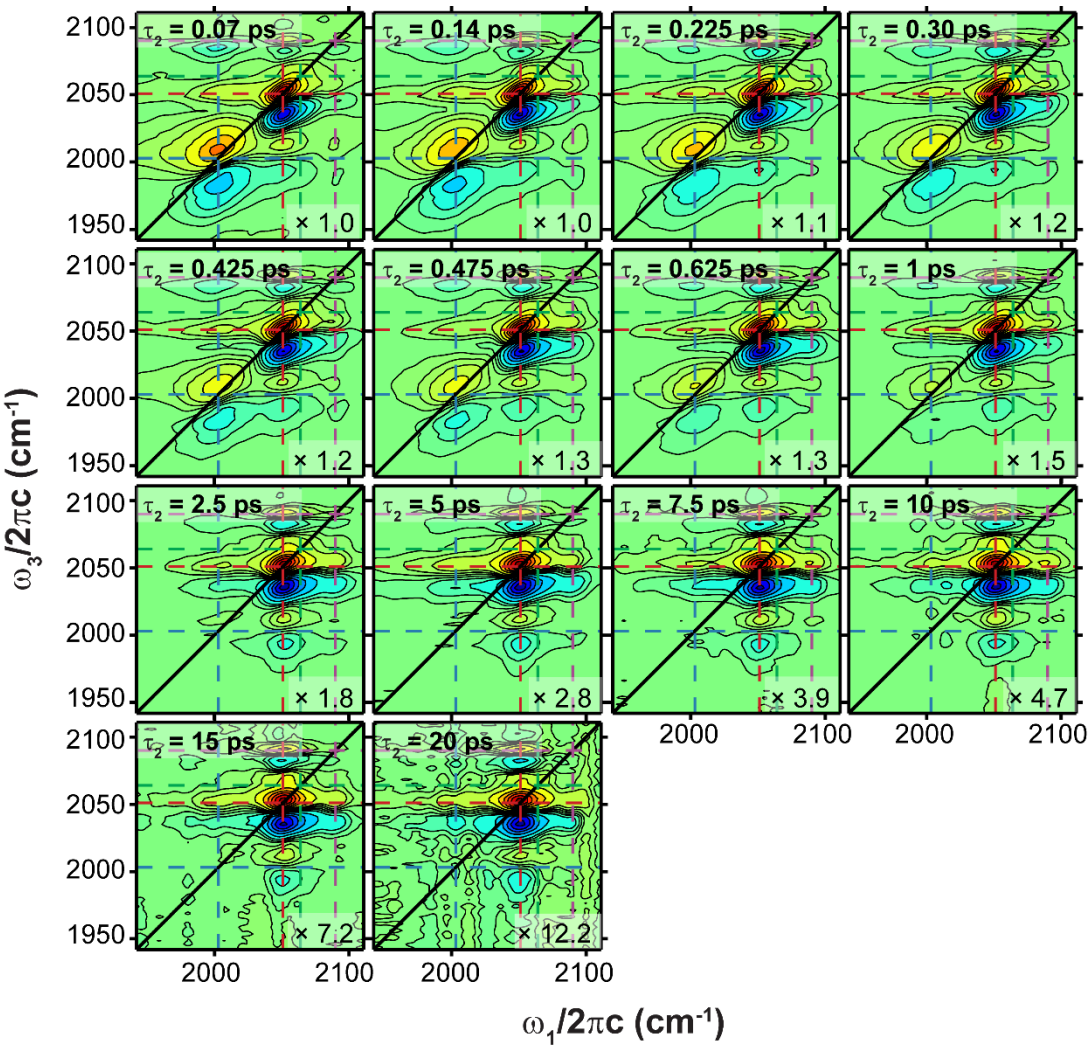


Figure 3.7: Crossed polarization 2D IR spectra for FeRu dissolved in FA for all time points collected. Each spectra has been normalized, and the relative amplitudes of each spectra are shown in the lower right corner. The frequencies of each ν_{CN} mode are shown by dashed lines and are color-coded to the modes shown in Figure 3.1.

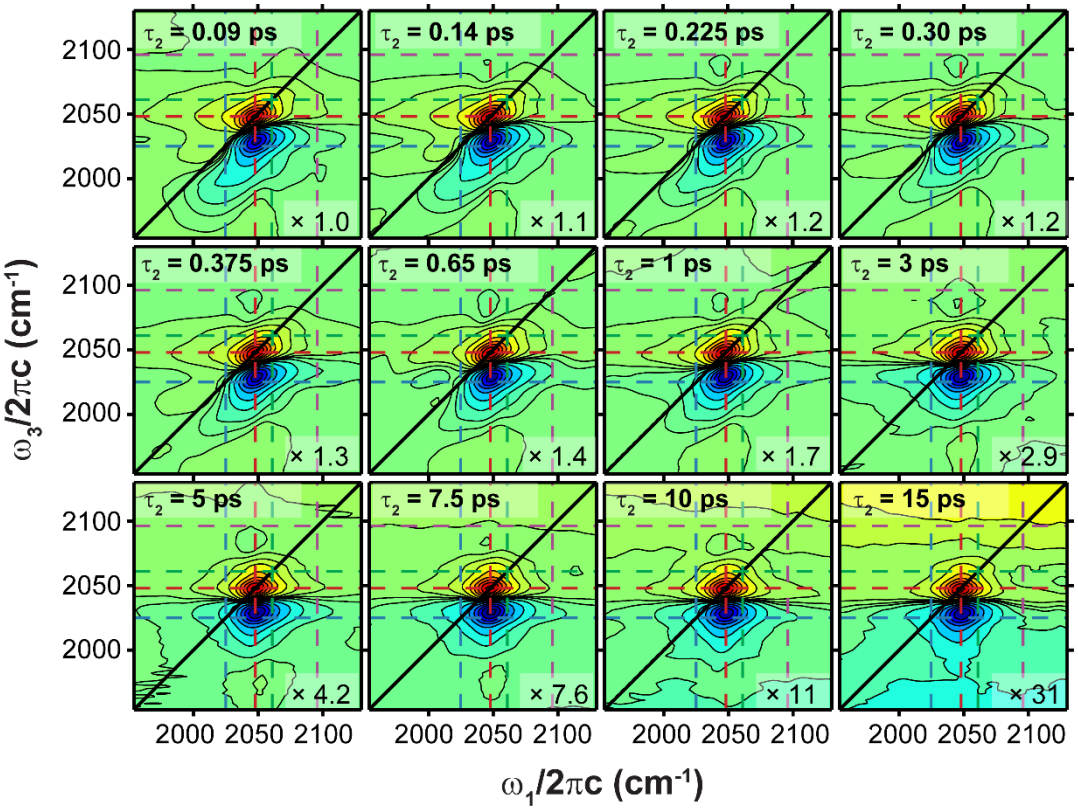


Figure 3.8: Parallel polarization 2D IR spectra for FeRu dissolved in D_2O for all time points collected. Each spectra has been normalized, and the relative amplitudes of each spectra are shown in the lower right corner. The frequencies of each ν_{CN} mode are shown by dashed lines and are color-coded to the modes shown in Figure 3.1.

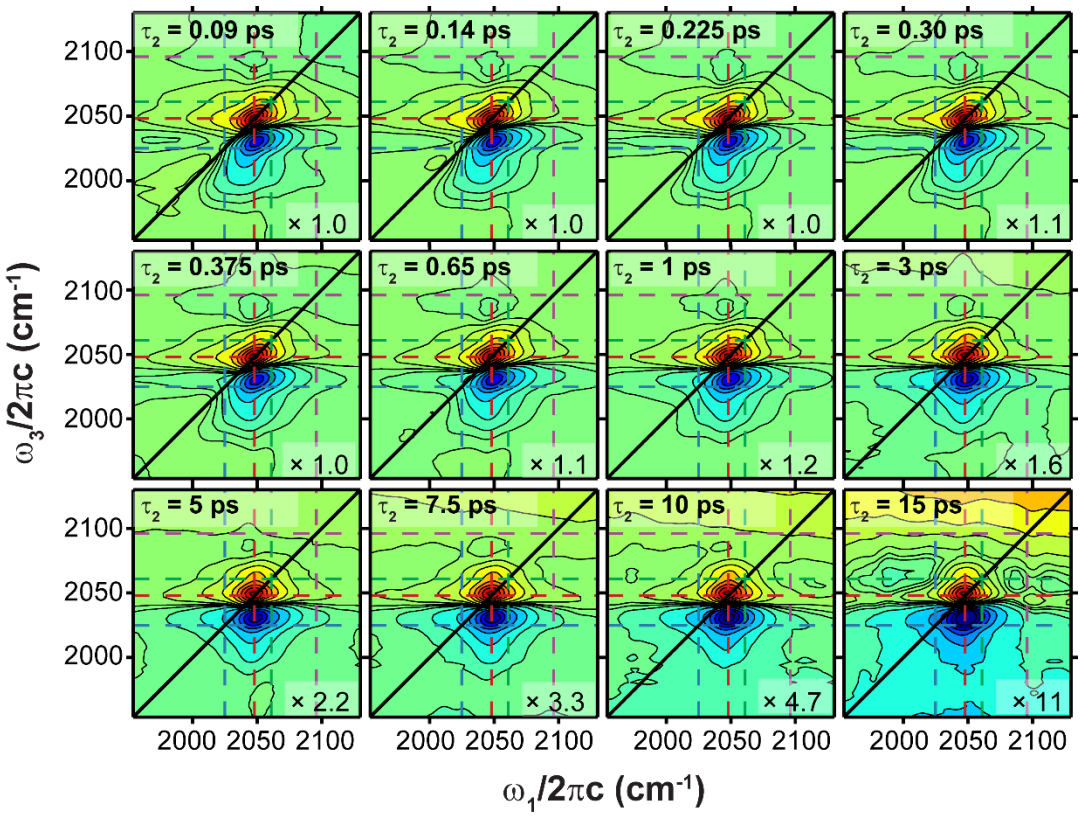


Figure 3.9: Crossed polarization 2D IR spectra for FeRu dissolved in D_2O for all time points collected. Each spectra has been normalized, and the relative amplitudes of each spectra are shown in the lower right corner. The frequencies of each ν_{CN} mode are shown by dashed lines and are color-coded to the modes shown in Figure 3.1.

REFERENCES

[1] Lynch, M. S.; Slenkamp, K. M.; Cheng, M.; Khalil, M. *J. Phys. Chem. A* **2012**, *116*, 7023.

[2] Lynch, M. S.; Slenkamp, K. M.; Khalil, M. *J. Chem. Phys.* **2012**, *136*, 241101.

[3] Lynch, M. S.; Van Kuiken, B. E.; Daifuku, S. L.; Khalil, M. *J. Phys. Chem. Lett.* **2011**, *2*, 2252.

[4] Tivanski, A. V.; Wang, C.; Walker, G. C. *J. Phys. Chem. A* **2003**, *107*, 9051.

[5] Wang, C.; Mohney, B. K.; Akhremitchev, B. B.; Walker, G. C. *J. Phys. Chem. A* **2000**, *104*, 4314.

[6] Doorn, S. K.; Dyer, R. B.; Stoutland, P. O.; Woodruff, W. H. *J. Am. Chemical. Soc.* **1993**, *115*.

[7] Slenkamp, K. M.; Lynch, M. S.; Van Kuiken, B. E.; Brookes, J. F.; Bannan, C. C.; Daifuku, S. L.; Khalil, M. *J. Chem. Phys.* **2014**, *140*, 084505.

[8] Hester, R. E.; Nour, E. M. *J. Chem. Soc., Dalton Trans* **1981**, 939.

[9] Swanson, B. I. *Inorg. Chem.* **1976**, *15*, 253.

[10] Swanson, B. I.; Rafalko, J. J. *Inorg. Chem.* **1976**, *15*, 249.

[11] Watson, D. F.; Tan, H. S.; Schreiber, E.; Mordas, C. J.; Bocarsly, A. B. *J. Phys. Chem. A* **2004**, *108*, 3261.

[12] Vogler, A.; Kisslinger, J. *J. Am. Chem. Soc.* **1982**, *104*, 2311.

[13] Vogler, A.; Osman, A. H.; Kunkely, H. *Inorg. Chem.* **1987**, *26*, 2337.

[14] Kwak, K.; Rosenfeld, D. E.; Fayer, M. D. *J. Chem. Phys.* **2008**, *128*, 204505.

[15] Roberts, S. T.; Ramasesha, K.; Petersen, P. B.; Mandal, A.; Tokmakoff, A. *J. Phys. Chem. A* **2011**, *115*, 3957.

[16] Petersen, P. B.; Roberts, S. T.; Ramasesha, K.; Nocera, D. G.; Tokmakoff, A. *J. Phys. Chem. B* **2008**, *112*, 13167.

[17] Fecko, C. J.; Eaves, J. D.; Loparo, J. J.; Tokmakoff, A.; Geissler, P. L. *Science* **2003**, *301*, 1698.

[18] Lenchenkov, V.; She, C.; Lian, T. *J. Phys. Chem. B* **2006**, *110*, 19990.

[19] Sando, G. M.; Zhong, Q.; Owrutsky, J. C. *J. Chem. Phys.* **2004**, *121*, 2158.

[20] Urbanek, D. C.; Vorobyev, D. Y.; Serrano, A. L.; Gai, F.; Hochstrasser, R. M. *J. Phys. Chem. Lett.* **2010**, *1*, 3311.

[21] Weidinger, D.; Sando, G. M.; Owrutsky, J. C. *Chem. Phys. Lett.* **2010**, *489*, 169.

[22] Burewicz, A.; Haim, A. *Inorg. Chem.* **1988**, *27*, 1611.

[23] Van Kuiken, B. E.; Valiev, M.; Daifuku, S. L.; Bannan, C.; Strader, M. L.; Cho, H.; Huse, N.; Schoenlein, R. W.; Govind, N.; Khalil, M. *J. Phys. Chem. A* **2013**.

[24] Ohta, K.; Maekawa, H.; Tominaga, K. *J. Phys. Chem. A* **2004**, *108*, 1333.

[25] Brookes, J. F.; Slenkamp, K. M.; Lynch, M. S.; Khalil, M. *J. Phys. Chem. A* **2013**, *117*, 6234.

[26] Asbury, J. B.; Steinel, T.; Kwak, K.; Corcelli, S. A.; Lawrence, C. P.; Skinner, J. L.; Fayer, M. D. *J. Chem. Phys.* **2004**, *121*, 12431.

[27] Shevchuk, R.; Agmon, N.; Rao, F. *J. Chem. Phys.* **2014**, *140*, 244502.

Chapter 4:

Synthesis and Characterization of Ruthenium (II) Polypyridine Cyanide Complexes

4.1 INTRODUCTION

Ruthenium containing complexes have been widely studied for applications in catalysis and photovoltaics with regards to energy production.[1-7] Usually these processes require some sort of charge transfer. While the ruthenium containing dimer (FeRu) discussed in Chapters 2 and 3 was used to study metal-to-metal charge transfer (MMCT) reactions, ruthenium polypyridyl derivatives have been investigated for their metal-to-ligand charge transfer (MLCT) reactions. Here, the electron is transferred from the ruthenium metal to the polypyridine ligand. One example shows a series of polypyridyl ruthenium (II) complexes containing 2,2'-bipyridine, 2,2'-bipyrazin or 2,2'-bipyrimidine as ligands where the electron transfer from the metal to the lowest lying π^* orbital is found to be coupled to a proton transfer.[1,4] Complexes like these are useful as models for several systems including the production of oxygen in photosystem II or for use in dye-sensitized solar cells to name a few.[2,3]

Studying these MLCT reactions and how they are coupled to nuclear motions can be accomplished using nonlinear vibrational spectroscopy like was discussed in Chapter 1 of this dissertation. Previous studies have shown that high frequency vibrations are highly coupled to back electron transfer reactions in cyanide-bridged mixed valence systems.[7-13] Just as the cyanide stretching vibrations were used as probes of the MMCT reactions in FeRu and FePtFe, high

frequency cyanide stretches could be used as probes of the MLCT reactions occurring in polypyridyl ruthenium complexes. Moving charges around the molecule can change its geometry as it comes to a new equilibrium, and the energy needed to move the electron has to leave the molecule somehow. Both of these processes will have an effect on molecular vibrations which can be monitored spectroscopically. Because each of these systems has fewer cyanide ligands than FeRu and FePtFe, it will be easier to interpret those vibrational spectra.

In this chapter, the syntheses of three polypyridyl ruthenium (II) complexes with cyanide ligands are described, and the complexes are characterized using ^{1}H NMR, FTIR, and UV/vis spectroscopies. These complexes are $[Ru^{II}(tpy)(bpy)CN]^+$ (monomer), $[\{Ru^{II}(tpy)(bpy)\}_2CN]^{3+}$ (dimer), and $[Ru^{II}\{CNRu^{II}(tpy)(bpy)\}_2]^{4+}$ (trimer), where 'tpy' stands for 2,2':6',2"-terpyridine and 'bpy' is 2,2'-bipyridine. Section 4.2 includes procedures used to synthesize each complex as well as sample preparation used to obtain spectra. Section 4.3 contains spectroscopic evidence used for product verification, and Section 4.4 looks at FTIR and UV/vis spectra of those products in several solvents. The final section, Section 4.5, includes a summary of the work done to this point with these complexes and what work we wish to do in the future.

4.2 METHODS

4.2.1 *Spectra Collection*

Most ^{1}H NMR spectra were collected in methanol-d_4 using an AV-300 300.13 MHz instrument. The trimer NMR spectra was collected in acetonitrile-d_3 using the same instrument. FTIR spectra were collected using a JASCO FT/IR-4100 with a spectral resolution of 0.5 cm^{-1}, and each final spectrum is an average of 64 spectra. Samples were placed in a cell between two BaF_2 windows separated by a 100 μm spacer. UV/vis spectra were collected at the same time as the FTIR spectra in the same sample cell using a JASCO V-630 with a spectral resolution of 0.2 nm.

4.2.2 *Synthesis*

Chemicals purchased were ruthenium(III) chloride hydrate (Sigma, Pressure Chemical), 2,2':6'2"-terpyridine (Sigma, Strem), 2,2'-bipyridyl (Sigma), potassium cyanide (Sigma), silver trifluoromethanesulfonate (AgOTf) 99% (Sigma), triethyl amine (Fisher), cis-dichlorobis(2,2'-bipyridine)ruthenium(II) hydrate (Strem) , ethanol (Decon Lab), diethyl ether (Sigma), lithium chloride (Sigma), methanol (Sigma), LiOTf (Sigma).

The syntheses of the $Ru(tpy)Cl_3$, $[Ru(tpy)Cl]_2Cl_2$, [Ru(tpy)(bpy)Cl]Cl and [Ru(tpy)(bpy)OTf]OTf were based off previously documented procedures with slightly different ligands.[2,6] These reactions typically resulted in yields between 60-90%. The synthesis of [Ru(tpy)(bpy)CN]OTf and $[\{Ru(tpy)(bpy)\}_2CN]OTf$ were procedures developed for the purposes of these experiments. The typical yields were in the range of 70-90%.

Synthesis of $Ru(tpy)Cl_3$.

$$RuCl_3 \cdot xH_2O + tpy \longrightarrow Ru(tpy)Cl_3$$

$RuCl_3 \cdot xH_2O$ (2.537 g, 12.2 mmol), 2,2':6',2"-terpyridine (tpy) (2.660 g, 11.4 mmol) and ethanol (150 mL) were charged to a 250 mL round bottom flask and brought to reflux under argon atmosphere. The reaction remained at reflux for 4 hours. The solution was allowed to cool to room temperature, and then the solution was filtered using a medium fritted funnel. The brown-orange solid was washed with ethanol until the wash appeared clear, and the solid was allowed to dry for several minutes. Then the solid was washed with ether for further drying, the solid $Ru(tpy)Cl_3$ product was collected (4.659 g, 10.6 mmol). The yield was ~92%.

Synthesis of [Ru(tpy)Cl]$_2$Cl$_2$.

$$Ru(tpy)Cl_3 \xrightarrow[N(CH_2CH_3)_3]{Reflux} [Ru(tpy)Cl]_2 Cl_2$$

$Ru(tpy)Cl_3$ (2.237 g, 5.07 mmol) was mixed with ethanol (120 mL) and triethyl amine (4 mL) in a 250 mL round bottom flask. The solution was flushed with argon and was left under argon for the remainder of the reaction. The solution was brought to reflux and left to reflux for 2 hours. The solution turned from a brown color to a dark brown-purple color. The product was cooled to room temperature and filtered. A dark purple solid was collected and washed with ethanol until the filtrate appeared clear. The product was then dried over vacuum and washed with ether to remove any remaining ethanol. The product was left to dry over vacuum for about 5 minutes. The purple solid was collected and stored (1.464 g, 1.89 mmol). The yield was ~75%.

Synthesis of [Ru(tpy)(bpy)Cl]Cl.

$$[Ru(tpy)Cl]_2 Cl_2 + 2(bpy) \xrightarrow{Reflux} [Ru(tpy)(bpy)Cl]Cl$$

$[Ru(tpy)Cl]_2Cl_2$ (1.2341 g, 1.59 mmol) was mixed with two molar equivalents of 2,2'-bipyridine (bpy) (0.5098 g, 3.26 mmol) in a 2:1 ethanol/H_2O mixture (60 mL). The solution was brought to reflux under a nitrogen atmosphere and left at reflux for 4 hours. Once the reaction was complete the solution turned a dark red. A saturated solution of LiCl was injected into the solution through a septum with a syringe and the solution was left to reflux for an additional 10 minutes. The solution was then filtered hot using gravity filtration to catch any unreacted material. The filtrate was collected and allowed to cool overnight. The ethanol was removed by rotary evaporation. The remaining solid was submerged in ether and scraped out of the rotary flask then filtered over vacuum. The dark brown-red solid product was collected (1.0679 g, 1.90 mmol) and a ^{1}H NMR was taken for verification of the product. The yield was ~60%.

Synthesis of [Ru(tpy)(bpy)OTf]OTf.

$$[\mathrm{Ru(tpy)(bpy)Cl}]\mathrm{Cl} + 2\mathrm{AgOTf} \xrightarrow[45\ ^\circ\mathrm{C}]{\text{methanol}} [\mathrm{Ru(tpy)(bpy)OTf}]\mathrm{OTf}$$

[Ru(tpy)(bpy)Cl]Cl and 2.05 molar equivalents of AgOTf (the slight excess was used to help drive the reaction) were dissolved in methanol and allowed to stir with slight heating (oil bath temperature was set to 45 ºC) overnight. The solution was removed from heat and allowed to cool back to room temperature. The solution was filtered over Celite and the AgCl precipitate was separated from the solution. The methanol was removed from the filtrate by rotary evaporation, and the precipitate was washed with ether. The red-brown solid was collected. The reaction resulted in a complete conversion.

Synthesis of [Ru(tpy)(bpy)CN]OTf.

$$[\mathrm{Ru(tpy)(bpy)OTf}]\mathrm{OTf} + 3\mathrm{KCN} \xrightarrow{\text{Reflux}} [\mathrm{Ru(tpy)(bpy)CN}]\mathrm{OTf}$$

[Ru(tpy)(bpy)OTf]OTf (0.7337 g, 0.930 mmol) was mixed with 3 molar equivalents of KCN in a 2:1 ethanol/H_2O mixture (30 mL) in a 100 mL round bottom flask. The KCN excess was used to help drive the reaction and prevent any of the dimer from forming. The solution was flushed with nitrogen and remained under nitrogen for the entirety of the reflux reaction. The solution was heated to reflux and the reaction proceeded overnight. The now vibrant wine-red colored solution was injected with a concentrated solution of LiOTf and allowed to continue to reflux for another 10 minutes. The solution was cooled and the ethanol was removed by rotary evaporation. The remaining liquid solution was cooled in a refrigerator overnight and then filtered and washed with a small amount of water and ethanol. The bright red solid was dried over vacuum for several minutes before being washed with ether. The solid was collected (0.6038 g, 0.899 mmol) and verified as the desired product by ^{1}H NMR and FTIR. The yield for this reaction was ~96%.

Synthesis of [{Ru(tpy)(bpy)}$_2$CN]OTf$_3$.

$$\left[\mathrm{Ru(tpy)(bpy)OTf}\right]\mathrm{OTf} + \tfrac{1}{2}\,\mathrm{KCN} \xrightarrow[\mathrm{LiOTf}]{\mathrm{Reflux}} \left[\left\{\mathrm{Ru(tpy)(bpy)}\right\}_2 \mathrm{CN}\right]\mathrm{OTf}_3$$

[Ru(tpy)(bpy)OTf]OTf (0.7664 g, 0.972 mmol) was mixed with 1/2 molar equivalent of KCN in a 2:1 ethanol/H_2O mixture (30 mL) in a 100 mL round bottom flask. The solution was flushed with nitrogen and remained under nitrogen for the entirety of the reflux reaction. The solution was heated to reflux and the reaction proceeded overnight. The now dark red colored solution was injected with a solution of LiOTf and allowed to continue to reflux for another 10 minutes. The solution was cooled and the ethanol was removed by rotary evaporation. The remaining liquid solution was cooled in a refrigerator overnight and filtered and washed with a small amount of water and ethanol. Then the dark brown-red solid was dried over vacuum for several minutes before being washed with ether. The solid was collected (0.6090 g, 0.419 mmol) and verified using ^{1}H NMR and FTIR analysis. The reaction resulted in a yield of ~86%.

Synthesis of (bpy)$_2$Ru(CN)$_2$.

$$\left(\mathrm{bpy}\right)_2 \mathrm{RuCl}_2 + 2\mathrm{AgOTf} \xrightarrow{50\ ^\circ\mathrm{C}} \left(\mathrm{bpy}\right)_2 \mathrm{RuOTf}_2 + \mathrm{AgCl}$$
$$\left(\mathrm{bpy}\right)_2 \mathrm{RuOTf}_2 + 4\mathrm{KCN} \xrightarrow{\mathrm{Reflux}} \left(\mathrm{bpy}\right)_2 \mathrm{Ru(CN)}_2$$

(bpy)$_2$RuCl$_2$ (0.5839 g, 1.12 mmol) was mixed with a 2.05 molar equivalent of AgOTf in a 2:1 ethanol/H_2O mixture in a round bottom flask. The reaction mixture was heated slightly (~50 °C) and stirred under a nitrogen atmosphere in the dark for ~2 hours. The solution was then filtered over Celite and rinsed with methanol to remove the solid AgCl. The filtered solution was then poured into a clean round bottom flask and 5 molar equivalents of KCN (assuming complete conversion of (bpy)$_2$RuCl$_2$ to (bpy)$_2$Ru(OTf)$_2$) were added. Under nitrogen atmosphere, bring the solution to reflux (~75 °C) and let the reaction run for ~4 hours. The solution turned a bright red-orange in color. The ethanol/methanol solvent was removed by rotary evaporation, and the

remaining product/H_2O mixture was refrigerated overnight to crystalize the product. Product was filtered using the same method as [Ru(tpy)(bpy)CN]OTf. The bright red-orange fine powder was collected (0.4015 g, 0.863 mmol). The yield for this reaction was ~75%.

Synthesis of [Ru{CNRu(tpy)(bpy)}$_2$]OTf$_4$.

$$(bpy)_2Ru(CN)_2 + 2[Ru(tpy)(bpy)OTf]OTf \xrightarrow[LiOTf]{Reflux} [Ru\{CNRu(tpy)(bpy)\}_2]OTf_4$$

A solution of $(bpy)_2Ru(CN)_2$ (0.304 g, 0.653 mmol) and 2.05 molar equivalents of [Ru(tpy)(bpy)OTf]OTf in a 2:1 ethanol/H_2O mixture was placed in a round bottom flask. The round bottom flask was attached to a water-cooled condenser, and the solution was placed under nitrogen atmosphere. The solution was heated to ~75 °C and allowed to reflux overnight. The reaction mixture was poured into a rotary evaporator flask with a LiOTf/H_2O solution. Ethanol was removed by rotary evaporation, and the mixture was placed in the refrigerator overnight. The product was scraped out of the flask (this may require some advanced-level scraping), and the flask washing was filtered with water rinses and dried over vacuum. The product was rinsed with ether before removing the vacuum. The product was collected (1.44 g, 0.609 mmol), and the reaction had a yield of ~90%.

4.2.3 *Sample Preparation*

Solutions of the monomer and dimer were made in the solvents listed below and studied using FTIR and UV/vis spectroscopies. The solvents used for the solvent study in Section 4.4 were methanol (Sigma), ethanol (Decon Lab), 1-butanol (Fischer), isopropanol (Mallinckrodt), acetone (Sigma), acetonitrile (EMD), benzonitrile (Alpha Aesar), and chloroform (Macron). All solutions were saturated with the exception of the monomer in methanol (6.25 mM) and the dimer in acetonitrile (3.44 mM).

4.3 COMPOUND VERIFICATION

To study MLCT reactions, and the role high frequency vibrations play in those reactions, we need to make sure we know what the complex is and that it does not have any impurities that might confuse the interpretation of any findings such as any unwanted RuCN combinations. Several spectroscopic techniques were used in conjunction with each other to verify the identity of each species and to try to gauge both the identity and amount of any impurities present. The primary method used to determine if the tpy and bpy ligands were attached to the ruthenium was ^{1}H NMR.

4.3.1 *^{1}H NMR Spectra*

The ^{1}H NMR spectra seen in Figure 4.1 were used as verification that the desired final products were made. The splitting patterns of the bipyridine and terpyridine seen in Figure 4.1 and differ from the literature for the isolated bipyridine and terpyridine complexes (12).[14] In general, peaks are shifted further downfield, and there is more splitting. The perpendicular conformation of the bipyridine and terpyridine affects the environments of surrounding hydrogens on the ligands, and having the nitrogens attached to Ru(II) instead of completely unattached like in the isolated species also affects the electron density in the polypyridines. Both of these will disturb the chemical shift of the hydrogens. The ^{1}H NMR spectra confirm that both the terpyridine and bipyridine were successfully coordinated to the ruthenium metal center.

In Figure 4.1a, the spectra of the chlorinated monomeric complex (pictured as a cartoon in Fig. 4.1e) displays a hydrogen that is shifted much further down field than the other hydrogens. Due to the electronegativity of the Cl ligand and the fact that the chloride shares a plane with it, this hydrogen is less shielded than the others on the pyridyl ligands and will experience a shift farther down field (δ = 10.25 ppm). The ^{1}H NMR spectrum of the monomer is displayed in Figure 4.1b. Here the chloride has been replaced with the desired cyanide ligand. By comparing these two

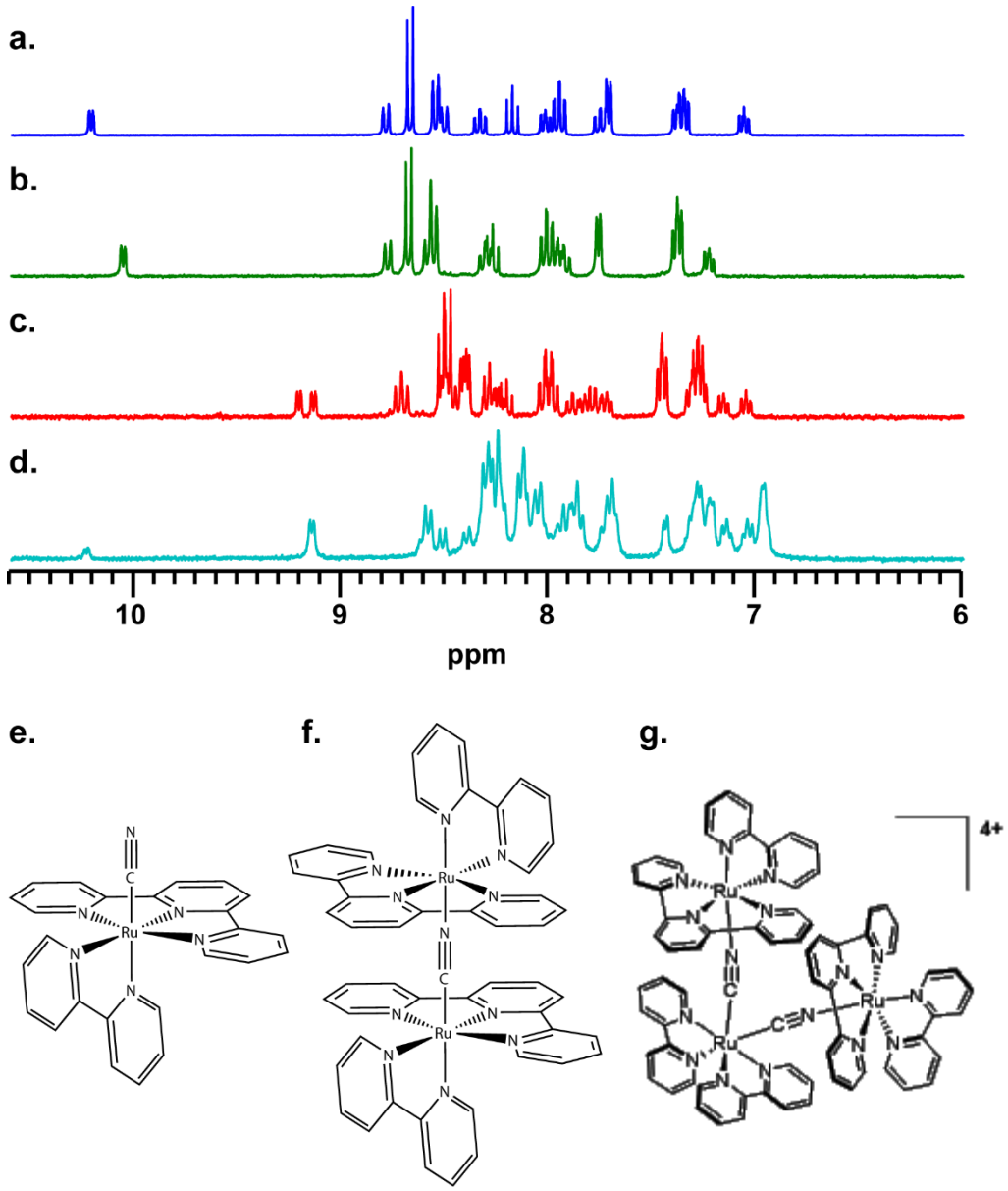


Figure 4.1: (a) ^{1}H NMR spectrum of $[Ru(tpy)(bpy)Cl]^{+}$. (b) ^{1}H NMR spectrum of $[Ru(tpy)(bpy)CN]^{+}$. (c) ^{1}H NMR spectrum of $[\{Ru(tpy)(bpy)\}_2CN]^{3+}$. (d) ^{1}H NMR spectrum for $[Ru\{CNRu(tpy)(bpy)\}_2]^{4+}$. The small peak at δ = 10.23 ppm is due to a monomer impurity in the sample. (e) Cartoon figure of monomer. (f) Cartoon figure of dimer. (g) Cartoon of trimer.

spectra (Fig. 4.1a,b), the greatest indication of the ligand exchange is in the hydrogen that is furthest down field. The hydrogen moved from a chemical shift of δ = 10.23 ppm to a shift of δ = 10.07 ppm. Since this is the hydrogen that was in the plane with the chloride, a change in the shift

indicates the coordinating ligand has also changed. Cyanide is less electronegative than the Cl ligand so the hydrogen should be shifted upfield, and this is what the spectra show.

The ^{1}H NMR spectrum for the dimer (Fig. 4.1c) is similar to the monomeric complex (Fig. 4.1b) except that all of the peaks show further doublet splitting. Since the molecule is not perfectly symmetric, 'matching' hydrogens on the polypyridines have slightly dissimilar environments. There are now two hydrogens in the plane of the bridging CN and their shifts are even farther up field, $\delta = 9.4$ ppm since the electron withdrawing effects of the cyanide are now split between two sets of bpy and tpy ligands. Figure 4.1d shows the ^{1}H NMR spectrum of the trimer. Here we see that the product is not entirely pure, because there is still a peak at a chemical shift of $\delta = 10.23$ ppm. This small amount of impurity can be assigned to the monomer, and further recrystallization should be able to remove it.

4.3.2 *FTIR Spectra*

While ^{1}H NMR verified that bpy and tpy were successfully attached to the ruthenium, it only hinted at the fact that the other coordinating ligand is indeed cyanide. To more conclusively verify this, FTIR spectra were taken of the monomer and dimer. The CN ligands on the dimer and monomer should have stretching frequencies in the range of 2000 - 2150 cm^{-1} based on previous infrared work on metal cyanide species in solution.[7,10,11,13,15-18] Both FTIR spectra for the dimer and monomer in methanol (Figure 4.2) show a peak around 2100 cm^{-1} confirming that the cyanide group was successfully coordinated to the ruthenium.

In addition to making sure that the cyanide ligand was attached to the ruthenium, we need to make sure that the compound will stay stable over a period of a couple of days, the length of time it takes to run a full set of nonlinear vibrational spectroscopy experiments. Solutions of monomer and dimer in methanol were left sitting in a vial for over two days to test this. Spectra taken at the

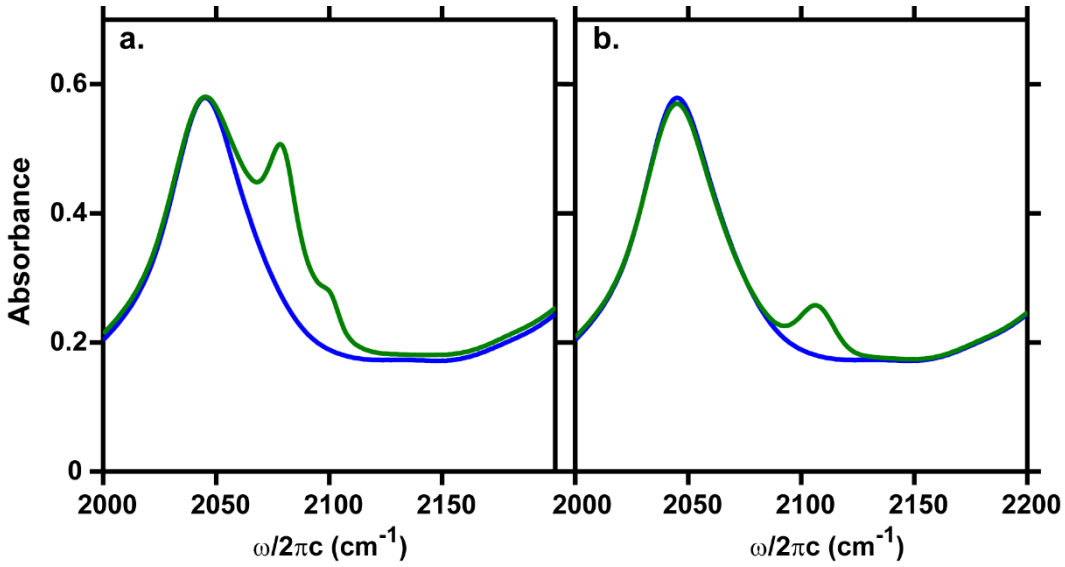


Figure 4.2: FTIR spectrum of the monomer (a) and dimer (b) in methanol. The methanol (blue) is overlayed with the monomer or dimer solution (green) to show the CN peak that occurs around 2080 cm^{-1} (a) and just above 2100 cm^{-1} (b).

beginning and end of the period were compared to see if any significant changes had occurred. Figure 4.3 shows the FTIR spectra taken on day one and day three. For the dimer in methanol (Fig. 4.3b), the absorbance increases, but the width and shape of the peak appear unchanged. Increased absorbance would indicate an increase of the concentration of the dimer solution. The

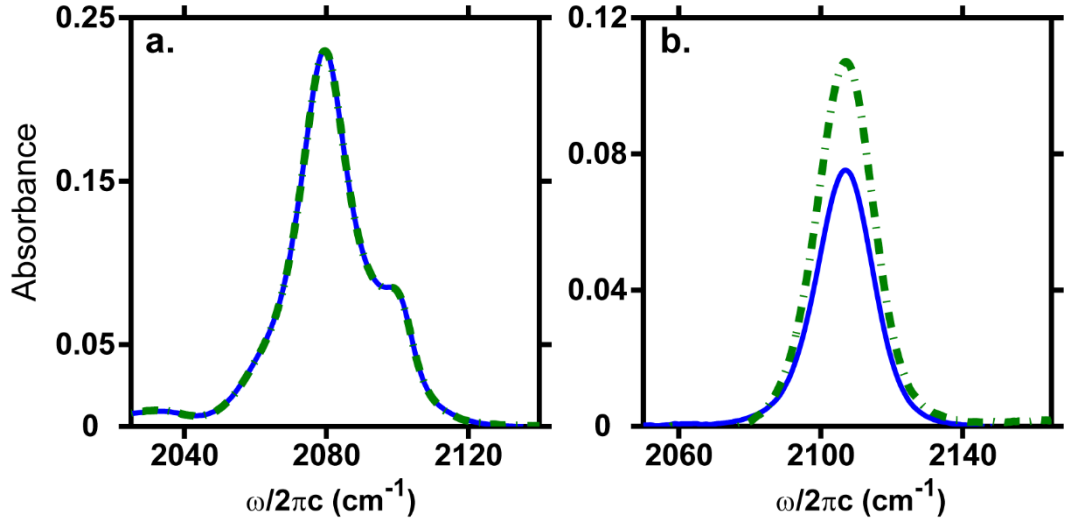


Figure 4.3: Testing stability of monomer (a) and dimer (b) in methanol over a period of three days. Day 1 is in blue, and day 3 is dashed green.

two spectra for the monomer (Fig. 4.3a) overlap completely. It appears the two complexes remain stable in solution over a period of several days.

4.3.3 *UV/vis Spectra*

Charge transfer transitions usually have spectroscopic signatures in the near-infrared to ultraviolet regions of the electromagnetic spectrum. The UV/vis spectra of the dimer and monomer were taken to visualize the MLCT from the ruthenium metal to the terpyridine ligand. This MLCT transition is of particular interest, because it relaxes directly into a long-lived triplet state[5] where we can study vibrational dynamics before the system relaxes back to the ground electronic state. The charge transfer reaction for FePtFe previously studied by our group relaxes within ~200 fs which only allows for the study of 'hot' vibrational states on the electronic ground state. The UV/vis spectra show the MLCT band at ~610 nm for both the monomer (Fig. 4.4, left) and the dimer (Fig. 4.4, right). Previous work with these compounds has shown this transition at ~610 nm providing one more confirmation the desired product was made.[5]

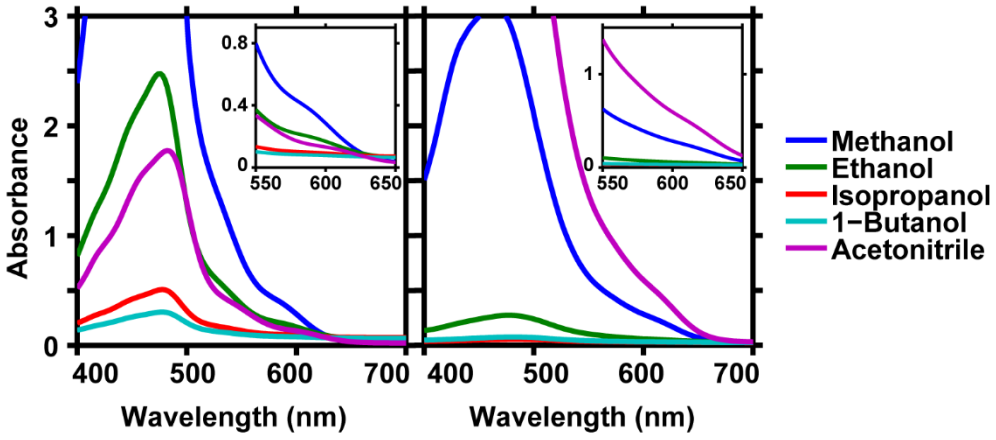


Figure 4.4: UV/vis spectra of the monomer (left panel) and dimer (right panel) in several solvents. Each inset shows a shoulder around 610 nm corresponding to the metal to ligand charge transfer of interest.

4.4 Solvent Study

There are no published FTIR for the monomer, dimer, or trimer as of this work. To find a solvent that would work for future experiments, both the dimer and monomer were placed in the following solvents to gauge their solubility: four alcohols (methanol, ethanol, isopropanol and 1-butanol), acetonitrile, benzonitrile, and chloroform, all of which are polar. These solvents were chosen for a combination of low absorbance in the IR region of interest, ease of access, and how easily they can be set up to flow through the sample cell for nonlinear experiments. With the exception of acetonitrile, the monomer was far more soluble in each solvent than the dimer. Ideally, an absorbance of 0.2 to 0.4 OD at the cyanide stretching frequency could be reached while minimizing solvent background. However, since the monomer and dimer are both charged and have large hydrophobic ligands, the compounds did not dissolve particularly well in highly polar solvents and hardly dissolve at all in the less polar solvents like the longer alcohol chains as can be seen in Figure 4.5. No solvent achieved greater than 0.2 OD for the dimer, and only methanol allowed

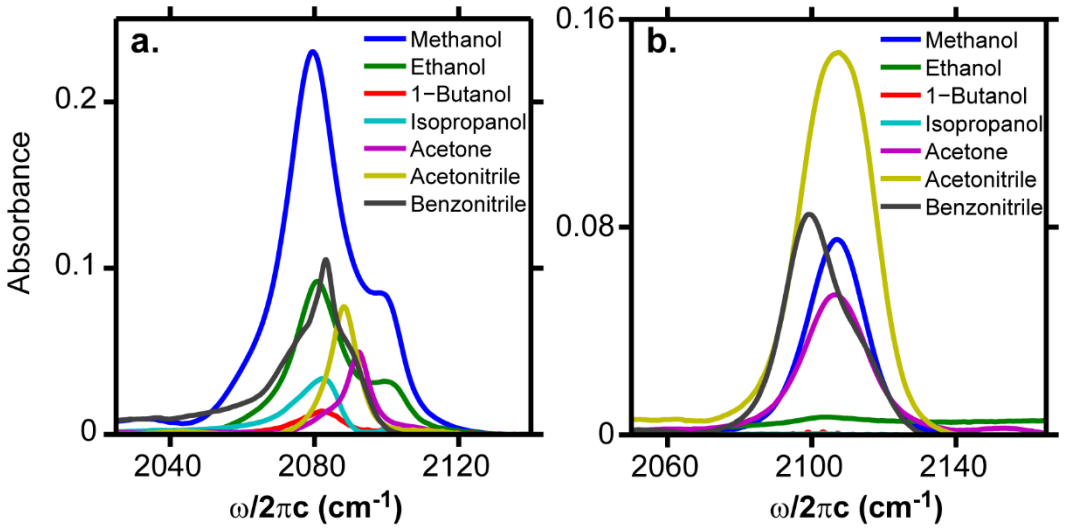


Figure 4.5: FTIR spectrum of monomer (a) and dimer (b) in several solvents where the solvent backgrounds have been subtracted.

the absorbance for the monomer to exceed this threshold. This greatly limits the choice of solvents available for experiments.

The cyanide stretching region of each FTIR spectrum shown in Figure 4.5 was fit using Voigt profiles. All dimer and the monomer samples not dissolved in alcohol were fit with a single peak. Monomer samples dissolved in alcohol were fit to two peaks. The resulting frequencies and widths along with their error bars are listed in Tables 4.1 and 4.2 for the dimer and monomer, respectively. Center frequencies for the dimer range from 2101 – 2107 cm^{-1}, while the frequencies for the main monomer peak clump right around 2080 cm^{-1} with the notable exceptions of acetone and acetonitrile which have center frequencies around 2090 cm^{-1}. It is interesting that the CN of the dimer, which should have less interaction with the solvent, has more variation in its frequencies than the CN of the monomer has for the same solvents. The second peak for the two monomers with enough amplitude to fit it, both have frequencies at 2100 cm^{-1}. The widths for the dimer samples range from 9 – 11 cm^{-1} while the widths of the monomer peaks range from 4 – 9 cm^{-1}. It is again interesting to note that the dimer has broader peaks than the monomer as this could indicate a more heterogeneous distribution of environments for the cyanide ligand when the cyanide ligand on the monomer should have more diverse interactions because it is readily available to the solvent. Of course, the broader peaks could be due to faster vibrational relaxation. Further experiments using 2D IR spectroscopy would help separate how much of the peak width comes from each component.

Table 4.1: Best fit parameters for the frequency and width of FTIR peaks for the dimer in different solvents. The error bars for these fits encompass a 95% confidence interval.

	Frequency (cm^{-1})	**Width (cm^{-1})**
Methanol	2106.8 ± 0.1	9.3 ± 0.1
Acetone	2106.5 ± 0.1	10.2 ± 0.1
Acetonitrile	2106.9 ± 0.1	11.4 ± 0.2
Benzonitrile	2101.1 ± 0.1	10.9 ± 0.2

Table 4.2: Best fit parameters for the frequency and width of FTIR peaks for the monomer in different solvents. The error bars for these fits encompass a 95% confidence interval.

	Peak 1		Peak 2	
	Frequency (cm^{-1})	**Width (cm^{-1})**	**Frequency (cm^{-1})**	**Width (cm^{-1})**
Methanol	2079.5 ± 0.1	9.21 ± 0.09	2099.2 ± 0.2	5.7 ± 0.2
Ethanol	2080.8 ± 0.1	7.8 ± 0.1	2100.3 ± 0.2	6.2 ± 0.2
Isopropanol	2080.3 ± 0.2	7.1 ± 0.3	--	--
1-butanol	2081.7 ± 0.1	7.0 ± 0.2	--	--
Acetone	2091.9 ± 0.1	4.4 ± 0.2	--	--
Acetonitrile	2088.1 ± 0.1	4.69 ± 0.08	--	--
Benzonitrile	2081.5 ± 0.2	9.1 ± 0.4	--	--

During the course of this solvent study, it was noticed that the monomer had two peaks in the alcohols. The second peak at ~2100 cm^{-1} was initially thought to be a dimer impurity. However, the frequency for the peak is too low, as can be seen from comparing the frequencies in Tables 4.1 and 4.2. It is possible that this peak might represent CN hydrogen-bonded to the solvent through the nitrogen since it only occurs in the solvents that contain hydroxide groups. Further investigation of this feature using temperature dependent FTIR to see how the two peaks change relative to each other or 2D IR to look at spectral diffusion rates should provide more information about the difference in environment for each population of CN ligands.

4.5 Conclusions and Future Work

Using ^{1}H NMR, FTIR, and UV/vis spectroscopies, we verified the synthesis of the monomer and dimer was successful. The trimer needs more purification, but its synthesis was also successful. The dimer and monomer were both found to be stable in solution over the period of a couple of days providing promise for future multi-day experiments. A MLCT band at ~610 nm was seen in the UV/vis spectra for both the monomer and the dimer in several solvents, and the FTIR spectra showed sufficient optical density in several solvents to see signal for even higher

nonlinear spectroscopy experiments. Further work will include synthesis of further purification of the trimer, as well as FTIR and UV/vis studies in multiple solvents. Temperature dependent FTIR studies of the monomer in methanol and ethanol would help elucidate the origin of the two cyanide stretching bands. Finally, nonlinear vibrational spectroscopy experiments could include a study of the ground electronic state using 2D IR and a study of the possible coupling of high-frequency cyanide stretches with the MLCT transition using vis-pump IR-probe experiments for the monomer, dimer, and trimer.

ACKNOWLEDGEMENTS

I would like to thank "my" undergraduate Laura Estergreen who actually did most of the synthesis, Dr. Michael Norris for his help in collecting NMR spectra and for helping us develop some of these syntheses, and Prof. Brandi Cossairt for kindly allowing us the use of her laboratory space for portions of these reactions.

REFERENCES

[1] Heshmat, M.; Baerends, E. J.; Polavarapu, P. L.; Nicu, V. P. *J. Phys. Chem. A* **2014**, *118*, 4766.

[2] Kraack, J. P.; Lotti, D.; Hamm, P. *J. Phys. Chem. Lett.* **2014**, *5*, 2325.

[3] Pluhařová, E.; Baer, M. D.; Mundy, C. J.; Schmidt, B.; Jungwirth, P. *J. Phys. Chem. Lett.* **2014**, *5*, 2235.

[4] Fleisher, A. J.; Bjork, B. J.; Bui, T. Q.; Cossel, K. C.; Okumura, M.; Ye, J. *J. Phys. Chem. Lett.* **2014**, *5*, 2241.

[5] Rey, R.; Hynes, J. T. *J. Chem. Phys.* **1996**, *104*, 2356.

[6] Kasyanenko, V. M.; Lin, Z.; Rubtsov, G. I.; Donahue, J. P.; Rubtsov, I. V. *J. Chem. Phys.* **2009**, *131*, 154508.

[7] Lynch, M. S.; Van Kuiken, B. E.; Daifuku, S. L.; Khalil, M. *J. Phys. Chem. Lett.* **2011**, *2*, 2252.

[8] Lynch, M. S.; Slenkamp, K. M.; Khalil, M. *J. Chem. Phys.* **2012**, *136*, 241101.

[9] Kwak, K.; Rosenfeld, D. E.; Fayer, M. D. *J. Chem. Phys.* **2008**, *128*, 204505.

[10] Tsai, C. N.; Allard, M. M.; Lord, R. L.; Luo, D. W.; Chen, Y. J.; Schlegel, H. B.; Endicott, J. F. *Inorg. Chem.* **2011**, *50*, 11965.

[11] Wang, C.; Mohney, B. K.; Akhremitchev, B. B.; Walker, G. C. *J. Phys. Chem. A* **2000**, *104*, 4314.

[12] Doorn, S. K.; Dyer, R. B.; Stoutland, P. O.; Woodruff, W. H. *J. Am. Chem. Soc.* **1993**, *115*.

[13] Wang, C.; Mohney, B. K.; Williams, R. D.; Petrov, V.; Hupp, J. T.; Walker, G. C. *J. Am. Chem. Soc.* **1998**, *120*, 5848.

[14] Slenkamp, K. M.; Lynch, M. S.; Van Kuiken, B. E.; Brookes, J. F.; Bannan, C. C.; Daifuku, S. L.; Khalil, M. *J. Chem. Phys.* **2014**, *140*, 084505.

[15] Ohta, K.; Maekawa, H.; Tominaga, K. *J. Phys. Chem. A* **2004**, *108*, 1333.

[16] Hester, R. E.; Nour, E. M. *J. Chem. Soc., Dalton Trans.* **1981**, 939.

[17] Zhu, N.; Vahrenkamp, H. *J. Organomet. Chem.* **1999**, *573*, 67.

[18] Sando, G. M.; Zhong, Q.; Owrutsky, J. C. *J. Chem. Phys.* **2004**, *121*, 2158.

Appendix A:

Matlab Code for the Simulation of 2D IR Spectra

This Appendix contains all the code used to simulate the 2D IR spectra shown in Chapter 1 and Chapter 2. The first three sections contain basic examples of how to set-up code for two coupled modes and four coupled modes. Section A.4 contains an example of the code I used for fitting experimental spectra. Finally, Section A.5 contains the code for the background calculations that are the same for all systems. The basis of the code was written by Ben Van Kuiken, and I edited it for the examples shown here and for use with Matlab's fitting routine as well as commenting it.

A.1 Six Level System with Homogeneous Lineshapes

This is the most basic code for two couple modes. It is written with homogeneous lineshape functions and parallel transition dipole moments. This code was used to produce Figures 1.5 and 1.8 both at $\tau_2 = 0$ fs.

A.1.1 *sixLevelHomLimPolarComp.m*

```
1    %% Six Level Test - Two Coupled Orthogonal Dipoles
2    % This is an example and test of specPack for a 6 level system using
3    % homogeneous limit lineshapes
4
5    %% Set Path
6    %This path allows several folders to access the same specPack file
7    addpath('C:\Users\Karla\Documents\GradSchool
     Work\Research\Theses\Thesis\six-level examples for intro\specPack3')
```

```
8
9    %% Clear Memory
10   clear %clears memory (clears variables in workspace)
11   clc %clears command window
12
13   %% Define system parameters
14
15   %'energies' contains all frequencies for each level in the system.  The
16   %order does not matter as long as you follow through with the ordering
     in
17   %all other matrices and arrays.  Frequencies have been shifted down by
     1500
18   %in this example to speed calculation time (but anharmonicities are not
19   %scaled).  This can be reversed in the final output by adding 1500 to
     each
20   %frequency axis.
21   energies = [0.0 500 560 985 1045 1105];
22
23   %Transition dipole magnitudes (relative amplitudes-I've always defined
     one
24   %transition with an amplitude of 1, and all the other have amplitudes
     less
25   %than 1)
26   rt2 = sqrt(2.0);
27   mu = [ 0.0 1.0 1.0 0.0 0.0 0.0;...
28   0.0 0.0 rt2 1.0 0.0;...
29   0.0 0.0 0.0 1.0 rt2;
30   rt2 0.0 0.0 0.0 0.0;...
31   1.0 1.0 0.0 0.0 0.0;...
```

```
32  0.0 rt2 0.0 0.0 0.0];
33
34  %Define the number of states/levels your system contains
35  nstates = 6; % 6 for a system of two coupled modes
36  for i=1:nstates
37  for j=1:nstates
38  for k = 1:3
39  muDir(i,j,k) = 0.0; %Defining a 3D matrix for transition dipole
    directions
40  end
41  end
42  end
43
44  %Any nonzero elements of the mu direction matrix need to be defined
    here
45  %muDir(i,j,k) = m where i and j are relevant states, k is the approp
    x,y, or z
46  %axis, and m is the magnitude along that axis
47  %The size of this list will change with the number of states in your
    system
48  muDir(1,2,1) = 1.0;
49  muDir(2,1,1) = 1.0;
50  muDir(1,3,1) = 1.0;
51  muDir(3,1,1) = 1.0;
52  muDir(2,4,1) = 1.0;
53  muDir(4,2,1) = 1.0; %These are currently written for parallel moments
54  muDir(2,5,1) = 1.0;
55  muDir(5,2,1) = 1.0;
56  muDir(3,5,1) = 1.0;
```

```
muDir(5,3,1) = 1.0;
muDir(3,6,1) = 1.0;
muDir(6,3,1) = 1.0;

% muDir(1,2,1) = 1.0;
% muDir(2,1,1) = 1.0;
% muDir(1,3,2) = 1.0;
% muDir(3,1,2) = 1.0;
% muDir(2,4,1) = 1.0;
% muDir(4,2,1) = 1.0; %These are currently written for perpendicular
moments
% muDir(2,5,2) = 1.0;
% muDir(5,2,2) = 1.0;
% muDir(3,5,1) = 1.0;
% muDir(5,3,1) = 1.0;
% muDir(3,6,2) = 1.0;
% muDir(6,3,2) = 1.0;

%Define your intial populations.  Here all population starts in the
ground
%state.  Size is 'nstates' dependent.
rho0 = [1.0 0.0 0.0 0.0 0.0 0.0];

%Defining a cell array for lineshape funtions
hfun = cell(3);
for i = 1:nstates
for j = 1:nstates
hfun{i,j}=@g0;
end
```

```
84  end
85
86  %All appropriate lineshapes functions for a 6-level system are called
    and
87  %assigned to their cell here.  Lineshape functions need to be defined
    in
88  %their own function files.  This list will grow as the number of states
    in
89  %the system grow.
90  hfun{2,2}=@gaaHL;
91  hfun{2,4}=@ga2aHL;
92  hfun{4,2}=@ga2aHL;
93  hfun{3,3}=@gbbHL;
94  hfun{3,6}=@gb2bHL;
95  hfun{6,3}=@gb2bHL;
96  hfun{4,4}=@g2a2aHL;
97  hfun{6,6}=@g2b2bHL;
98  hfun{2,3}=@gabHL;
99  hfun{3,2}=@gabHL;
100 hfun{2,5}=@gaabHL;
101 hfun{5,2}=@gaabHL;
102 hfun{3,5}=@gbabHL;
103 hfun{5,3}=@gbabHL;
104 hfun{5,5}=@gababHL;
105
106 %Orientational diffusion constant
107 dor = 2.0e-5;
108
109 %% Initialize structs
```

```
110 % In this section all data about system, experiment and calculation
    options
111 % are packed in to c-style structures for convenient data handling
112
113 %center frequencies for experiment's three "pulses" and local
    oscillator
114 %field
115 field_freq = [530 530 530 530];
116
117 %Calls 'initSystem' function from specPack and passes along all the
    system
118 %information defined above
119 sys = initSystem( nstates, energies, mu, muDir, rho0, hfun, dor);
120
121 %Experiment specific information - Here it is initiating a 2DIR
    measurement
122 %in the ZZZZ polarization
123 expr = initExperiment(field_freq, '2DIR', 'ZZZZ');
124
125 %'sp_SetOptions' should always be opened and checked manually to make
    sure
126 %all parameters are appropriate for the system
127 opts = sp_SetOptions('tstep',8.0);
128
129 %% Call 2D calculation function
130
131 %'tau2' can be a single time point or a list (in femtoseconds)
132 tau2 = [0 50];
133
```

```
%the first value passed to 'calc2DImpulse' is a user-defined named that
all
%generated files will be saved under
calc2DImpulse('sixLevel_ZZZZ',sys,expr,tau2,opts);

%% Now calculate YYZZ Polarization
% Redefine the system parameters to calculate the YYZZ polarization and
% recompute the spectrum

expr = initExperiment(field_freq, '2DIR', 'YYZZ');

calc2DImpulse('sixLevel_YYZZ', sys, expr, tau2, opts);

%% Plot 2DIR Correlation Spectrum (ZZZZ Polarization)

cspec1 = load('sixLevel_ZZZZ_t2-0_abs.mat');

figure(1)
contourf(cspec1.freq(1251:1312),cspec1.freq(1251:1312),...
cspec1.abs1(1251:1312,1251:1312),20)
xlim([461 584])
ylim([461 584])
axis square
set(gca, 'Tickdir', 'out');
set(gca, 'FontWeight','bold')
set(gca,'FontSize',16)
set(gca, 'FontName','Arial')
set(gca,'TickLength', [0.03, 0.01])
set(gca,'LineWidth', 2)
```

```
xlabel('\omega_{1}/2\pi (cm^{-1})')
ylabel('\omega_{3}/2\pi (cm^{-1})')
title('ZZZZ, \tau_2 = 0')


clear cspec1

%% Plot 2DIR Correlation Spectrum (YYZZ Polarization)

cspec1 = load('sixLevel_YYZZ_t2-0_abs.mat');

figure(2)
contourf(cspec1.freq(1251:1312),cspec1.freq(1251:1312),...
cspec1.abs1(1251:1312,1251:1312),20)
xlim([461 584])
ylim([461 584])
axis square
set(gca, 'Tickdir', 'out');
set(gca, 'FontWeight','bold')
set(gca,'FontSize',16)
set(gca, 'FontName','Arial')
set(gca,'TickLength', [0.03, 0.01])
set(gca,'LineWidth', 2)
xlabel('\omega_{1}/2\pi (cm^{-1})')
ylabel('\omega_{3}/2\pi (cm^{-1})')
title('YYZZ, \tau_2 = 0')


clear cspec1
```

A.1.2 *gaaHL.m*

This is a lineshape function, gbbHL.m follows the same format.

```
1    function y = gaaHL( t )
2    %GAA Lineshape function g_{aa}
3    %   An example of a homogeneous lineshape function for mode a
4
5    fac = 2000;
6
7    y = abs(t)/fac;
8    end
```

A.1.3 *ga2aHL.m*

The lineshape function gb2bHL.m follows the same format.

```
1    function y = ga2aHL( t )
2    %GAA Lineshape function g_{a2a}
3    %   An example of a homogeneous lineshape function for the transition
     from
4    %   a to 2a
5
6    fac = 2000;
7
8    y = 2.0*abs(t)/fac;
9    end
```

A.1.4 *g2a2aHL.m*

The lineshape function g2b2bHL.m follows the same format.

```
1    function y = g2a2aHL( t )
2    %GAA Lineshape function g_{aa}
```

```
3     %   An example of a homogeneous lineshape function for the 2a energy
      level
4
5     fac = 2000;
6
7     y = 4.0*abs(t)/fac;
8     end
```

A.1.5 *gabHL.m*

```
1     function y = gabHL( t )
2     %GAA Lineshape function g_{ab}
3
4     fac = 2000;
5
6     y = abs(t)/fac;
7     end
```

A.1.6 *gaabHL.m*

The lineshape function gbabHL.m follows the same format.

```
1     function y = gaabHL( t )
2     %GAA Lineshape function g_{aab}
3     %   An example of a homogeneous lineshape function for the transition
      from
4     %   a to ab
5
6     fac1 = 2000;
7     fac2 = 2000;
8
9     y = abs(t)/fac1+abs(t)/fac2;
```

```
10    end
```

A.1.7 *gababHL.m*

```
1     function y = gababHL( t )
2     %GAA Lineshape function g_{abab}
3     %   An example of a homogeneous lineshape function for the ab energy
      level
4
5     fac1 = 2000; %same as fac for gaa
6     fac2 = 2000; %same as fac for gab
7     fac3 = 2000; %same as fac for gbb
8
9     y = abs(t)/fac1+2.0*abs(t)/fac2+abs(t)/fac3;
10    end
```

A.2 SIX LEVEL SYSTEM WITH POPULATION RELAXATION

Make sure the sp_SetOptions.m file in SpecPack3 is updated to have a 'true' values for the population relaxation value. This code was used to make figure 1.6.

A.2.1 *sixLevel_poprlx.m*

```
1     %% Six Level Test - Two Coupled Orthogonal Dipoles
2     % This is an example and test of specPack for a 6 level system.  Here
3     % population relaxation and 45 degree angle between dipole moments is
4     % shown.  General comments on code can be found in the homogeneous
      limit
5     % example, and only comments on changes are noted.
6
7     %Warning!!! I believe there is an error in the implementation of the
8     %population lifetimes.  Things don't seem to relax as quickly (or
```

```
%correctly) as they should.

%% Set Path
addpath('C:\Users\Karla\Documents\Grad School
Work\Research\Theses\Thesis\six-level examples for intro\specpack3')

%% Clear Memory
clear
clc

%% Define system parameters

energies = [0.0 500 560 985 1045 1105];

rt2 = sqrt(2.0);
mu = [ 0.0 1.0 1.0 0.0 0.0 0.0;...
0.0 0.0 rt2 1.0 0.0;...
0.0 0.0 0.0 1.0 rt2;
rt2 0.0 0.0 0.0 0.0;...
1.0 1.0 0.0 0.0 0.0;...
0.0 rt2 0.0 0.0 0.0];

nstates = 6;
for i=1:nstates
for j=1:nstates
for k = 1:3
muDir(i,j,k) = 0.0;
end
end
```

```
37   end
38
39
40   muDir(1,2,2) = 1.0/rt2;
41   muDir(2,1,2) = 1.0/rt2;
42   muDir(1,2,3) = 1.0/rt2;
43   muDir(2,1,3) = 1.0/rt2;
44   muDir(1,3,2) = 1.0;
45   muDir(3,1,2) = 1.0;
46   muDir(2,4,2) = 1.0/rt2;
47   muDir(4,2,2) = 1.0/rt2;
48   muDir(2,4,3) = 1.0/rt2;%This is an example of how to set up 45 degrees
     btwn dipole moments
49   muDir(4,2,3) = 1.0/rt2;
50   muDir(2,5,2) = 1.0;
51   muDir(5,2,2) = 1.0;
52   muDir(3,5,2) = 1.0/rt2;
53   muDir(5,3,2) = 1.0/rt2;
54   muDir(3,5,3) = 1.0/rt2;
55   muDir(5,3,3) = 1.0/rt2;
56   muDir(3,6,2) = 1.0;
57   muDir(6,3,2) = 1.0;
58
59
60   rho0 = [1.0 0.0 0.0 0.0 0.0 0.0];
61
62
63   hfun = cell(3);
64   for i = 1:nstates
```

```
65   for j = 1:nstates
66   hfun{i,j}=@g0;
67   end
68   end
69
70
71   hfun{2,2}=@gaa;
72   hfun{2,4}=@ga2a;
73   hfun{4,2}=@ga2a;
74   hfun{3,3}=@gbb;
75   hfun{3,6}=@gb2b;
76   hfun{6,3}=@gb2b;
77   hfun{4,4}=@g2a2a;
78   hfun{6,6}=@g2b2b;
79   hfun{2,3}=@gab;
80   hfun{3,2}=@gab;
81   hfun{2,5}=@gaab;
82   hfun{5,2}=@gaab;
83   hfun{3,5}=@gbab;
84   hfun{5,3}=@gbab;
85   hfun{5,5}=@gabab;
86
87   dor = 2.0e-5;
88
89   %This is where the timescales for population lifetimes are entered.
     There
90   %needs to be a value for each energy level, so here six timescales.
91   lifetimes = [1e10 10.0e3 5.0e3 10e3 7.5e3 5e3];
92
```

```
93
94  %% Initialize structs
95
96  sys = initSystem( nstates, energies, mu, muDir, rho0, hfun, dor,
    lifetimes);
97
98  expr = initExperiment([500 500 500 500], '2DIR', 'ZZZZ');
99
100 opts = sp_SetOptions('prefacThresh',0.1,'tstep',10.0);
101
102 %% Call 2D calculation function
103 tau2 = [0,200,500,1000,2000,3500,5000,7000,10000];
104 calc2DImpulse('sixLevel_test_45deg_poprlx_zzzz',sys,expr,tau2,opts);
105
106
107 %Below commented out are other ways to plot the data (for rephasing and
108 %non-rephasing, in time or freq domain)
109
110 % %% Create Rephasing Time Plot
111 %
112 % rph_time = load('sixLevel_test_t2-250_rt.mat');
113 %
114 % figure(1)
115 % contourf(rph_time.t1(1:400),rph_time.t3(1:400),...
116 %   real(rph_time.response_time(1:400,1:400)),10)
117 %
118 % clear rph_time
119 % %% Create NonRephasing Time plot
120 %
```

```
121  % nr_time = load('sixLevel_test_t2-250_nrt.mat');
122  %
123  % figure(2)
124  % contourf(nr_time.t1(1:400),nr_time.t3(1:400),...
125  %   nr_time.response_time(1:400,1:400),10)
126  % axis square
127  %
128  % clear nr_time
129  %
130  % %% Plot rephasing frequency domain
131  %
132  % rf = load('sixLevel_test_t2-250_rf.mat');
133  %
134  % % Construct Frequency Axes
135  % f = 0.25/2*linspace(0,1,2048/2+1)/(2.998e-5);
136  %
137  % figure(3)
138  % contourf(rf.freq,rf.freq,rf.response_freq,10)
139  % clear rf
140  %
141  % %% Plot Non-Rephasing Spectrum
142  %
143  % nrf = load('sixLevel_test_t2-250_nrf.mat');
144  %
145  % figure(4)
146  % contourf(nrf.freq,nrf.freq,nrf.response_freq,10)
147  % axis square
148  % clear nrf
149
```

```
150  %% Plot 2DIR Correlation Spectrum
151
152  cspec1 = load('sixLevel_test_45deg_poprlx_zzzz_t2-0_abs.mat');
153  cspec2 = load('sixLevel_test_45deg_poprlx_zzzz_t2-200_abs.mat');
154  cspec3 = load('sixLevel_test_45deg_poprlx_zzzz_t2-500_abs.mat');
155  cspec4 = load('sixLevel_test_45deg_poprlx_zzzz_t2-1000_abs.mat');
156  cspec5 = load('sixLevel_test_45deg_poprlx_zzzz_t2-2000_abs.mat');
157  cspec6 = load('sixLevel_test_45deg_poprlx_zzzz_t2-3500_abs.mat');
158  cspec7 = load('sixLevel_test_45deg_poprlx_zzzz_t2-5000_abs.mat');
159  cspec8 = load('sixLevel_test_45deg_poprlx_zzzz_t2-7000_abs.mat');
160  cspec9 = load('sixLevel_test_45deg_poprlx_zzzz_t2-10000_abs.mat');
161
162  z = [-60000:8000:60000];
163
164  figure
165  contourf(cspec1.freq(1255:1310),cspec1.freq(1255:1310),...
166  real(cspec1.abs1(1255:1310,1255:1310)),z)
167  xlim([470 580])
168  ylim([470 580])
169  caxis([-60000 60000])
170  axis square
171  set(gca, 'Tickdir', 'out');
172  set(gca, 'FontWeight','bold')
173  set(gca,'FontSize',16)
174  set(gca, 'FontName','Arial')
175  set(gca,'TickLength', [0.03, 0.01])
176  set(gca,'LineWidth', 2)
177  xlabel('\omega_{1}/2\pi (cm^{-1})')
178  ylabel('\omega_{3}/2\pi (cm^{-1})')
```

```
179  title('ZZZZ, \tau_2 = 0')
180
181  figure
182  contourf(cspec2.freq(1255:1310),cspec2.freq(1255:1310),...
183  real(cspec2.abs1(1255:1310,1255:1310)),z)
184  xlim([470 580])
185  ylim([470 580])
186  caxis([-60000 60000])
187  axis square
188  set(gca, 'Tickdir', 'out');
189  set(gca, 'FontWeight','bold')
190  set(gca,'FontSize',16)
191  set(gca, 'FontName','Arial')
192  set(gca,'TickLength', [0.03, 0.01])
193  set(gca,'LineWidth', 2)
194  xlabel('\omega_{1}/2\pi (cm^{-1})')
195  ylabel('\omega_{3}/2\pi (cm^{-1})')
196  title('ZZZZ, \tau_2 = 200')
197
198  figure
199  contourf(cspec3.freq(1255:1310),cspec3.freq(1255:1310),...
200  real(cspec3.abs1(1255:1310,1255:1310)),z)
201  xlim([470 580])
202  ylim([470 580])
203  caxis([-60000 60000])
204  axis square
205  set(gca, 'Tickdir', 'out');
206  set(gca, 'FontWeight','bold')
207  set(gca,'FontSize',16)
```

```
set(gca, 'FontName','Arial')
set(gca,'TickLength', [0.03, 0.01])
set(gca,'LineWidth', 2)
xlabel('\omega_{1}/2\pi (cm^{-1})')
ylabel('\omega_{3}/2\pi (cm^{-1})')
title('ZZZZ, \tau_2 = 500')

figure
contourf(cspec4.freq(1255:1310),cspec4.freq(1255:1310),...
real(cspec4.abs1(1255:1310,1255:1310)),z)
xlim([470 580])
ylim([470 580])
caxis([-60000 60000])
axis square
set(gca, 'Tickdir', 'out');
set(gca, 'FontWeight','bold')
set(gca,'FontSize',16)
set(gca, 'FontName','Arial')
set(gca,'TickLength', [0.03, 0.01])
set(gca,'LineWidth', 2)
xlabel('\omega_{1}/2\pi (cm^{-1})')
ylabel('\omega_{3}/2\pi (cm^{-1})')
title('ZZZZ, \tau_2 = 1000')

figure
contourf(cspec5.freq(1255:1310),cspec5.freq(1255:1310),...
real(cspec5.abs1(1255:1310,1255:1310)),z)
xlim([470 580])
ylim([470 580])
```

```
caxis([-60000 60000])
axis square
set(gca, 'Tickdir', 'out');
set(gca, 'FontWeight','bold')
set(gca,'FontSize',16)
set(gca, 'FontName','Arial')
set(gca,'TickLength', [0.03, 0.01])
set(gca,'LineWidth', 2)
xlabel('\omega_{1}/2\pi (cm^{-1})')
ylabel('\omega_{3}/2\pi (cm^{-1})')
title('ZZZZ, \tau_2 = 2000')

figure
contourf(cspec6.freq(1255:1310),cspec6.freq(1255:1310),...
real(cspec6.abs1(1255:1310,1255:1310)),z)
xlim([470 580])
ylim([470 580])
caxis([-60000 60000])
axis square
set(gca, 'Tickdir', 'out');
set(gca, 'FontWeight','bold')
set(gca,'FontSize',16)
set(gca, 'FontName','Arial')
set(gca,'TickLength', [0.03, 0.01])
set(gca,'LineWidth', 2)
xlabel('\omega_{1}/2\pi (cm^{-1})')
ylabel('\omega_{3}/2\pi (cm^{-1})')
title('ZZZZ, \tau_2 = 3500')

```

```
figure
contourf(cspec7.freq(1255:1310),cspec7.freq(1255:1310),...
real(cspec7.abs1(1255:1310,1255:1310)),z)
xlim([470 580])
ylim([470 580])
caxis([-60000 60000])
axis square
set(gca, 'Tickdir', 'out');
set(gca, 'FontWeight','bold')
set(gca,'FontSize',16)
set(gca, 'FontName','Arial')
set(gca,'TickLength', [0.03, 0.01])
set(gca,'LineWidth', 2)
xlabel('\omega_{1}/2\pi (cm^{-1})')
ylabel('\omega_{3}/2\pi (cm^{-1})')
title('ZZZZ, \tau_2 = 5000')

figure
contourf(cspec8.freq(1255:1310),cspec8.freq(1255:1310),...
real(cspec8.abs1(1255:1310,1255:1310)),z)
xlim([470 580])
ylim([470 580])
caxis([-60000 60000])
axis square
set(gca, 'Tickdir', 'out');
set(gca, 'FontWeight','bold')
set(gca,'FontSize',16)
set(gca, 'FontName','Arial')
set(gca,'TickLength', [0.03, 0.01])
```

```
295  set(gca,'LineWidth', 2)
296  xlabel('\omega_{1}/2\pi (cm^{-1})')
297  ylabel('\omega_{3}/2\pi (cm^{-1})')
298  title('ZZZZ, \tau_2 = 7000')
299
300  figure
301  contourf(cspec9.freq(1255:1310),cspec9.freq(1255:1310),...
302  real(cspec9.abs1(1255:1310,1255:1310)),z)
303  xlim([470 580])
304  ylim([470 580])
305  caxis([-60000 60000])
306  axis square
307  set(gca, 'Tickdir', 'out');
308  set(gca, 'FontWeight','bold')
309  set(gca,'FontSize',16)
310  set(gca, 'FontName','Arial')
311  set(gca,'TickLength', [0.03, 0.01])
312  set(gca,'LineWidth', 2)
313  xlabel('\omega_{1}/2\pi (cm^{-1})')
314  ylabel('\omega_{3}/2\pi (cm^{-1})')
315  title('ZZZZ, \tau_2 = 10000')
316
317  figure(11)
318  subplot(2,3,1)
319
     contourf(cspec1.freq(1220:1400),cspec1.freq(1220:1400),real(cspec1.abs1
     (1220:1400,1220:1400)),z)
320  axis square
321  xlim([470 580])
```

```
322  ylim([470 580])
323  caxis([-40000 40000])
324  subplot(2,3,2)
325
     contourf(cspec2.freq(1220:1400),cspec2.freq(1220:1400),real(cspec2.abs1
     (1220:1400,1220:1400)),z)
326  axis square
327  xlim([470 580])
328  ylim([470 580])
329  caxis([-40000 40000])
330  subplot(2,3,3)
331
     contourf(cspec3.freq(1220:1400),cspec3.freq(1220:1400),real(cspec3.abs1
     (1220:1400,1220:1400)),z)
332  axis square
333  xlim([470 580])
334  ylim([470 580])
335  caxis([-40000 40000])
336  subplot(2,3,4)
337
     contourf(cspec4.freq(1220:1400),cspec4.freq(1220:1400),real(cspec4.abs1
     (1220:1400,1220:1400)),z)
338  axis square
339  xlim([470 580])
340  ylim([470 580])
341  caxis([-40000 40000])
342  subplot(2,3,5)
```

```
343
     contourf(cspec5.freq(1220:1400),cspec5.freq(1220:1400),real(cspec5.abs1
     (1220:1400,1220:1400)),z)
344  axis square
345  xlim([470 580])
346  ylim([470 580])
347  caxis([-40000 40000])
348  subplot(2,3,6)
349
     contourf(cspec6.freq(1220:1400),cspec6.freq(1220:1400),real(cspec6.abs1
     (1220:1400,1220:1400)),z)
350  axis square
351  xlim([470 580])
352  ylim([470 580])
353  caxis([-40000 40000])
```

A.2.2 *gaa.m*

The lineshape function gbb.m follows a similar format with its own correlation amplitude and timescale (Figure 1.6 uses the same values for modes a and b).

```
1    function y = gaa( t )
2    %GAA Lineshape function g_{aa}
3    %   See HL examples
4
5    t11 = 10.0e3; %Correlation time
6    s11 = 0.0005; %Amplitude from correlation function
7
8    y= s11^2*t11^2*(exp(-abs(t)/t11)-1.0+abs(t)/t11);
9
10   end
```

A.2.3 *ga2a.m*

The lineshape function gb2b.m follows a similar format.

```
1    function y = ga2a( t )
2    %GAA Lineshape function g_{aa}
3
4    t11 = 10.0e3;
5    s11 = 0.0005;
6
7    y= 2.0*s11^2*t11^2*(exp(-abs(t)/t11)-1.0+abs(t)/t11);
8
9    end
```

A.2.4 *g2a2a.m*

The lineshape function g2b2b.m follows a similar format.

```
1    function y = g2a2a( t )
2    %GAA Lineshape function g_{aa}
3
4    t11 = 10.0e3;
5    s11 = 0.0005;
6
7    y= 4.0*s11^2*t11^2*(exp(-abs(t)/t11)-1.0+abs(t)/t11);
8
9    end
```

A.2.5 *gab.m*

```
1    function y = gab( t )
2
3    %Gtt Lineshape function g_{tt}
```

```
4
5     A_a=0.0005;
6     A_b=0.0005;
7     tau_ab=10000;
8
9     y = A_a*A_b*tau_ab^2*(exp(-abs(t)/tau_ab)+abs(t)/tau_ab-1);
10
11    end
```

A.2.6 *gaab.m*

The lineshape function gbab.m follows a similar format.

```
1     function y = gaab( t )
2     %GAA Lineshape function g_{aa}
3
4     ta = 10.0e3;
5     sa = 0.0005;
6     tab = 10.0e3;
7     sab = 0.0005;
8     rho = 1.0;
9
10    y= sa^2*ta^2*(exp(-abs(t)/ta)-1.0+abs(t)/ta)+...
11    rho*sab^2*tab^2*(exp(-abs(t)/tab)-1.0+abs(t)/tab);
12
13    end
```

A.2.7 *gabab.m*

```
1     function y = gabab( t )
2     %GAA Lineshape function g_{aa}
3
```

```
4     ta = 10.0e3;
5     sa = 0.0005;
6     tb = 10.0e3;
7     sb = 0.0005;
8     tab = 10.0e3;
9     sab = 0.0005;
10    rho = 1.0;
11
12    y= sa^2*ta^2*(exp(-abs(t)/ta)-1.0+abs(t)/ta)+...
13    sb^2*tb^2*(exp(-abs(t)/tb)-1.0+abs(t)/tb)+...
14    2.0*rho*sab^2*tab^2*(exp(-abs(t)/tab)-1.0+abs(t)/tab);
15
16    end
```

A.3 Fifteen Level System in the Homogeneous Limit

This is the most basic code for a system of four coupled vibrations. It is written with homogeneous lineshape functions for four parallel transition dipole moments. This code was used to produce Figure 1.9.

A.3.1 *fifteenLevelHomLim.m*

```
1     %% 15 Level Test - Four Coupled Dipoles
2     % This is an example and test of specPack for a 15 level system with
3     % homogeneous lineshapes.  See the six level homogeneous example for
4     % more detailed comments.
5
6     %% Set Path
7     addpath('C:\Users\Karla\Documents\Grad School
      Work\Research\Theses\Thesis\six-level examples for intro\specpack3')
```

```
8
9     %% Clear Memory
10    clear
11    clc
12
13    %% Define system parameters
14
15    %Energies (in cm^-1)
16    a=500; %4 vibrations: a,b,c,d
17    b=540;
18    c=580;
19    d=620;
20    a2=970; %overtones
21    b2=1050;
22    c2=1130;
23    d2=1210;
24    ab=1020; %combination bands
25    ac=1060;
26    ad=1100;
27    bc=1100;
28    bd=1140;
29    cd=1180;
30
31    energies = [0 a b c d a2 b2 c2 d2 ab ac ad bc bd cd];
32
33    rt2 = sqrt(2.0);
34    mu = [ 0.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0;...
      %1
35    0.0 0.0 0.0 0.0 rt2 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0;... %2
```

```
0.0 0.0 0.0 0.0 0.0 rt2 0.0 0.0 1.0 0.0 0.0 1.0 1.0 0.0;... %3
0.0 0.0 0.0 0.0 0.0 0.0 rt2 0.0 0.0 1.0 0.0 1.0 0.0 1.0;... %4
0.0 0.0 0.0 0.0 0.0 0.0 0.0 rt2 0.0 0.0 1.0 0.0 1.0 1.0;... %5
rt2 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0;... %6
0.0 rt2 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0;... %7
0.0 0.0 rt2 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0;... %8
0.0 0.0 0.0 rt2 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0;... %9
1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0;... %10
1.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0;... %11
1.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0;... %12
0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0;... %13
0.0 1.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0;... %14
0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0];   %15

nstates = 15;
for i=1:nstates
for j=1:nstates
for k = 1:3
muDir(i,j,k) = 0.0;
end
end
end

muDir(1,2,1) = 1.0;
muDir(2,1,1) = 1.0;
muDir(1,3,1) = 1.0;
muDir(3,1,1) = 1.0;
muDir(1,4,1) = 1.0;
muDir(4,1,1) = 1.0;
```

```
65    muDir(1,5,1) = 1.0;
66    muDir(5,1,1) = 1.0;
67
68    muDir(2,6,1) = 1.0;
69    muDir(6,2,1) = 1.0;
70    muDir(3,7,1) = 1.0;
71    muDir(7,3,1) = 1.0;
72    muDir(4,8,1) = 1.0;
73    muDir(8,4,1) = 1.0;
74    muDir(5,9,1) = 1.0;
75    muDir(9,5,1) = 1.0;
76
77    muDir(2,10,1) = 1.0;
78    muDir(10,2,1) = 1.0;
79    muDir(2,11,1) = 1.0;
80    muDir(11,2,1) = 1.0;
81    muDir(2,12,1) = 1.0;
82    muDir(12,2,1) = 1.0;
83    muDir(3,10,1) = 1.0;
84    muDir(10,3,1) = 1.0;
85    muDir(3,13,1) = 1.0;
86    muDir(13,3,1) = 1.0;
87    muDir(3,14,1) = 1.0;
88    muDir(14,3,1) = 1.0;
89    muDir(4,11,1) = 1.0;
90    muDir(11,4,1) = 1.0;
91    muDir(4,13,1) = 1.0;
92    muDir(13,4,1) = 1.0;
93    muDir(4,15,1) = 1.0;
```

```
94   muDir(15,4,1) = 1.0;
95   muDir(5,12,1) = 1.0;
96   muDir(12,5,1) = 1.0;
97   muDir(5,14,1) = 1.0;
98   muDir(14,5,1) = 1.0;
99   muDir(5,15,1) = 1.0;
100  muDir(15,5,1) = 1.0;
101
102
103  rho0 = [1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0];
104
105  hfun = cell(3);
106  for i = 1:nstates
107  for j = 1:nstates
108  hfun{i,j}=@g0;
109  end
110  end
111
112  hfun{2,2}=@gaaHL;
113  hfun{2,6}=@ga2aHL;
114  hfun{6,2}=@ga2aHL;
115  hfun{3,3}=@gbbHL;
116  hfun{3,7}=@gb2bHL;
117  hfun{7,3}=@gb2bHL;
118  hfun{4,4}=@gccHL;
119  hfun{4,8}=@gc2cHL;
120  hfun{8,4}=@gc2cHL;
121  hfun{5,5}=@gddHL;
122  hfun{5,9}=@gd2dHL;
```

```
123  hfun{9,5}=@gd2dHL;
124  hfun{6,6}=@g2a2aHL;
125  hfun{7,7}=@g2b2bHL;
126  hfun{8,8}=@g2c2cHL;
127  hfun{9,9}=@g2d2dHL;
128
129  hfun{2,3}=@gabHL;
130  hfun{3,2}=@gabHL;
131  hfun{2,10}=@gaabHL;
132  hfun{10,2}=@gaabHL;
133  hfun{3,10}=@gbabHL;
134  hfun{10,3}=@gbabHL;
135  hfun{10,10}=@gababHL;
136
137  hfun{2,4}=@gacHL;
138  hfun{4,2}=@gacHL;
139  hfun{2,11}=@gaacHL;
140  hfun{11,2}=@gaacHL;
141  hfun{4,11}=@gcacHL;
142  hfun{11,4}=@gcacHL;
143  hfun{11,11}=@gacacHL;
144
145  hfun{2,5}=@gadHL;
146  hfun{5,2}=@gadHL;
147  hfun{2,12}=@gaadHL;
148  hfun{12,2}=@gaadHL;
149  hfun{5,12}=@gdadHL;
150  hfun{12,5}=@gdadHL;
151  hfun{12,12}=@gadadHL;
```

```
152
153  hfun{3,4}=@gbcHL;
154  hfun{4,3}=@gbcHL;
155  hfun{3,13}=@gbbcHL;
156  hfun{13,3}=@gbbcHL;
157  hfun{4,13}=@gcbcHL;
158  hfun{13,4}=@gcbcHL;
159  hfun{13,13}=@gbcbcHL;
160
161  hfun{3,5}=@gbdHL;
162  hfun{5,3}=@gbdHL;
163  hfun{3,14}=@gbbdHL;
164  hfun{14,3}=@gbbdHL;
165  hfun{5,14}=@gdbdHL;
166  hfun{14,5}=@gdbdHL;
167  hfun{14,14}=@gbdbdHL;
168
169  hfun{4,5}=@gcdHL;
170  hfun{5,4}=@gcdHL;
171  hfun{4,15}=@gccdHL;
172  hfun{15,4}=@gccdHL;
173  hfun{5,15}=@gdcdHL;
174  hfun{15,5}=@gdcdHL;
175  hfun{15,15}=@gcdcdHL;
176
177  dor = 2.0e-5;
178
179  %% Initialize structs
180
```

```
181  sys = initSystem( nstates, energies, mu, muDir, rho0, hfun, dor);
182
183  expr = initExperiment([560 560 560 560], '2DIR', 'ZZZZ');
184
185  opts = sp_SetOptions('tstep',6.0);
186
187  %% Call 2D calculation function
188  tau2 = [0,200];
189  calc2DImpulse('fifteenLevel_test',sys,expr,tau2,opts);
190
191  %% Create Rephasing Time Plot
192
193  rph_time = load('fifteenLevel_test_t2-200_rt.mat');
194
195  figure(1)
196  contourf(rph_time.t1(1:400),rph_time.t3(1:400),...
197  real(rph_time.response_time(1:400,1:400)),10)
198
199  clear rph_time
200  %% Create NonRephasing Time plot
201
202  nr_time = load('fifteenLevel_test_t2-200_nrt.mat');
203
204  figure(2)
205  contourf(nr_time.t1(1:400),nr_time.t3(1:400),...
206  nr_time.response_time(1:400,1:400),10)
207  axis square
208
209  clear nr_time
```

```
210
211  %% Plot rephasing frequency domain
212
213  rf = load('fifteenLevel_test_t2-200_rf.mat');
214
215  % Construct Frequency Axes
216  f = 0.25/2*linspace(0,1,2048/2+1)/(2.998e-5);
217
218  figure(3)
219  contourf(rf.freq,rf.freq,rf.response_freq,10)
220  clear rf
221
222  %% Plot Non-Rephasing Spectrum
223
224  nrf = load('fifteenLevel_test_t2-200_nrf.mat');
225
226  figure(4)
227  contourf(nrf.freq,nrf.freq,nrf.response_freq,10)
228  axis square
229  clear nrf
230
231  %% Plot 2DIR Correlation Spectrum
232
233  cspec1 = load('fifteenLevel_test_t2-0_abs.mat');
234
235  z = [-65e3:4e3:65e3];
236
237  figure(5)
238  contourf(cspec1.freq(1194:1265),cspec1.freq(1194:1265),...
```

```
239  cspec1.abs1(1194:1265,1194:1265),15)
240  xlim([460 650])
241  ylim([460 650])
242  axis square
243  set(gca, 'Tickdir', 'out');
244  set(gca, 'FontWeight','bold')
245  set(gca,'FontSize',16)
246  set(gca, 'FontName','Arial')
247  set(gca,'TickLength', [0.03, 0.01])
248  set(gca,'LineWidth', 2)
249  xlabel('\omega_{1}/2\pi (cm^{-1})')
250  ylabel('\omega_{3}/2\pi (cm^{-1})')
251  title('ZZZZ, \tau_2 = 0')
```

A.3.2 *Lineshape Functions*

The lineshape functions used to produce Figure 1.9 all follow that same format as the lineshape functions shown in Sections A.1.2 – A.1.7. Figure 1.9 used a 'fac' value of 2000 for all four modes for all possible lineshape functions.

A.4 FITTING 2D IR SIMULATIONS TO EXPERIMENTAL SPECTRA

Following are the fitting files used to fit the FePtFe spectra in Chapter 2. They can be used as an example for setting up any fitting routine using this simulation code. There is an outside file where the parameters that are to be fit are defined and any constraints on them are placed. Then there is the function Matlab calls to which the experimental data is compared. Finally, this section has the file I used to constrain the angles between all four transition dipole moments.

A.4.1 *fit_2D_FePtFe_angles_relative.m*

This is the main front file where parameters to be fit are defined and the fitting function is called. This is also where all the plots to compare the simulated and experimental spectra are produced.

```
1    clear all
2    clc %clear command window
3
4    global ym ys expPar energy
5
6    % Set Path - adds the path to the specPack3 fundtion files
7    addpath('C:\Users\Karla\Documents\Documents\Grad School
     Work\Research\tests\Mikes 2D trimer data\Fitting\specPack3')
8
9    %timing starts
10   tic
11   starttime = fix(clock)
12
13   %This section loads experimental parameters and makes sure the
     frequency
14   %variables match the spectra files.  w1 and w3 are the freq axes, and
     the
15   %c150_x variables are the spectra for the parallel and cross spectra
     for
16   %the 150 fs tau2 point.
17   load w3new
18   load ('w3_5220.txt')
19   w3_p=w3_5220(2:63);
20   w3_c=w3_5220(2:63);
21   load w1new
```

```
22  w1_p=w1new(46:178);
23  w1_c=w1new(46:178);
24  load c150new_p
25  load c150new_p2
26  load c150new_c
27  load c150new_c2
28
29  %Makes a 2D freq matrix that experimental spectra are interpolated to
30  [w1new1,w3new1]=meshgrid(w1new(46:178),w3new);
31
32  %Makes a 2D freq matrix from the original freq axes for the
    experimental
33  %spectra
34  [w1P,w3P]=meshgrid(w1_p,w3_p);
35  [w1C,w3C]=meshgrid(w1_c,w3_c);
36
37  %Averages the 2 spectra from each polarization, and then interpolates
    those
38  %new matrices to the new w1new1 and w3new1 freq axes
39  y1a = (c150new_p(:,46:178)+c150new_p2(:,46:178))/2;%this is the matrix
    of 2D spectra to fit
40  y2a = (c150new_c(:,46:178)+c150new_c2(:,46:178))/2;
41  y1 = interp2(w1new1,w3new1,y1a,w1P,w3P);
42  y2 = interp2(w1new1,w3new1,y2a,w1C,w3C);
43  ym(:,:,1)=(y1./max(max(y1)))+0.01;
44  ym(:,:,2)=y2./max(max(y1));
45  ym(:,106:end,:)=ym(:,106:end,:).*4;
46  ym(50:end,:,:)=ym(50:end,:,:).*4;
47
```

```
48  %Takes slices from each averaged experimental spectra at the freq for
    each
49  %mode and puts them into a single matrix that the simulated spectra can
    be
50  %also be fit to if it gets difficult to fit the smaller dipole moments
51  ys(:,1)=y1(:,74)./max(y1(:,74));
52  ys(:,2)=y1(:,66)./max(y1(:,66));
53  ys(:,3)=y1(:,84)./max(y1(:,84));
54  ys(:,4)=y1(:,115)./max(y1(:,115));
55  ys(:,5)=y2(:,74)./max(y2(:,74));
56  ys(:,6)=y2(:,66)./max(y2(:,66));
57  ys(:,7)=y2(:,84)./max(y2(:,84));
58  ys(:,8)=y2(:,115)./max(y2(:,115));
59
60
61  name = 'FePtFe_fit2_angles'; %what you want your simulated files to be
    called
62  %this is user-defined
63
64  tau2 = 150; %the tau2 point of the spectra you're fitting the
    simulation too
65
66  dor=2.0e-5; %diffusion coefficient
67
68  %these are experimental parameters that are passed to the fitting
    function
69  expPar.w1P=w1P;
70  expPar.w3P=w3P;
71  expPar.w1C=w1C;
```

```
72  expPar.w3C=w3C;
73  expPar.name=name;
74  expPar.tau2=tau2;
75  expPar.dor=dor;
76
77
78  %This is the section where the simulation parameters are defined.  Any
    or
79  %all of these might be fit.  This examples is only fitting the dipole
80  %moments and the relative angles between the transition dipoles
81  A_t=0.0022;
82  A_r=0.0031;
83  A_a=0.007;  %the amplitudes for the correlation functions
84  A_b=0.0021;
85
86  tau_t=540;
87  tau_r=1035;
88  tau_a=4990;  %the timescales for the correlation functions
89  tau_b=251;
90
91  tau_tr=192;
92  tau_ta=1320;
93  tau_tb=177;
94  tau_ra=1100;
95  tau_rb=523;
96  tau_ab=1011;
97
98  T2=389200; %the lorentzian component to each lineshape for each mode.
    One of these can be defined for each mode.
```

```
99
100  % Energies (in cm^-1)
101  t=560; %4 vibrations: t,r,a,b
102  r=550;
103  a=574;
104  b=616;
105  an_t=25; %anharmonicities
106  an_r=21.5;
107  an_a=19;
108  an_b=24;
109  an_tr=2.2;
110  an_ta=13;
111  an_tb=9.6;
112  an_ra=9.3;
113  an_rb=2.01;
114  an_ba=10.6;
115
116  energ.t=t;
117  energ.r=r;
118  energ.a=a;
119  energ.b=b;
120
121  m_t=0.732; %dipole moment amplitudes
122  % m_r=1;
123  m_a=0.313;
124  m_b=0.423;
125
126  %relative angles between most dipole transitions.  For this example,
     the
```

```
%bridge was defined to be along the z axis, and the radial was
constrained
%to the xz plane.
theta_rt = 89;
theta_bt = 3;
theta_ar = 82;
theta_br = 88;
theta_ba = 1;

%In case the amplitude between the parallel and crossed spectra isn't
what
%you expected due to experimental issues.  This can only be used when
you
%have more than two coupled modes if you are trying to extract relative
%angles btwn transition dipoles
cross_corr = 0.882;

%the parameters that are currently being fit
f0 = [m_t m_a m_b ...
theta_rt theta_bt theta_ar theta_br theta_ba];

%the upper and lower bounds placed on the parameters you're fitting
lb=[0.4, 0.2, 0.3,...
50, 0, 50, 50, 0];
ub=[0.8, 0.4, 0.5...
90, 40, 90, 90, 40];

%the options on for the fit
```

```
152 options=optimoptions(@fmincon,'Algorithm','interior-
    point','MaxFunEvals',1000,'MaxIter',2000,'Display','iter','tolfun',1E-
    100);
153
154 %Calling the fitting function with the parameters you're fitting, and
    their
155 %bounds, as well as any non-linear constraints placed on the parameters
156
    [f,resnorm,residual,exitflag,output,lambda,jacobian]=fmincon('fifteenLe
    velFit_angles_relative',f0,[],[],[],[],lb,ub,@angle_const,options);
157
158 %timing ends
159 Toc
160
161 %in case something happens and the fit doesn't finish.  This still
    saves
162 %the latest fit parameters
163 save f f
164 save output output
165
166 theta_rt = f(4);
167 theta_bt = f(5);
168 theta_ar = f(6);
169 theta_br = f(7);
170 theta_ba = f(8);
171
172 %calculates the final angle that wasn't necessary to fit
```

```
173  theta_at = acosd(cosd(theta_ba)*cosd(theta_bt) + ((cosd(theta_ar)-
     (cosd(theta_ba)*cosd(theta_br)))*(cosd(theta_rt)-
     (cosd(theta_br)*cosd(theta_bt))))/((sind(theta_br))^2)...
174  + sind(theta_ba)*sind(theta_bt)*(sqrt(1-((cosd(theta_ar)-
     (cosd(theta_ba)*cosd(theta_br)))/(sind(theta_ba)*sind(theta_br)))^2)*sq
     rt(1-((cosd(theta_rt)-
     (cosd(theta_br)*cosd(theta_bt)))/(sind(theta_br)*sind(theta_bt)))^2)))
175
176  %calculates confidence intervals on the fit parameters (kind of)
177  ci=nlparci(f,residual,'jacobian',jacobian,'alpha',0.1);
178  cihalf=((ci(:,2)-ci(:,1))./2)';
179
180  save ci ci
181  save cihalf cihalf
182
183  load yfit3 %matrix of simulated spectra along the same axes as the
     experimental spectra
184
185
186  %A whole bunch of figures to compare the fit to the experimental
     spectra
187  yfit4=yfit3;
188  yfit4(:,106:end,:)=yfit3(:,106:end,:).*4;
189  yfit4(50:end,:,:)=yfit3(50:end,:,:).*4;
190
191  yfits(:,1)=yfit4(:,74,1)./max(yfit4(:,74,1));
192  yfits(:,2)=yfit4(:,66,1)./max(yfit4(:,66,1));
193  yfits(:,3)=yfit4(:,84,1)./max(yfit4(:,84,1));
194  yfits(:,4)=yfit4(:,115,1)./max(yfit4(:,115,1));
```

```
yfits(:,5)=yfit4(:,74,2)./max(yfit4(:,74,2));
yfits(:,6)=yfit4(:,66,2)./max(yfit4(:,66,2));
yfits(:,7)=yfit4(:,84,2)./max(yfit4(:,84,2));
yfits(:,8)=yfit4(:,115,2)./max(yfit4(:,115,2));

ym(:,:,2)=cross_corr*ym(:,:,2);

figure
plot(w3_p,ys(:,1),'o',w3_p,yfits(:,1))
title('trans, zzzz')

figure
plot(w3_p,ys(:,2),'o',w3_p,yfits(:,2))
title('radial, zzzz')

figure
plot(w3_p,ys(:,3),'o',w3_p,yfits(:,3))
title('axial, zzzz')

figure
plot(w3_p,ys(:,4),'o',w3_p,yfits(:,4))
title('bridge, zzzz')

figure
plot(w3_c,ys(:,5),'o',w3_c,yfits(:,5))
title('trans, yyzz')

figure
plot(w3_c,ys(:,6),'o',w3_c,yfits(:,6))
```

```
224  title('radial, yyzz')
225
226  figure
227  plot(w3_c,ys(:,7),'o',w3_c,yfits(:,7))
228  title('axial, yyzz')
229
230  figure
231  plot(w3_c,ys(:,8),'o',w3_c,yfits(:,8))
232  title('bridge, yyzz')
233
234
235
236  figure
237  plot(w3_p,yfit4(:,74,1),w3_p,ym(:,74,1),'o')
238  title('trans, zzzz')
239
240  figure
241  plot(w3_p,yfit4(:,66,1),w3_p,ym(:,66,1),'o')
242  title('radial, zzzz')
243
244  figure
245  plot(w3_p,yfit4(:,84,1),w3_p,ym(:,84,1),'o')
246  title('axial, zzzz')
247
248  figure
249  plot(w3_p,yfit4(:,115,1),w3_p,ym(:,115,1),'o')
250  title('bridge, zzzz')
251
252
```

```
figure
plot(w3_c,yfit4(:,74,2),w3_c,ym(:,74,2),'o')
title('trans, yyzz')

figure
plot(w3_c,yfit4(:,66,2),w3_c,ym(:,66,2),'o')
title('radial, yyzz')

figure
plot(w3_c,yfit4(:,84,2),w3_c,ym(:,84,2),'o')
title('axial, yyzz')

figure
plot(w3_c,yfit4(:,115,2),w3_c,ym(:,115,2),'o')
title('bridge, yyzz')


figure('Name','Fit')
subplot(1,2,1)
contourf(w1_p,w3_p,yfit4(:,:,1),15)
xlabel('\omega_1/2\pi (cm^{-1})')
ylabel('\omega_3/2\pi (cm^{-1})')
title('\tau_2 = 150, zzzz')
axis square
subplot(1,2,2)
contourf(w1_c,w3_c,yfit4(:,:,2),15)
xlabel('\omega_1/2\pi (cm^{-1})')
ylabel('\omega_3/2\pi (cm^{-1})')
title('\tau_2 = 150, yyzz')
```

```
282  axis square
283
284  figure('Name','Experimental')
285  subplot(1,2,1)
286  contourf(w1_p,w3_p,ym(:,:,1),15)
287  xlabel('\omega_1/2\pi (cm^{-1})')
288  ylabel('\omega_3/2\pi (cm^{-1})')
289  title('\tau_2 = 150, zzzz')
290  axis square
291  subplot(1,2,2)
292  contourf(w1_c,w3_c,ym(:,:,2),15)
293  xlabel('\omega_1/2\pi (cm^{-1})')
294  ylabel('\omega_3/2\pi (cm^{-1})')
295  title('\tau_2 = 150, yyzz')
296  axis square
```

A.4.2 *angle_const.m*

This is the file for some necessary nonlinear constraints on the values for the relative angles between transition dipole moments. If those angles are currently being fit, this function needs to be called when calling the fitting function as it is in the example from Section A.4.1.

```
1    function [c, ceq] = angle_const(f)
2
3    %the angles that are being fit.  Make sure the f(#) values are updated
     if
4    %the number of parameters you're fitting changes.
5    theta_rt = f(4);
6    theta_bt = f(5);
7    theta_ar = f(6);
```

```
8     theta_br = f(7);
9     theta_ba = f(8);
10
11    %These inequalties have to be true from the geometry btwn 4 vectors
12    ineq1 = ((cosd(theta_ar) -
      cosd(theta_ba)*cosd(theta_br))/(sind(theta_ba)*sind(theta_br)))^2;
13    ineq2 = ((cosd(theta_rt) -
      cosd(theta_br)*cosd(theta_bt))/(sind(theta_br)*sind(theta_bt)))^2;
14
15    %Nonlinear inequality constraints (greater than zero)
16    c = [ineq1-1 ; ineq2-1];
17
18    %Nonlinear equality constraints
19    ceq = [];
```

A.4.3 *fifteenLevelFit_angles_relative.m*

This is the fitting function that Matlab calls for the example given in Section A.4.1. It is currently only fitting the transition dipole moment amplitudes and the relative angles between them. Make sure all of the correct variables are made global, and that the correct values are updated when changing the parameters that are being fit.

```
1     function retf = fifteenLevelFit_angles_relative(fitvar)
2     %15 Level Test - Four Coupled Dipoles
3     % This is an example for a 15 level system.  See the
4     % sixLevelHomLimPolarComp.m file for more detailed comments on some
      sections
5
6     %All of the correlation function amplitudes and timescales need to be
7     %global to pass to the lineshape function files.  These parameters also
```

```
8    %need to be made global in the lineshape function files.
9    global ym expPar energ A_t A_r A_a A_b tau_t tau_r tau_a tau_b tau_tr
     tau_ta tau_tb tau_ra tau_rb tau_ab T2
10
11   % Define Lineshape
     Parameters%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
12   % Anything that isn't being fit should have an actual number.  Anything
13   % that is being fit should have a value of fitvar(#) for this example.
     The
14   % # should correspond to the correct position that you defined in the
     main
15   % front file where you called this fitting function.
16   A_t=0.0021;
17   A_r=0.0032;
18   A_a=0.0080;
19   A_b=0.0022;
20
21   tau_t=540;
22   tau_r=1030;
23   tau_a=4989;
24   tau_b=270;
25
26   tau_tr=200;
27   tau_ta=1313;
28   tau_tb=200;
29   tau_ra=1100;
30   tau_rb=500;
31   tau_ab=1000;
32
```

```
33  T2=389200;
34
35  cross_corr = 0.87;
36  y=ym;
37  y(:,:,2)=cross_corr*y(:,:,2); % 'cross_corr' can be a value that you
    fit
38
39  % Define system parameters%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
40
41  %Energies (in cm^-1)
42  %they are shifted down in energy by 1500 cm-1 to reduce calc time
43  t=energ.t; %4 vibrations: t,r,a,b
44  r=energ.r;
45  a=energ.a;
46  b=energ.b;
47  t2=2*t-24; %overtones.  All of the anharmonicities can also be fit and
    the values can be replaced with fitvar(#)
48  r2=2*r-21;
49  a2=2*a-19;
50  b2=2*b-24;
51  tr=t+r-2.2; %combination bands
52  ta=t+a-13.1;%20;
53  tb=t+b-8.5;
54  ra=a+r-9;%3;
55  rb=b+r-2;
56  ba=b+a-8.3;
57
58  %putting the above energies into a vector
59  energies = [0 t r a b t2 r2 a2 b2 tr ta tb ra rb ba];
```

```
60
61    %mu values are being fit in this example
62    m_t=fitvar(1);%0.732;
63    m_r=1;
64    m_a=fitvar(2);%0.313;
65    m_b=fitvar(3);%0.423;
66
67    rt2 = sqrt(2.0);
68    m2t=rt2*m_t;
69    m2r=rt2*m_r;
70    m2a=rt2*m_a;
71    m2b=rt2*m_b;
72
73    mu = [ 0.0 m_t m_r m_a m_b 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0;...
      %1
74    m_t 0.0 0.0 0.0 0.0 m2t 0.0 0.0 0.0 m_r m_a m_b 0.0 0.0 0.0;... %2
75    m_r 0.0 0.0 0.0 0.0 0.0 m2r 0.0 0.0 m_t 0.0 0.0 m_a m_b 0.0;... %3
76    m_a 0.0 0.0 0.0 0.0 0.0 0.0 m2a 0.0 0.0 m_t 0.0 m_r 0.0 m_b;... %4
77    m_b 0.0 0.0 0.0 0.0 0.0 0.0 0.0 m2b 0.0 0.0 m_t 0.0 m_r m_a;... %5
78    0.0 m2t 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0;... %6
79    0.0 0.0 m2r 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0;... %7
80    0.0 0.0 0.0 m2a 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0;... %8
81    0.0 0.0 0.0 0.0 m2b 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0;... %9
82    0.0 m_r m_t 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0;... %10
83    0.0 m_a 0.0 m_t 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0;... %11
84    0.0 m_b 0.0 0.0 m_t 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0;... %12
85    0.0 0.0 m_a m_r 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0;... %13
86    0.0 0.0 m_b 0.0 m_r 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0;... %14
87    0.0 0.0 0.0 m_b m_a 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0];   %15
```

```
88
89   nstates = 15;
90   for i=1:nstates
91   for j=1:nstates
92   for k = 1:3
93   muDir(i,j,k) = 0.0;
94   end
95   end
96   end
97
98   %angles are also being fit in this example
99   theta_rt = fitvar(4);
100  theta_bt = fitvar(5);
101  theta_ar = fitvar(6);
102  theta_br = fitvar(7);
103  theta_ba = fitvar(8);
104
105  %Using geometry to assign the appropriate amplitude from each
     transition
106  %dipole moment to the correct axes. The bridge mode is defined along
     the
107  %z-axis, and the radial mode is defined to the xz plane.
108  muDir(1,2,1) = (cosd(theta_rt)-
     cosd(theta_bt)*cosd(theta_br))./sind(theta_br);
109  muDir(2,1,1) = (cosd(theta_rt)-
     cosd(theta_bt)*cosd(theta_br))./sind(theta_br);
110  muDir(1,2,2) = sind(theta_bt)*sqrt(1-((cosd(theta_rt)-
     cosd(theta_br)*cosd(theta_bt))./(sind(theta_br)*sind(theta_bt)))^2);
```

```
111  muDir(2,1,2) = sind(theta_bt)*sqrt(1-((cosd(theta_rt)-
     cosd(theta_br)*cosd(theta_bt))./(sind(theta_br)*sind(theta_bt)))^2);
112  muDir(1,2,3) = cosd(theta_bt);
113  muDir(2,1,3) = cosd(theta_bt);
114
115  muDir(1,3,1) = sind(theta_br);
116  muDir(3,1,1) = sind(theta_br);
117  muDir(1,3,3) = cosd(theta_br);
118  muDir(3,1,3) = cosd(theta_br);
119
120  muDir(1,4,1) = (cosd(theta_ar)-
     cosd(theta_ba)*cosd(theta_br))./sind(theta_br);
121  muDir(4,1,1) = (cosd(theta_ar)-
     cosd(theta_ba)*cosd(theta_br))./sind(theta_br);
122  muDir(1,4,2) = sind(theta_ba)*sqrt(1-((cosd(theta_ar)-
     cosd(theta_ba)*cosd(theta_br))./(sind(theta_ba)*sind(theta_br)))^2);
123  muDir(4,1,2) = sind(theta_ba)*sqrt(1-((cosd(theta_ar)-
     cosd(theta_ba)*cosd(theta_br))./(sind(theta_ba)*sind(theta_br)))^2);
124  muDir(1,4,3) = cosd(theta_ba);
125  muDir(4,1,3) = cosd(theta_ba);
126
127  muDir(1,5,3) = 1.0;
128  muDir(5,1,3) = 1.0;
129
130  muDir(2,6,1) = (cosd(theta_rt)-
     cosd(theta_bt)*cosd(theta_br))./sind(theta_br);
131  muDir(6,2,1) = (cosd(theta_rt)-
     cosd(theta_bt)*cosd(theta_br))./sind(theta_br);
```

```
132  muDir(2,6,2) = sind(theta_bt)*sqrt(1-((cosd(theta_rt)-
     cosd(theta_br)*cosd(theta_bt))./(sind(theta_br)*sind(theta_bt)))^2);
133  muDir(6,2,2) = sind(theta_bt)*sqrt(1-((cosd(theta_rt)-
     cosd(theta_br)*cosd(theta_bt))./(sind(theta_br)*sind(theta_bt)))^2);
134  muDir(2,6,3) = cosd(theta_bt);
135  muDir(6,2,3) = cosd(theta_bt);
136
137  muDir(3,7,1) = sind(theta_br);
138  muDir(7,3,1) = sind(theta_br);
139  muDir(3,7,3) = cosd(theta_br);
140  muDir(7,3,3) = cosd(theta_br);
141
142  muDir(4,8,1) = (cosd(theta_ar)-
     cosd(theta_ba)*cosd(theta_br))./sind(theta_br);
143  muDir(8,4,1) = (cosd(theta_ar)-
     cosd(theta_ba)*cosd(theta_br))./sind(theta_br);
144  muDir(4,8,2) = sind(theta_ba)*sqrt(1-((cosd(theta_ar)-
     cosd(theta_ba)*cosd(theta_br))./(sind(theta_ba)*sind(theta_br)))^2);
145  muDir(8,4,2) = sind(theta_ba)*sqrt(1-((cosd(theta_ar)-
     cosd(theta_ba)*cosd(theta_br))./(sind(theta_ba)*sind(theta_br)))^2);
146  muDir(4,8,3) = cosd(theta_ba);
147  muDir(8,4,3) = cosd(theta_ba);
148
149  muDir(5,9,3) = 1.0;
150  muDir(9,5,3) = 1.0;
151
152  muDir(2,10,1) = sind(theta_br);
153  muDir(10,2,1) = sind(theta_br);
154  muDir(2,10,3) = cosd(theta_br);
```

```
155  muDir(10,2,3) = cosd(theta_br);
156
157  muDir(2,11,1) = (cosd(theta_ar)-
     cosd(theta_ba)*cosd(theta_br))./sind(theta_br);
158  muDir(11,2,1) = (cosd(theta_ar)-
     cosd(theta_ba)*cosd(theta_br))./sind(theta_br);
159  muDir(2,11,2) = sind(theta_ba)*sqrt(1-((cosd(theta_ar)-
     cosd(theta_ba)*cosd(theta_br))./(sind(theta_ba)*sind(theta_br)))^2);
160  muDir(11,2,2) = sind(theta_ba)*sqrt(1-((cosd(theta_ar)-
     cosd(theta_ba)*cosd(theta_br))./(sind(theta_ba)*sind(theta_br)))^2);
161  muDir(2,11,3) = cosd(theta_ba);
162  muDir(11,2,3) = cosd(theta_ba);
163
164  muDir(2,12,3) = 1.0;
165  muDir(12,2,3) = 1.0;
166
167  muDir(3,10,1) = (cosd(theta_rt)-
     cosd(theta_bt)*cosd(theta_br))./sind(theta_br);
168  muDir(10,3,1) = (cosd(theta_rt)-
     cosd(theta_bt)*cosd(theta_br))./sind(theta_br);
169  muDir(3,10,2) = sind(theta_bt)*sqrt(1-((cosd(theta_rt)-
     cosd(theta_br)*cosd(theta_bt))./(sind(theta_br)*sind(theta_bt)))^2);
170  muDir(10,3,2) = sind(theta_bt)*sqrt(1-((cosd(theta_rt)-
     cosd(theta_br)*cosd(theta_bt))./(sind(theta_br)*sind(theta_bt)))^2);
171  muDir(3,10,3) = cosd(theta_bt);
172  muDir(10,3,3) = cosd(theta_bt);
173
174  muDir(3,13,1) = (cosd(theta_ar)-
     cosd(theta_ba)*cosd(theta_br))./sind(theta_br);
```

```
175  muDir(13,3,1) = (cosd(theta_ar)-
     cosd(theta_ba)*cosd(theta_br))./sind(theta_br);
176  muDir(3,13,2) = sind(theta_ba)*sqrt(1-((cosd(theta_ar)-
     cosd(theta_ba)*cosd(theta_br))./(sind(theta_ba)*sind(theta_br)))^2);
177  muDir(13,3,2) = sind(theta_ba)*sqrt(1-((cosd(theta_ar)-
     cosd(theta_ba)*cosd(theta_br))./(sind(theta_ba)*sind(theta_br)))^2);
178  muDir(3,13,3) = cosd(theta_ba);
179  muDir(13,3,3) = cosd(theta_ba);
180
181  muDir(3,14,3) = 1.0;
182  muDir(14,3,3) = 1.0;
183
184  muDir(4,11,1) = (cosd(theta_rt)-
     cosd(theta_bt)*cosd(theta_br))./sind(theta_br);
185  muDir(11,4,1) = (cosd(theta_rt)-
     cosd(theta_bt)*cosd(theta_br))./sind(theta_br);
186  muDir(4,11,2) = sind(theta_bt)*sqrt(1-((cosd(theta_rt)-
     cosd(theta_br)*cosd(theta_bt))./(sind(theta_br)*sind(theta_bt)))^2);
187  muDir(11,4,2) = sind(theta_bt)*sqrt(1-((cosd(theta_rt)-
     cosd(theta_br)*cosd(theta_bt))./(sind(theta_br)*sind(theta_bt)))^2);
188  muDir(4,11,3) = cosd(theta_bt);
189  muDir(11,4,3) = cosd(theta_bt);
190
191  muDir(4,13,1) = sind(theta_br);
192  muDir(13,4,1) = sind(theta_br);
193  muDir(4,13,3) = cosd(theta_br);
194  muDir(13,4,3) = cosd(theta_br);
195
196  muDir(4,15,3) = 1.0;
```

```
197  muDir(15,4,3) = 1.0;
198
199  muDir(5,12,1) = (cosd(theta_rt)-
     cosd(theta_bt)*cosd(theta_br))./sind(theta_br);
200  muDir(12,5,1) = (cosd(theta_rt)-
     cosd(theta_bt)*cosd(theta_br))./sind(theta_br);
201  muDir(5,12,2) = sind(theta_bt)*sqrt(1-((cosd(theta_rt)-
     cosd(theta_br)*cosd(theta_bt))./(sind(theta_br)*sind(theta_bt)))^2);
202  muDir(12,5,2) = sind(theta_bt)*sqrt(1-((cosd(theta_rt)-
     cosd(theta_br)*cosd(theta_bt))./(sind(theta_br)*sind(theta_bt)))^2);
203  muDir(5,12,3) = cosd(theta_bt);
204  muDir(12,5,3) = cosd(theta_bt);
205
206  muDir(5,14,1) = sind(theta_br);
207  muDir(14,5,1) = sind(theta_br);
208  muDir(5,14,3) = cosd(theta_br);
209  muDir(14,5,3) = cosd(theta_br);
210
211  muDir(5,15,1) = (cosd(theta_ar)-
     cosd(theta_ba)*cosd(theta_br))./sind(theta_br);
212  muDir(15,5,1) = (cosd(theta_ar)-
     cosd(theta_ba)*cosd(theta_br))./sind(theta_br);
213  muDir(5,15,2) = sind(theta_ba)*sqrt(1-((cosd(theta_ar)-
     cosd(theta_ba)*cosd(theta_br))./(sind(theta_ba)*sind(theta_br)))^2);
214  muDir(15,5,2) = sind(theta_ba)*sqrt(1-((cosd(theta_ar)-
     cosd(theta_ba)*cosd(theta_br))./(sind(theta_ba)*sind(theta_br)))^2);
215  muDir(5,15,3) = cosd(theta_ba);
216  muDir(15,5,3) = cosd(theta_ba);
217
```

```
218
219  %all population starts out in the v = 0 state in this example
220  rho0 = [1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0];
221
222  hfun = cell(3);
223  for i = 1:nstates
224  for j = 1:nstates
225  hfun{i,j}=@g0;
226  end
227  end
228
229  %All of the lineshape functions
230  hfun{2,2}=@gttL;
231  hfun{2,6}=@gt2tL;
232  hfun{6,2}=@gt2tL;
233  hfun{3,3}=@grrL;
234  hfun{3,7}=@gr2rL;
235  hfun{7,3}=@gr2rL;
236  hfun{4,4}=@gaaL;
237  hfun{4,8}=@ga2aL;
238  hfun{8,4}=@ga2aL;
239  hfun{5,5}=@gbbL;
240  hfun{5,9}=@gb2bL;
241  hfun{9,5}=@gb2bL;
242  hfun{6,6}=@g2t2tL;
243  hfun{7,7}=@g2r2rL;
244  hfun{8,8}=@g2a2aL;
245  hfun{9,9}=@g2b2bL;
246
```

```
247  hfun{2,3}=@gtrL;
248  hfun{3,2}=@gtrL;
249  hfun{2,10}=@gttrL;
250  hfun{10,2}=@gttrL;
251  hfun{3,10}=@grtrL;
252  hfun{10,3}=@grtrL;
253  hfun{10,10}=@gtrtrL;
254
255  hfun{2,4}=@gtaL;
256  hfun{4,2}=@gtaL;
257  hfun{2,11}=@gttaL;
258  hfun{11,2}=@gttaL;
259  hfun{4,11}=@gataL;
260  hfun{11,4}=@gataL;
261  hfun{11,11}=@gtataL;
262
263  hfun{2,5}=@gtbL;
264  hfun{5,2}=@gtbL;
265  hfun{2,12}=@gttbL;
266  hfun{12,2}=@gttbL;
267  hfun{5,12}=@gbtbL;
268  hfun{12,5}=@gbtbL;
269  hfun{12,12}=@gtbtbL;
270
271  hfun{3,4}=@graL;
272  hfun{4,3}=@graL;
273  hfun{3,13}=@grraL;
274  hfun{13,3}=@grraL;
275  hfun{4,13}=@garaL;
```

```
276  hfun{13,4}=@garaL;
277  hfun{13,13}=@graraL;
278
279  hfun{3,5}=@grbL;
280  hfun{5,3}=@grbL;
281  hfun{3,14}=@grrbL;
282  hfun{14,3}=@grrbL;
283  hfun{5,14}=@gbrbL;
284  hfun{14,5}=@gbrbL;
285  hfun{14,14}=@grbrbL;
286
287  hfun{4,5}=@gabL;
288  hfun{5,4}=@gabL;
289  hfun{4,15}=@gaabL;
290  hfun{15,4}=@gaabL;
291  hfun{5,15}=@gbabL;
292  hfun{15,5}=@gbabL;
293  hfun{15,15}=@gababL;
294
295  dor = expPar.dor;
296
297  % Initialize structs%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
298
299  sys = initSystem( nstates, energies, mu, muDir, rho0, hfun, dor);
300
301  expr = initExperiment([550 550 550 550], '2DIR', 'YYZZ');
302  opts = sp_SetOptions('tstep',8.0); %Open sp_SetOptions and manually
     adjust these values
303
```

```
304  % Call 2D calculation function%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
305  tau2 = expPar.tau2;
306  calc2DImpulse([expPar.name,'_YYZZ'],sys,expr,tau2,opts);
307
308  % Initialize structs%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
309  sys = initSystem( nstates, energies, mu, muDir, rho0, hfun, dor);
310
311  expr = initExperiment([550 550 550 550], '2DIR', 'ZZZZ');
312
313  opts = sp_SetOptions('tstep',8.0);
314
315  % Call 2D calculation
     function%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
316  tau2 = expPar.tau2;
317  calc2DImpulse([expPar.name,'_ZZZZ'],sys,expr,tau2,opts);
318
319
320  %Loads the simulated spectra just calculated
321  c = load([expPar.name,'_YYZZ','_t2-150_abs']);
322  p = load([expPar.name,'_ZZZZ','_t2-150_abs']);
323
324  %Makes a 2D matrix of the freq axes for the simulated spectra (the
     +1500 is
325  %to bring the freq axis back to the correct range - see line 42)
326  [fx,fy]=meshgrid(c.freq(1025:end)+1500);
327
328  %Puts simulated spectra onto the same freq axes as the experimental
     spectra
```

```
329
    yfit1=interp2(fx,fy,real(p.abs1(1025:end,1025:end)),expPar.w1P,expPar.w
    3P);
330
    yfit2=interp2(fx,fy,real(c.abs1(1025:end,1025:end)),expPar.w1C,expPar.w
    3C);
331 yfit3(:,:,1)=yfit1./max(max(yfit1));
332 yfit3(:,:,2)=yfit2./max(max(yfit1));
333
334 yfit4=yfit3;
335
336 %trying to get a better fit on the bridge region by making its
    amplitude
337 %larger
338 yfit4(:,106:end,:)=yfit3(:,106:end,:).*4;
339 yfit4(50:end,:,:)=yfit3(50:end,:,:).*4;
340
341
342 save yfit3 yfit3
343 save fitvar fitvar
344
345 %Taking the difference btwn the simulated and experimental spectra
346 diff_f=(yfit4-y);
347
348 sq_diff = diff_f.^2;
349
350 %summing up the square of the difference along all three dimensions to
    get
351 %a single value to try to minimize
```

```
352   retf = sum(sum(sum(sq_diff)));
```

A.4.4 *gttL.m*

This is an example of the lineshape functions used FePtFe. The other three modes had lineshape functions with similar formats, but with their own A and tau values. These would include the grrL.m, gaaL.m, and gbbL.m functions.

```
1     function y = gttL(t)
2
3     global A_t tau_t T2
4
5     %If you aren't using the fitting files with defined global variables,
      you
6     %need to explicitly define these values.
7     % A_t=0.001;
8     % tau_t=1000;
9     % T2=135; %You can define different T2 values for each mode if they
      have
10    % different Lorentzian percentages
11
12    y = A_t^2*tau_t^2*(exp(-abs(t)/tau_t)+abs(t)/tau_t-1)...
13    +(abs(t)/T2)*(0.5);
14
15
16    end
```

A.4.5 *gt2tL.m*

The other three modes had lineshape functions with similar formats, but again with their own A and tau values. These would include the gr2rL.m, ga2aL.m, and gb2bL.m functions.

```
1    function y = gt2tL( t )
2
3    global A_t tau_t T2
4
5    %If you aren't using the fitting files with defined global variables,
     you
6    %need to explicitly define these values.
7    % A_t=0.001;
8    % tau_t=1000;
9    % T2t=135;
10
11
12   y = 2*A_t^2*tau_t^2*(exp(-abs(t)/tau_t)+abs(t)/tau_t-1)...
13   +2*(abs(t)./T2)*(0.5);
14
15   end
```

A.4.6 *g2t2tL.m*

The other three modes had lineshape functions with similar formats, but again with their own A and tau values. These would include the g2r2rL.m, g2a2aL.m, and g2b2bL.m functions.

```
1    function y = g2t2tL( t )
2
3    global A_t tau_t T2
4
5    %If you aren't using the fitting files with defined global variables,
     you
6    %need to explicitly define these values.
7    % A_t=0.001;
8    % tau_t=1000;
```

```
9     % T2t=135;
10
11    y = 4*A_t^2*tau_t^2*(exp(-abs(t)/tau_t)+abs(t)/tau_t-1)...
12    +4*(abs(t)./T2)*(0.5);
13
14    end
```

A.4.7 *gtrL.m*

There will be five other lineshape functions with similar formats: gtaL.m, gtbL.m, graL.m, grbL.m, and gabL.m. Each of these will have two A values, the same as used in the lineshape functions in Sections A.4.5 – A.4.7, and a new tau value.

```
1     function y = gtrL( t )
2
3     global A_t A_r tau_tr T2
4
5     %If you aren't using the fitting files with defined global variables,
      you
6     %need to explicitly define these values.
7     % A_t=0.001;
8     % A_r=0.0018;
9     % tau_tr=1000;
10    % T2=135;
11
12
13    y = A_t*A_r*tau_tr^2*(exp(-abs(t)/tau_tr)+abs(t)/tau_tr-1)...
14    +(abs(t)./T2)*(0.5);
15
16    end
```

A.4.8 *gttrL.m*

There will be 11 other lineshape functions with similar formats. All values for this will have been used in other types of lineshape functions so make sure the correct variables are called.

```
1    function y = gttrL( t )
2
3    global A_t A_r tau_t tau_tr T2
4
5    %If you aren't using the fitting files with defined global variables,
     you
6    %need to explicitly define these values.
7    % A_t=0.001;
8    % A_r=0.0018;%0.0018
9    % tau_t=1000;
10   % tau_tr=1000;
11   % T2=6000;
12
13   y = A_t^2*tau_t^2.*(exp(-abs(t)./tau_t)+abs(t)./tau_t-1)...
14   +A_t*A_r*tau_tr^2.*(exp(-abs(t)./tau_tr)+abs(t)./tau_tr-1)...
15   +(abs(t)./T2)*(1);
16
17   end
```

A.4.9 *gtrtrL.m*

Five other lineshape functions follow the same format: gtataL.m, gtbtbL.m, graraL.m, grbrbL.m, and gababL.m. Again, all variables should match the values used in other lineshape functions.

```
1    function y = gtrtrL( t )
2
3    global A_t A_r tau_t tau_r tau_tr T2
```

```
4
5    %If you aren't using the fitting files with defined global variables,
     you
6    %need to explicitly define these values.
7    % A_t=0.001;
8    % A_r=0.0018;%0.0018
9    % tau_t=1000;
10   % tau_r=1800;
11   % tau_tr=1000;
12   % T2=6000;
13
14   y = A_t^2*tau_t^2.*(exp(-abs(t)./tau_t)+abs(t)./tau_t-1)...
15   +A_r^2*tau_r^2.*(exp(-abs(t)./tau_r)+abs(t)./tau_r-1)...
16   +2*A_t*A_r*tau_tr^2.*(exp(-abs(t)./tau_tr)+abs(t)./tau_tr-1)...
17   +(abs(t)./T2)*(2);
18
19   end
```

A.5 SPECPACK3

These code files are used for all types of systems with minimal changes needed between types. Updates need to be made to sp_SetOptions.m. Updates may need to be made to the threshold in buildRot.m. The function files are presented in alphabetical order.

A.5.1 *buildEcho.m*

This function file builds the echo (rephasing) response functions.

```
1    function rfn = buildEcho(sys, expr, rwaThresh, prefacThresh, l_poprlx)
2    %BUILDECHO construct photon echo response function
```

```
3     %  Description: This function takes system and experimental parameters,
      and
4     %  it constructs a response function for the ks = -k1+k2+k3 phase
      matching
5     %  condition
6     %
7     %  Parameters: sys - Structure containing system parameters
8     %              exp - Structure containing experimental parameters
9     %              rwaThresh - Threshold for resonance for RWA ( units =
      cm^-1)
10    %              prefacThresh - product of rho0 and dipoles is assumed
11    %                             to be 0 if smaller than prefacThresh
12    %              l_poprlx - logical controlling population relaxation
13    %
14    %  Returns: rfn - structure with response funciton parameters
15    %           rfn.nterm - number of non-zero terms in response function
16    %           rfn.rtype - array with rfn types
17    %           rfn.aa - array with 1st eigenstate for each function
18    %           rfn.bb - array with 2nd eigenstate for each function
19    %           rfn.cc - array with 3rd eigenstate for each function
20    %           rfn.dd - array with 4th eigenstate for each function
21    %           rfn.w1 - array with t1 frequency for each function
22    %           rfn.w2 - array with t2 frequency for each function
23    %           rfn.w3 - array with t3 frequency for each function
24    %           rfn.rho0 - population in state a at t = 0
25    %           rfn.T1_1 - array with population relaxation contrib to t1
26    %           rfn.T1_2 - array with population relaxation contrib to t2
27    %           rfn.T1_3 - array with population relaxation contrib to t3
28
```

```
29  rfn.nterm = 0;    % Number of terms in Response fucntion
30  sizeGuess = sys.nStates*6; % Guess of number of rfn terms
31
32  % Begin Timing
33  ticID = tic;
34
35  % Preallocated rfs parameter arrays
36  rfn.rtype = zeros(1,sizeGuess);
37  rfn.aa = zeros(1,sizeGuess);
38  rfn.bb = zeros(1,sizeGuess);
39  rfn.cc = zeros(1,sizeGuess);
40  rfn.dd = zeros(1,sizeGuess);
41  rfn.w1 = zeros(1,sizeGuess);
42  rfn.w2 = zeros(1,sizeGuess);
43  rfn.w3 = zeros(1,sizeGuess);
44  rfn.rho0 = zeros(1,sizeGuess);
45  rfn.nYTerms = zeros(1,sizeGuess);
46
47  %Checks for each type of response function, R1 through R8 (JPCA, 2003,
    107,
48  %5258-5279). Compares prefactors and differences btwn frequencies
    against
49  %prefacThresh and RWA threshold.
50  for a=1:sys.nStates
51  for b=1:sys.nStates
52  for c=1:sys.nStates
53  for d=1:sys.nStates
54  % Check for R1 Members
55  prefac = sys.mu(c,d)*sys.mu(b,c)*sys.mu(a,b)*sys.mu(a,d)*...
```

```
56   sys.rho0(a);
57   if (prefac > prefacThresh)
58   if (abs(sys.w(d,c) - expr.w(4)) <= rwaThresh)
59   if (abs(sys.w(d,b) - (expr.w(2) - expr.w(1))) ...
60   <= rwaThresh)
61   if (abs(sys.w(d,a) + expr.w(1)) <= rwaThresh)
62   rfn.nterm = rfn.nterm + 1;
63   rfn.rtype(rfn.nterm)= 1;
64   rfn.aa(rfn.nterm)= a;
65   rfn.bb(rfn.nterm)= b;
66   rfn.cc(rfn.nterm)= c;
67   rfn.dd(rfn.nterm)= d;
68   rfn.w1(rfn.nterm)= sys.w(d,a);
69   rfn.w2(rfn.nterm)= sys.w(d,b);
70   rfn.w3(rfn.nterm)= sys.w(d,c);
71   rfn.rho0(rfn.nterm)= sys.rho0(a);
72   [nYTmp, yFacTmp, yTypeTmp] = ...
73   buildRot(a,b,c,d,sys.muDir,...
74   rfn.rtype(rfn.nterm), expr.polar);
75   rfn.nYTerms(rfn.nterm) = nYTmp;
76   rfn.yFac(rfn.nterm,1:nYTmp) = yFacTmp;
77   rfn.yType(rfn.nterm,1:nYTmp) = yTypeTmp;
78   rfn.prefac(rfn.nterm) = prefac;
79   if l_poprlx %if set to true in sp_SetOptions.m, then adds in pop rlx
     component
80   rfn.T1_1(rfn.nterm) = 0.5/sys.popRlx(d) + 0.5/sys.popRlx(a);
81   rfn.T1_2(rfn.nterm) = 0.5/sys.popRlx(d) + 0.5/sys.popRlx(b);
82   rfn.T1_3(rfn.nterm) = 0.5/sys.popRlx(d) + 0.5/sys.popRlx(c);
83   end
```

```
84   end
85   end
86   end
87   end
88   % Check for R2 Members
89   prefac = sys.mu(c,b)*sys.mu(d,c)*sys.mu(b,a)*sys.mu(a,d)*...
90   sys.rho0(a);
91   if (prefac > prefacThresh)
92   if (abs(sys.w(b,c) - expr.w(4)) <= rwaThresh)
93   if (abs(sys.w(d,b) + (expr.w(2) - expr.w(1))) ...
94   <= rwaThresh)
95   if (abs(sys.w(d,a) - expr.w(1)) <= rwaThresh)
96   rfn.nterm = rfn.nterm + 1;
97   rfn.rtype(rfn.nterm) = 2;
98   rfn.aa(rfn.nterm) = a;
99   rfn.bb(rfn.nterm) = b;
100  rfn.cc(rfn.nterm) = c;
101  rfn.dd(rfn.nterm) = d;
102  rfn.w1(rfn.nterm) = sys.w(d,a);
103  rfn.w2(rfn.nterm) = sys.w(d,b);
104  rfn.w3(rfn.nterm) = sys.w(b,c);
105  rfn.rho0(rfn.nterm) = sys.rho0(a);
106  [nYTmp, yFacTmp, yTypeTmp] = ...
107  buildRot(a,b,c,d,sys.muDir,...
108  rfn.rtype(rfn.nterm), expr.polar);
109  rfn.nYTerms(rfn.nterm) = nYTmp;
110  rfn.yFac(rfn.nterm,1:nYTmp) = yFacTmp;
111  rfn.yType(rfn.nterm,1:nYTmp) = yTypeTmp;
112  rfn.prefac(rfn.nterm) = prefac;
```

```
113  if l_poprlx
114  rfn.T1_1(rfn.nterm) = 0.5/sys.popRlx(d) + 0.5/sys.popRlx(a);
115  rfn.T1_2(rfn.nterm) = 0.5/sys.popRlx(d) + 0.5/sys.popRlx(b);
116  rfn.T1_3(rfn.nterm) = 0.5/sys.popRlx(b) + 0.5/sys.popRlx(c);
117  end
118  end
119  end
120  end
121  end
122  % Check for R3 Members
123  prefac = sys.mu(c,b)*sys.mu(b,a)*sys.mu(d,c)*sys.mu(d,a)*...
124  sys.rho0(a);
125  if (prefac > prefacThresh)
126  if (abs(sys.w(b,c) - expr.w(4)) <= rwaThresh)
127  if (abs(sys.w(c,a) + (expr.w(2) - expr.w(1))) ...
128  <= rwaThresh)
129  if (abs(sys.w(d,a) - expr.w(1)) <= rwaThresh)
130  rfn.nterm = rfn.nterm + 1;
131  rfn.rtype(rfn.nterm)= 3;
132  rfn.aa(rfn.nterm)= a;
133  rfn.bb(rfn.nterm)= b;
134  rfn.cc(rfn.nterm)= c;
135  rfn.dd(rfn.nterm)= d;
136  rfn.w1(rfn.nterm)= sys.w(d,a);
137  rfn.w2(rfn.nterm)= sys.w(c,a);
138  rfn.w3(rfn.nterm)= sys.w(b,c);
139  rfn.rho0(rfn.nterm)= sys.rho0(a);
140  [nYTmp, yFacTmp, yTypeTmp] = ...
141  buildRot(a,b,c,d,sys.muDir,...
```

```
142  rfn.rtype(rfn.nterm), expr.polar);
143  rfn.nYTerms(rfn.nterm) = nYTmp;
144  rfn.yFac(rfn.nterm,1:nYTmp) = yFacTmp;
145  rfn.yType(rfn.nterm,1:nYTmp) = yTypeTmp;
146  rfn.prefac(rfn.nterm) = prefac;
147  if l_poprlx
148  rfn.T1_1(rfn.nterm) = 0.5/sys.popRlx(d) + 0.5/sys.popRlx(a);
149  rfn.T1_2(rfn.nterm) = 0.5/sys.popRlx(a) + 0.5/sys.popRlx(c);
150  rfn.T1_3(rfn.nterm) = 0.5/sys.popRlx(b) + 0.5/sys.popRlx(c);
151  end
152  end
153  end
154  end
155  end
156  % Check for R4 Members
157  prefac = sys.mu(a,b)*sys.mu(b,c)*sys.mu(c,d)*sys.mu(d,a)*...
158  sys.rho0(a);
159  if (prefac > prefacThresh)
160  if (abs(sys.w(b,a) - expr.w(4)) <= rwaThresh)
161  if (abs(sys.w(c,a) - (expr.w(2) - expr.w(1))) ...
162  <= rwaThresh)
163  if (abs(sys.w(d,a) + expr.w(1)) <= rwaThresh)
164  rfn.nterm = rfn.nterm + 1;
165  rfn.rtype(rfn.nterm)= 4;
166  rfn.aa(rfn.nterm)= a;
167  rfn.bb(rfn.nterm)= b;
168  rfn.cc(rfn.nterm)= c;
169  rfn.dd(rfn.nterm)= d;
170  rfn.w1(rfn.nterm)= sys.w(d,a);
```

```
171  rfn.w2(rfn.nterm)= sys.w(c,a);
172  rfn.w3(rfn.nterm)= sys.w(b,a);
173  rfn.rho0(rfn.nterm)= sys.rho0(a);
174  [nYTmp, yFacTmp, yTypeTmp] = ...
175  buildRot(a,b,c,d,sys.muDir,...
176  rfn.rtype(rfn.nterm), expr.polar);
177  rfn.nYTerms(rfn.nterm) = nYTmp;
178  rfn.yFac(rfn.nterm,1:nYTmp) = yFacTmp;
179  rfn.yType(rfn.nterm,1:nYTmp) = yTypeTmp;
180  rfn.prefac(rfn.nterm) = prefac;
181  if l_poprlx
182  rfn.T1_1(rfn.nterm) = 0.5/sys.popRlx(d) + 0.5/sys.popRlx(a);
183  rfn.T1_2(rfn.nterm) = 0.5/sys.popRlx(c) + 0.5/sys.popRlx(a);
184  rfn.T1_3(rfn.nterm) = 0.5/sys.popRlx(b) + 0.5/sys.popRlx(a);
185  end
186  end
187  end
188  end
189  end
190  % Check for R5 (R1*) Members
191  prefac = sys.mu(c,d)*sys.mu(b,c)*sys.mu(a,b)*sys.mu(a,d)*...
192  sys.rho0(a);
193  if (prefac > prefacThresh)
194  if (abs(sys.w(d,c) + expr.w(4)) <= rwaThresh)
195  if (abs(sys.w(d,b) + (expr.w(2) - expr.w(1))) ...
196  <= rwaThresh)
197  if (abs(sys.w(d,a) - expr.w(1)) <= rwaThresh)
198  rfn.nterm = rfn.nterm + 1;
199  rfn.rtype(rfn.nterm)= 5;
```

```
200  rfn.aa(rfn.nterm)= a;
201  rfn.bb(rfn.nterm)= b;
202  rfn.cc(rfn.nterm)= c;
203  rfn.dd(rfn.nterm)= d;
204  rfn.w1(rfn.nterm)= sys.w(d,a);
205  rfn.w2(rfn.nterm)= sys.w(d,b);
206  rfn.w3(rfn.nterm)= sys.w(d,c);
207  rfn.rho0(rfn.nterm)= sys.rho0(a);
208  [nYTmp, yFacTmp, yTypeTmp] = ...
209  buildRot(a,b,c,d,sys.muDir,...
210  rfn.rtype(rfn.nterm), expr.polar);
211  rfn.nYTerms(rfn.nterm) = nYTmp;
212  rfn.yFac(rfn.nterm,1:nYTmp) = yFacTmp;
213  rfn.yType(rfn.nterm,1:nYTmp) = yTypeTmp;
214  rfn.prefac(rfn.nterm) = prefac;
215  if l_poprlx
216  rfn.T1_1(rfn.nterm) = 0.5/sys.popRlx(d) + 0.5/sys.popRlx(a);
217  rfn.T1_2(rfn.nterm) = 0.5/sys.popRlx(d) + 0.5/sys.popRlx(b);
218  rfn.T1_3(rfn.nterm) = 0.5/sys.popRlx(d) + 0.5/sys.popRlx(c);
219  end
220  end
221  end
222  end
223  end
224  % Check for R6 (R2*) Members
225  prefac = sys.mu(c,b)*sys.mu(d,c)*sys.mu(b,a)*sys.mu(a,d)*...
226  sys.rho0(a);
227  if (prefac > prefacThresh)
228  if (abs(sys.w(b,c) + expr.w(4)) <= rwaThresh)
```

```
229  if (abs(sys.w(d,b) - (expr.w(2) - expr.w(1))) ...
230  <= rwaThresh)
231  if (abs(sys.w(d,a) + expr.w(1)) <= rwaThresh)
232  rfn.nterm = rfn.nterm + 1;
233  rfn.rtype(rfn.nterm)= 6;
234  rfn.aa(rfn.nterm)= a;
235  rfn.bb(rfn.nterm)= b;
236  rfn.cc(rfn.nterm)= c;
237  rfn.dd(rfn.nterm)= d;
238  rfn.w1(rfn.nterm)= sys.w(d,a);
239  rfn.w2(rfn.nterm)= sys.w(d,b);
240  rfn.w3(rfn.nterm)= sys.w(b,c);
241  rfn.rho0(rfn.nterm)= sys.rho0(a);
242  [nYTmp, yFacTmp, yTypeTmp] = ...
243  buildRot(a,b,c,d,sys.muDir,...
244  rfn.rtype(rfn.nterm), expr.polar);
245  rfn.nYTerms(rfn.nterm) = nYTmp;
246  rfn.yFac(rfn.nterm,1:nYTmp) = yFacTmp;
247  rfn.yType(rfn.nterm,1:nYTmp) = yTypeTmp;
248  rfn.prefac(rfn.nterm) = prefac;
249  if l_poprlx
250  rfn.T1_1(rfn.nterm) = 0.5/sys.popRlx(d) + 0.5/sys.popRlx(a);
251  rfn.T1_2(rfn.nterm) = 0.5/sys.popRlx(d) + 0.5/sys.popRlx(b);
252  rfn.T1_3(rfn.nterm) = 0.5/sys.popRlx(b) + 0.5/sys.popRlx(c);
253  end
254  end
255  end
256  end
257  end
```

```
% Check for R7 (R3*) Members
prefac = sys.mu(c,b)*sys.mu(b,a)*sys.mu(d,c)*sys.mu(d,a)*...
sys.rho0(a);
if (prefac > prefacThresh)
if (abs(sys.w(b,c) + expr.w(4)) <= rwaThresh)
if (abs(sys.w(c,a) - (expr.w(2) - expr.w(1))) ...
<= rwaThresh)
if (abs(sys.w(d,a) + expr.w(1)) <= rwaThresh)
rfn.nterm = rfn.nterm + 1;
rfn.rtype(rfn.nterm)= 7;
rfn.aa(rfn.nterm)= a;
rfn.bb(rfn.nterm)= b;
rfn.cc(rfn.nterm)= c;
rfn.dd(rfn.nterm)= d;
rfn.w1(rfn.nterm)= sys.w(d,a);
rfn.w2(rfn.nterm)= sys.w(c,a);
rfn.w3(rfn.nterm)= sys.w(b,c);
rfn.rho0(rfn.nterm)= sys.rho0(a);
[nYTmp, yFacTmp, yTypeTmp] = ...
buildRot(a,b,c,d,sys.muDir,...
rfn.rtype(rfn.nterm), expr.polar);
rfn.nYTerms(rfn.nterm) = nYTmp;
rfn.yFac(rfn.nterm,1:nYTmp) = yFacTmp;
rfn.yType(rfn.nterm,1:nYTmp) = yTypeTmp;
rfn.prefac(rfn.nterm) = prefac;
if l_poprlx
rfn.T1_1(rfn.nterm) = 0.5/sys.popRlx(d) + 0.5/sys.popRlx(a);
rfn.T1_2(rfn.nterm) = 0.5/sys.popRlx(c) + 0.5/sys.popRlx(a);
rfn.T1_3(rfn.nterm) = 0.5/sys.popRlx(b) + 0.5/sys.popRlx(c);
```

```
287  end
288  end
289  end
290  end
291  end
292  % Check for R8 (R4*) Members
293  prefac = sys.mu(a,b)*sys.mu(b,c)*sys.mu(c,d)*sys.mu(d,a)*...
294  sys.rho0(a);
295  if (prefac > prefacThresh)
296  if (abs(sys.w(b,a) + expr.w(4)) <= rwaThresh)
297  if (abs(sys.w(c,a) + (expr.w(2) - expr.w(1))) ...
298  <= rwaThresh)
299  if (abs(sys.w(d,a) - expr.w(1)) <= rwaThresh)
300  rfn.nterm = rfn.nterm + 1;
301  rfn.rtype(rfn.nterm)= 8;
302  rfn.aa(rfn.nterm)= a;
303  rfn.bb(rfn.nterm)= b;
304  rfn.cc(rfn.nterm)= c;
305  rfn.dd(rfn.nterm)= d;
306  rfn.w1(rfn.nterm)= sys.w(d,a);
307  rfn.w2(rfn.nterm)= sys.w(c,a);
308  rfn.w3(rfn.nterm)= sys.w(b,a);
309  rfn.rho0(rfn.nterm)= sys.rho0(a);
310  [nYTmp, yFacTmp, yTypeTmp] = ...
311  buildRot(a,b,c,d,sys.muDir,...
312  rfn.rtype(rfn.nterm), expr.polar);
313  rfn.nYTerms(rfn.nterm) = nYTmp;
314  rfn.yFac(rfn.nterm,1:nYTmp) = yFacTmp;
315  rfn.yType(rfn.nterm,1:nYTmp) = yTypeTmp;
```

```
316  rfn.prefac(rfn.nterm) = prefac;
317  if l_poprlx
318  rfn.T1_1(rfn.nterm) = 0.5/sys.popRlx(d) + 0.5/sys.popRlx(a);
319  rfn.T1_2(rfn.nterm) = 0.5/sys.popRlx(c) + 0.5/sys.popRlx(a);
320  rfn.T1_3(rfn.nterm) = 0.5/sys.popRlx(b) + 0.5/sys.popRlx(a);
321  end
322  end
323  end
324  end
325  end
326  end
327  end
328  end
329  end
330  % Deallocate Unused Memory
331  if (rfn.nterm < sizeGuess)
332  rfn.rtype(rfn.nterm+1:sizeGuess) = [];
333  rfn.aa(rfn.nterm+1:sizeGuess) = [];
334  rfn.bb(rfn.nterm+1:sizeGuess) = [];
335  rfn.cc(rfn.nterm+1:sizeGuess) = [];
336  rfn.dd(rfn.nterm+1:sizeGuess) = [];
337  rfn.w1(rfn.nterm+1:sizeGuess) = [];
338  rfn.w2(rfn.nterm+1:sizeGuess) = [];
339  rfn.w3(rfn.nterm+1:sizeGuess) = [];
340  rfn.rho0(rfn.nterm+1:sizeGuess) = [];
341  rfn.nYTerms(rfn.nterm+1:sizeGuess) = [];
342  end
343
```

```
344  % Convert wave numbers to inverse femtoseconds (for each type of tau
     delay)
345  cm2fs = 2*pi*2.998e-5;
346  rfn.w1 = rfn.w1*cm2fs;
347  rfn.w2 = rfn.w2*cm2fs;
348  rfn.w3 = rfn.w3*cm2fs;
349
350  % Add Lineshape functions to response function
351  rfn.hfun = sys.hfun;
352
353  % Add Rotational Diffusion coefficient to Response function
354  rfn.dor = sys.dor;
355
356  % End timing
357  time_elapsed = toc(ticID);
358  outStr = strcat('Time elapsed for buildEcho:  ',num2str(time_elapsed));
359  disp(outStr);
360
361  end % buildEcho
```

A.5.2 *buildRot.m*

This function file builds the orientational response portion of the response function.

```
1    function [nYTerms, yFac, yType] = buildRot(a, b, c, d,...
2    dipoleDir, rType, polarization)
3    % BUILDROT - build oritentational contribution to the response funciton
4    % Each of the pieces in this code can be viewed in a more straight-
     forward
5    % manner in Table II of JCP, 2001, 115, 297. The code view each
     possible
```

```
6    % contribution as orthogonal transition dipole moments, which is why
     the
7    % contribution each dipole made to the 3 cardinal directions had to be
8    % calculated in the matrix muDir when inputing the system parameters.
9
10   % termThresh is currently hard coded to be 1.0%, but this can be
     changed if
11   % needed.
12   termThresh = 0.01;
13
14   %For each case (which are determined in buildEcho.m, it calculates the
15   %product of the directions for each transition in all 3 dimensions
16   %(molecules can be oriented in any direction). This is 'yCosTerm'.
17
18   %Determines the type of orientation of orientational contribution based
     on
19   %allowed directions/dimensions for polarizations of light
     interactions).
20   %This is in 'selectYtype' towards bottom of file.
21
22   %Terms are added for each case that meets all the requirements.
23   nYTerms = 0;
24   switch rType
25   case 1
26   for ii = 1:3
27   for jj = 1:3
28   for kk = 1:3
29   for ll = 1:3
30   yCosTerm = dipoleDir(c,d,ll)*dipoleDir(b,c,kk)*...
```

```
31   dipoleDir(a,b,jj)*dipoleDir(d,a,ii);
32   type = selectYType(ii,jj,kk,ll,polarization);
33   if ((yCosTerm > termThresh) && (type ~= 0))
34   nYTerms = nYTerms + 1;
35   yFac(nYTerms) = calcYFac(yCosTerm, type);
36   yType(nYTerms) = type;
37   end
38   end
39   end
40   end
41   end
42   case 2
43   for ii = 1:3
44   for jj = 1:3
45   for kk = 1:3
46   for ll = 1:3
47   yCosTerm = dipoleDir(c,b,ii)*dipoleDir(d,c,jj)*...
48   dipoleDir(b,a,kk)*dipoleDir(a,d,ll);
49   type = selectYType(ii,jj,kk,ll,polarization);
50   if ((yCosTerm > termThresh) && (type ~= 0))
51   nYTerms = nYTerms + 1;
52   yFac(nYTerms) = calcYFac(yCosTerm, type);
53   yType(nYTerms) = type;
54   end
55   end
56   end
57   end
58   end
59   case 3
```

```
60   for ii = 1:3
61   for jj = 1:3
62   for kk = 1:3
63   for ll = 1:3
64   yCosTerm = dipoleDir(c,b,ii)*dipoleDir(b,a,jj)*...
65   dipoleDir(d,c,kk)*dipoleDir(a,d,ll);
66   type = selectYType(ii,jj,kk,ll,polarization);
67   if ((yCosTerm > termThresh) && (type ~= 0))
68   nYTerms = nYTerms + 1;
69   yFac(nYTerms) = calcYFac(yCosTerm, type);
70   yType(nYTerms) = type;
71   end
72   end
73   end
74   end
75   end
76   case 4
77   for ii = 1:3
78   for jj = 1:3
79   for kk = 1:3
80   for ll = 1:3
81   yCosTerm = dipoleDir(a,b,ii)*dipoleDir(b,c,jj)*...
82   dipoleDir(c,d,kk)*dipoleDir(a,d,ll);
83   type = selectYType(ii,jj,kk,ll,polarization);
84   if ((yCosTerm > termThresh) && (type ~= 0))
85   nYTerms = nYTerms + 1;
86   yFac(nYTerms) = calcYFac(yCosTerm, type);
87   yType(nYTerms) = type;
88   end
```

```
89   end
90   end
91   end
92   end
93   case 5
94   for ii = 1:3
95   for jj = 1:3
96   for kk = 1:3
97   for ll = 1:3
98   yCosTerm = dipoleDir(c,d,ll)*dipoleDir(b,c,kk)*...
99   dipoleDir(a,b,jj)*dipoleDir(d,a,ii);
100  type = selectYType(ii,jj,kk,ll,polarization);
101  if ((yCosTerm > termThresh) && (type ~= 0))
102  nYTerms = nYTerms + 1;
103  yFac(nYTerms) = calcYFac(yCosTerm, type);
104  yType(nYTerms) = type;
105  end
106  end
107  end
108  end
109  end
110  case 6
111  for ii = 1:3
112  for jj = 1:3
113  for kk = 1:3
114  for ll = 1:3
115  yCosTerm = dipoleDir(c,b,ii)*dipoleDir(d,c,jj)*...
116  dipoleDir(b,a,kk)*dipoleDir(a,d,ll);
117  type = selectYType(ii,jj,kk,ll,polarization);
```

```
118  if ((yCosTerm > termThresh) && (type ~= 0))
119  nYTerms = nYTerms + 1;
120  yFac(nYTerms) = calcYFac(yCosTerm, type);
121  yType(nYTerms) = type;
122  end
123  end
124  end
125  end
126  end
127  case 7
128  for ii = 1:3
129  for jj = 1:3
130  for kk = 1:3
131  for ll = 1:3
132  yCosTerm = dipoleDir(c,b,ii)*dipoleDir(b,a,jj)*...
133  dipoleDir(d,c,kk)*dipoleDir(a,d,ll);
134  type = selectYType(ii,jj,kk,ll,polarization);
135  if ((yCosTerm > termThresh) && (type ~= 0))
136  nYTerms = nYTerms + 1;
137  yFac(nYTerms) = calcYFac(yCosTerm, type);
138  yType(nYTerms) = type;
139  end
140  end
141  end
142  end
143  end
144  case 8
145  for ii = 1:3
146  for jj = 1:3
```

```
147  for kk = 1:3
148  for ll = 1:3
149  yCosTerm = dipoleDir(a,b,ii)*dipoleDir(b,c,jj)*...
150  dipoleDir(c,d,kk)*dipoleDir(a,d,ll);
151  type = selectYType(ii,jj,kk,ll,polarization);
152  if ((yCosTerm > termThresh) && (type ~= 0))
153  nYTerms = nYTerms + 1;
154  yFac(nYTerms) = calcYFac(yCosTerm, type);
155  yType(nYTerms) = type;
156  end
157  end
158  end
159  end
160  end
161  otherwise
162  error('specPack:buildRot','No Response function matches this type');
163  end
164
165  end
166
167  function type = selectYType(ii,jj,kk,ll,polarization)
168  %SELECTYTYPE chooses type of y term
169  %
170
171  if strcmpi(polarization,'ZZZZ')
172  if (ii == jj) && (ii == kk) && (ii == ll)
173  type = 1;
174  elseif (ii == jj) && (ii ~= kk) && (kk == ll)
175  type = 2;
```

```
176  elseif (ii ~= jj) && (jj == kk) && (ii == ll)
177  type = 3;
178  elseif (ii ~= jj) && (jj == ll) && (ii == kk)
179  type = 3;
180  else
181  type = 0;
182  end
183  elseif strcmpi(polarization,'YYZZ')
184  if (ii == jj) && (ii == kk) && (ii == ll)
185  type = 2;
186  elseif (ii == jj) && (ii ~= kk) && (kk == ll)
187  type = 4;
188  elseif (ii ~= jj) && (jj == kk) && (ii == ll)
189  type = 5;
190  elseif (ii ~= jj) && (jj == ll) && (ii == kk)
191  type = 5;
192  else
193  type = 0;
194  end
195  else
196  error('specPack:selectYType','Polarization must be ZZZZ or YYZZ')
197  end
198
199  end
200
201  function y = calcYFac(cosTerm, yType)
202  %CALCYFAC Calculate prefactor for rotational contribution to spectrum
203  %If 'yCosTerm' is larger than threshold and 'yType' is not zero, this
```

```
204   %determines the orientational factor, which is just 'yCosTerm' divided
      by
205   %the factor determined by the 'yType'.
206
207   switch yType
208   case 1
209   y = cosTerm/9.0;
210   case 2
211   y = cosTerm/9.0;
212   case 3
213   y = cosTerm/15.0;
214   case 4
215   y = cosTerm/9.0;
216   case 5
217   y = -cosTerm/30.0;
218   otherwise
219   error('specPack:calcYFac','Unknown yType');
220   end
221
222   end
```

A.5.3 *buildVirtEcho.m*

This function file builds the virtual echo (non-rephasing) response functions.

```
1     function rfn = buildVirtEcho(sys, expr, rwaThresh, prefacThresh,
      l_poprlx)
2     %BUILDVIRTECHO construct virtual photon echo response function
3     %  Description: This function takes system and experimental parameters,
      and
```

```
4    %  it constructs a response function for the ks = k1-k2+k3 phase
     matching
5    %  condition.  Additional comments can be found in buildEcho.m
6    %
7    %  Parameters: sys - Structure containing system parameters
8    %              exp - Structure containing experimental parameters
9    %              rwaThresh - Threshold for resonance for RWA ( units =
     cm^-1)
10   %              dipoleThresh - product of rho0 and dipoles is assumed
11   %                             to be 0 if smaller than prefacThresh
12   %              l_poprlx - logical controlling population relaxation
13   %
14   %  Returns: rfn - structure with response funciton parameters
15   %           rfn.nterms - number of non-zero terms in response function
16   %           rfn.rtype - array with rfn types
17   %           rfn.aa - array with 1st eigenstate for each function
18   %           rfn.bb - array with 2nd eigenstate for each function
19   %           rfn.cc - array with 3rd eigenstate for each function
20   %           rfn.dd - array with 4th eigenstate for each function
21   %           rfn.w1 - array with t1 frequency for each function
22   %           rfn.w2 - array with t2 frequency for each function
23   %           rfn.w3 - array with t3 frequency for each function
24   %           rfn.rho0 - population in state a at t = 0
25   %           rfn.T1_1 - array with population relaxation contrib to t1
26   %           rfn.T1_2 - array with population relaxation contrib to t2
27   %           rfn.T1_3 - array with population relaxation contrib to t3
28
29   % Start Timing
30   timeID = tic;
```

```
31
32   rfn.nterm = 0;
33   sizeGuess = sys.nStates*6;
34
35   % Preallocated rfs parameter arrays
36   rfn.rtype = zeros(1, sizeGuess);
37   rfn.aa = zeros(1,sizeGuess);
38   rfn.bb = zeros(1,sizeGuess);
39   rfn.cc = zeros(1,sizeGuess);
40   rfn.dd = zeros(1,sizeGuess);
41   rfn.w1 = zeros(1,sizeGuess);
42   rfn.w2 = zeros(1,sizeGuess);
43   rfn.w3 = zeros(1,sizeGuess);
44   rfn.rho0 = zeros(1,sizeGuess);
45   rfn.nYTerms = zeros(1,sizeGuess);
46   for a=1:sys.nStates
47   for b=1:sys.nStates
48   for c=1:sys.nStates
49   for d=1:sys.nStates
50   % Check for R1 Members
51   prefac = sys.mu(c,d)*sys.mu(b,c)*sys.mu(a,b)*sys.mu(a,d)*...
52   sys.rho0(a);
53   if (prefac > prefacThresh)
54   if (abs(sys.w(d,c) - expr.w(4)) <= rwaThresh)
55   if (abs(sys.w(d,b) - (expr.w(2) - expr.w(1))) ...
56   <= rwaThresh)
57   if (abs(sys.w(d,a) - expr.w(2)) <= rwaThresh)
58   rfn.nterm = rfn.nterm + 1;
59   rfn.rtype(rfn.nterm)= 1;
```

```
60   rfn.aa(rfn.nterm)= a;
61   rfn.bb(rfn.nterm)= b;
62   rfn.cc(rfn.nterm)= c;
63   rfn.dd(rfn.nterm)= d;
64   rfn.w1(rfn.nterm)= sys.w(d,a);
65   rfn.w2(rfn.nterm)= sys.w(d,b);
66   rfn.w3(rfn.nterm)= sys.w(d,c);
67   rfn.rho0(rfn.nterm)= sys.rho0(a);
68   [nYTmp, yFacTmp, yTypeTmp] = ...
69   buildRot(a,b,c,d,sys.muDir,...
70   rfn.rtype(rfn.nterm), expr.polar);
71   rfn.nYTerms(rfn.nterm) = nYTmp;
72   rfn.yFac(rfn.nterm,1:nYTmp) = yFacTmp;
73   rfn.yType(rfn.nterm,1:nYTmp) = yTypeTmp;
74   rfn.prefac(rfn.nterm) = prefac;
75   if l_poprlx
76   rfn.T1_1(rfn.nterm) = 0.5/sys.popRlx(d) + 0.5/sys.popRlx(a);
77   rfn.T1_2(rfn.nterm) = 0.5/sys.popRlx(d) + 0.5/sys.popRlx(b);
78   rfn.T1_3(rfn.nterm) = 0.5/sys.popRlx(d) + 0.5/sys.popRlx(c);
79   end
80   end
81   end
82   end
83   end
84   % Check for R2 Members
85   prefac = sys.mu(c,b)*sys.mu(d,c)*sys.mu(b,a)*sys.mu(a,d)*...
86   sys.rho0(a);
87   if (prefac > prefacThresh)
88   if (abs(sys.w(b,c) - expr.w(4)) <= rwaThresh)
```

```
89   if (abs(sys.w(d,b) + (expr.w(2) - expr.w(1))) ...
90   <= rwaThresh)
91   if (abs(sys.w(d,a) + expr.w(2)) <= rwaThresh)
92   rfn.nterm = rfn.nterm + 1;
93   rfn.rtype(rfn.nterm)= 2;
94   rfn.aa(rfn.nterm)= a;
95   rfn.bb(rfn.nterm)= b;
96   rfn.cc(rfn.nterm)= c;
97   rfn.dd(rfn.nterm)= d;
98   rfn.w1(rfn.nterm)= sys.w(d,a);
99   rfn.w2(rfn.nterm)= sys.w(d,b);
100  rfn.w3(rfn.nterm)= sys.w(b,c);
101  rfn.rho0(rfn.nterm)= sys.rho0(a);
102  [nYTmp, yFacTmp, yTypeTmp] = ...
103  buildRot(a,b,c,d,sys.muDir,...
104  rfn.rtype(rfn.nterm), expr.polar);
105  rfn.nYTerms(rfn.nterm) = nYTmp;
106  rfn.yFac(rfn.nterm,1:nYTmp) = yFacTmp;
107  rfn.yType(rfn.nterm,1:nYTmp) = yTypeTmp;
108  rfn.prefac(rfn.nterm) = prefac;
109  if l_poprlx
110  rfn.T1_1(rfn.nterm) = 0.5/sys.popRlx(d) + 0.5/sys.popRlx(a);
111  rfn.T1_2(rfn.nterm) = 0.5/sys.popRlx(d) + 0.5/sys.popRlx(b);
112  rfn.T1_3(rfn.nterm) = 0.5/sys.popRlx(b) + 0.5/sys.popRlx(c);
113  end
114  end
115  end
116  end
117  end
```

```
% Check for R3 Members
prefac = sys.mu(c,b)*sys.mu(b,a)*sys.mu(d,c)*sys.mu(d,a)*...
sys.rho0(a);
if (prefac > prefacThresh)
if (abs(sys.w(b,c) - expr.w(4)) <= rwaThresh)
if (abs(sys.w(c,a) + (expr.w(2) - expr.w(1))) ...
<= rwaThresh)
if (abs(sys.w(d,a) + expr.w(2)) <= rwaThresh)
rfn.nterm = rfn.nterm + 1;
rfn.rtype(rfn.nterm)= 3;
rfn.aa(rfn.nterm)= a;
rfn.bb(rfn.nterm)= b;
rfn.cc(rfn.nterm)= c;
rfn.dd(rfn.nterm)= d;
rfn.w1(rfn.nterm)= sys.w(d,a);
rfn.w2(rfn.nterm)= sys.w(c,a);
rfn.w3(rfn.nterm)= sys.w(b,c);
rfn.rho0(rfn.nterm)= sys.rho0(a);
[nYTmp, yFacTmp, yTypeTmp] = ...
buildRot(a,b,c,d,sys.muDir,...
rfn.rtype(rfn.nterm), expr.polar);
rfn.nYTerms(rfn.nterm) = nYTmp;
rfn.yFac(rfn.nterm,1:nYTmp) = yFacTmp;
rfn.yType(rfn.nterm,1:nYTmp) = yTypeTmp;
rfn.prefac(rfn.nterm) = prefac;
if l_poprlx
rfn.T1_1(rfn.nterm) = 0.5/sys.popRlx(d) + 0.5/sys.popRlx(a);
rfn.T1_2(rfn.nterm) = 0.5/sys.popRlx(a) + 0.5/sys.popRlx(c);
rfn.T1_3(rfn.nterm) = 0.5/sys.popRlx(b) + 0.5/sys.popRlx(c);
```

```
147  end
148  end
149  end
150  end
151  end
152  % Check for R4 Members
153  prefac = sys.mu(a,b)*sys.mu(b,c)*sys.mu(c,d)*sys.mu(d,a)*...
154  sys.rho0(a);
155  if (prefac > prefacThresh)
156  if (abs(sys.w(b,a) - expr.w(4)) <= rwaThresh)
157  if (abs(sys.w(c,a) - (expr.w(2) - expr.w(1))) ...
158  <= rwaThresh)
159  if (abs(sys.w(d,a) - expr.w(2)) <= rwaThresh)
160  rfn.nterm = rfn.nterm + 1;
161  rfn.rtype(rfn.nterm)= 4;
162  rfn.aa(rfn.nterm)= a;
163  rfn.bb(rfn.nterm)= b;
164  rfn.cc(rfn.nterm)= c;
165  rfn.dd(rfn.nterm)= d;
166  rfn.w1(rfn.nterm)= sys.w(d,a);
167  rfn.w2(rfn.nterm)= sys.w(c,a);
168  rfn.w3(rfn.nterm)= sys.w(b,a);
169  rfn.rho0(rfn.nterm)= sys.rho0(a);
170  [nYTmp, yFacTmp, yTypeTmp] = ...
171  buildRot(a,b,c,d,sys.muDir,...
172  rfn.rtype(rfn.nterm), expr.polar);
173  rfn.nYTerms(rfn.nterm) = nYTmp;
174  rfn.yFac(rfn.nterm,1:nYTmp) = yFacTmp;
175  rfn.yType(rfn.nterm,1:nYTmp) = yTypeTmp;
```

```
176  rfn.prefac(rfn.nterm) = prefac;
177  if l_poprlx
178  rfn.T1_1(rfn.nterm) = 0.5/sys.popRlx(d) + 0.5/sys.popRlx(a);
179  rfn.T1_2(rfn.nterm) = 0.5/sys.popRlx(c) + 0.5/sys.popRlx(a);
180  rfn.T1_3(rfn.nterm) = 0.5/sys.popRlx(b) + 0.5/sys.popRlx(a);
181  end
182  end
183  end
184  end
185  end
186  % Check for R5 (R1*) Members
187  prefac = sys.mu(c,d)*sys.mu(b,c)*sys.mu(a,b)*sys.mu(a,d)*...
188  sys.rho0(a);
189  if (prefac > prefacThresh)
190  if (abs(sys.w(d,c) + expr.w(4)) <= rwaThresh)
191  if (abs(sys.w(d,b) + (expr.w(2) - expr.w(1))) ...
192  <= rwaThresh)
193  if (abs(sys.w(d,a) + expr.w(2)) <= rwaThresh)
194  rfn.nterm = rfn.nterm + 1;
195  rfn.rtype(rfn.nterm)= 5;
196  rfn.aa(rfn.nterm)= a;
197  rfn.bb(rfn.nterm)= b;
198  rfn.cc(rfn.nterm)= c;
199  rfn.dd(rfn.nterm)= d;
200  rfn.w1(rfn.nterm)= sys.w(d,a);
201  rfn.w2(rfn.nterm)= sys.w(d,b);
202  rfn.w3(rfn.nterm)= sys.w(d,c);
203  rfn.rho0(rfn.nterm)= sys.rho0(a);
204  [nYTmp, yFacTmp, yTypeTmp] = ...
```

```
buildRot(a,b,c,d,sys.muDir,...
rfn.rtype(rfn.nterm), expr.polar);
rfn.nYTerms(rfn.nterm) = nYTmp;
rfn.yFac(rfn.nterm,1:nYTmp) = yFacTmp;
rfn.yType(rfn.nterm,1:nYTmp) = yTypeTmp;
rfn.prefac(rfn.nterm) = prefac;
if l_poprlx
rfn.T1_1(rfn.nterm) = 0.5/sys.popRlx(d) + 0.5/sys.popRlx(a);
rfn.T1_2(rfn.nterm) = 0.5/sys.popRlx(d) + 0.5/sys.popRlx(b);
rfn.T1_3(rfn.nterm) = 0.5/sys.popRlx(d) + 0.5/sys.popRlx(c);
end
end
end
end
end
% Check for R6 (R2*) Members
prefac = sys.mu(c,b)*sys.mu(d,c)*sys.mu(b,a)*sys.mu(a,d)*...
sys.rho0(a);
if (prefac > prefacThresh)
if (abs(sys.w(b,c) + expr.w(4)) <= rwaThresh)
if (abs(sys.w(d,b) - (expr.w(2) - expr.w(1))) ...
<= rwaThresh)
if (abs(sys.w(d,a) - expr.w(2)) <= rwaThresh)
rfn.nterm = rfn.nterm + 1;
rfn.rtype(rfn.nterm)= 6;
rfn.aa(rfn.nterm)= a;
rfn.bb(rfn.nterm)= b;
rfn.cc(rfn.nterm)= c;
rfn.dd(rfn.nterm)= d;
```

```
234  rfn.w1(rfn.nterm)= sys.w(d,a);
235  rfn.w2(rfn.nterm)= sys.w(d,b);
236  rfn.w3(rfn.nterm)= sys.w(b,c);
237  rfn.rho0(rfn.nterm)= sys.rho0(a);
238  [nYTmp, yFacTmp, yTypeTmp] = ...
239  buildRot(a,b,c,d,sys.muDir,...
240  rfn.rtype(rfn.nterm), expr.polar);
241  rfn.nYTerms(rfn.nterm) = nYTmp;
242  rfn.yFac(rfn.nterm,1:nYTmp) = yFacTmp;
243  rfn.yType(rfn.nterm,1:nYTmp) = yTypeTmp;
244  rfn.prefac(rfn.nterm) = prefac;
245  if l_poprlx
246  rfn.T1_1(rfn.nterm) = 0.5/sys.popRlx(d) + 0.5/sys.popRlx(a);
247  rfn.T1_2(rfn.nterm) = 0.5/sys.popRlx(d) + 0.5/sys.popRlx(b);
248  rfn.T1_3(rfn.nterm) = 0.5/sys.popRlx(b) + 0.5/sys.popRlx(c);
249  end
250  end
251  end
252  end
253  end
254  % Check for R7 (R3*) Members
255  prefac = sys.mu(c,b)*sys.mu(b,a)*sys.mu(d,c)*sys.mu(d,a)*...
256  sys.rho0(a);
257  if (prefac > prefacThresh)
258  if (abs(sys.w(b,c) + expr.w(4)) <= rwaThresh)
259  if (abs(sys.w(c,a) - (expr.w(2) - expr.w(1))) ...
260  <= rwaThresh)
261  if (abs(sys.w(d,a) - expr.w(2)) <= rwaThresh)
262  rfn.nterm = rfn.nterm + 1;
```

```
263  rfn.rtype(rfn.nterm)= 7;
264  rfn.aa(rfn.nterm)= a;
265  rfn.bb(rfn.nterm)= b;
266  rfn.cc(rfn.nterm)= c;
267  rfn.dd(rfn.nterm)= d;
268  rfn.w1(rfn.nterm)= sys.w(d,a);
269  rfn.w2(rfn.nterm)= sys.w(c,a);
270  rfn.w3(rfn.nterm)= sys.w(b,c);
271  rfn.rho0(rfn.nterm)= sys.rho0(a);
272  [nYTmp, yFacTmp, yTypeTmp] = ...
273  buildRot(a,b,c,d,sys.muDir,...
274  rfn.rtype(rfn.nterm), expr.polar);
275  rfn.nYTerms(rfn.nterm) = nYTmp;
276  rfn.yFac(rfn.nterm,1:nYTmp) = yFacTmp;
277  rfn.yType(rfn.nterm,1:nYTmp) = yTypeTmp;
278  rfn.prefac(rfn.nterm) = prefac;
279  if l_poprlx
280  rfn.T1_1(rfn.nterm) = 0.5/sys.popRlx(d) + 0.5/sys.popRlx(a);
281  rfn.T1_2(rfn.nterm) = 0.5/sys.popRlx(a) + 0.5/sys.popRlx(c);
282  rfn.T1_3(rfn.nterm) = 0.5/sys.popRlx(b) + 0.5/sys.popRlx(c);
283  end
284  end
285  end
286  end
287  end
288  % Check for R8 (R4*) Members
289  prefac = sys.mu(a,b)*sys.mu(b,c)*sys.mu(c,d)*sys.mu(d,a)*...
290  sys.rho0(a);
291  if (prefac > prefacThresh)
```

```
292  if (abs(sys.w(b,a) + expr.w(4)) <= rwaThresh)
293  if (abs(sys.w(c,a) + (expr.w(2) - expr.w(1))) ...
294  <= rwaThresh)
295  if (abs(sys.w(d,a) + expr.w(2)) <= rwaThresh)
296  rfn.nterm = rfn.nterm + 1;
297  rfn.rtype(rfn.nterm)= 8;
298  rfn.aa(rfn.nterm)= a;
299  rfn.bb(rfn.nterm)= b;
300  rfn.cc(rfn.nterm)= c;
301  rfn.dd(rfn.nterm)= d;
302  rfn.w1(rfn.nterm)= sys.w(d,a);
303  rfn.w2(rfn.nterm)= sys.w(c,a);
304  rfn.w3(rfn.nterm)= sys.w(b,a);
305  rfn.rho0(rfn.nterm)= sys.rho0(a);
306  [nYTmp, yFacTmp, yTypeTmp] = ...
307  buildRot(a,b,c,d,sys.muDir,...
308  rfn.rtype(rfn.nterm), expr.polar);
309  rfn.nYTerms(rfn.nterm) = nYTmp;
310  rfn.yFac(rfn.nterm,1:nYTmp) = yFacTmp;
311  rfn.yType(rfn.nterm,1:nYTmp) = yTypeTmp;
312  rfn.prefac(rfn.nterm) = prefac;
313  if l_poprlx
314  rfn.T1_1(rfn.nterm) = 0.5/sys.popRlx(d) + 0.5/sys.popRlx(a);
315  rfn.T1_2(rfn.nterm) = 0.5/sys.popRlx(c) + 0.5/sys.popRlx(a);
316  rfn.T1_3(rfn.nterm) = 0.5/sys.popRlx(b) + 0.5/sys.popRlx(a);
317  end
318  end
319  end
320  end
```

```
321  end
322  end
323  end
324  end
325  end
326  % Deallocate Unused Memory
327  if (rfn.nterm < sizeGuess)
328  rfn.rtype(rfn.nterm+1:sizeGuess) = [];
329  rfn.aa(rfn.nterm+1:sizeGuess) = [];
330  rfn.bb(rfn.nterm+1:sizeGuess) = [];
331  rfn.cc(rfn.nterm+1:sizeGuess) = [];
332  rfn.dd(rfn.nterm+1:sizeGuess) = [];
333  rfn.w1(rfn.nterm+1:sizeGuess) = [];
334  rfn.w2(rfn.nterm+1:sizeGuess) = [];
335  rfn.w3(rfn.nterm+1:sizeGuess) = [];
336  rfn.rho0(rfn.nterm+1:sizeGuess) = [];
337  rfn.nYTerms(rfn.nterm+1:sizeGuess) = [];
338  end
339
340  % Convert wave numbers to inverse femtoseconds
341  cm2w = 2*pi*2.998e-5;
342  rfn.w1 = rfn.w1*cm2w;
343  rfn.w2 = rfn.w2*cm2w;
344  rfn.w3 = rfn.w3*cm2w;
345
346  % Set lineshape fuctions
347  rfn.hfun = sys.hfun;
348
349  % Set Orientational Diffusion Coefficient
```

```
350  rfn.dor = sys.dor;
351
352  % End Timing
353  time_elapsed = toc(timeID);
354  % Print Timing
355  outstr = strcat('Time elapsed for buildVirtEcho:
     ',num2str(time_elapsed));
356  disp(outstr)
357
358  end
```

A.5.4 *calc2DImpulse.m*

```
1    function calc2DImpulse(basename, sys, expr, tau2, opts)
2    %CALC2DIMPULSE -
3    % Description: This function provides a wrapper for calculating 2DIR
4    % spectra by automating the calculation of rephasing and nonrephasing
5    % contributions. All laser pulses are assumed to be delta functions for
6    % this calculation.
7    %
8    % Parameters: basename - This is the string that defines the basename
     for
9    %                        all output files
10   %             sys      - struct containing molecular system parameters
11   %             expr     - struct containing exprimental parameters
12   %             tau2     - vector of tau2 values for the spectra
13   %             opts     - struct containing calculation options and
14   %                        thresholds
15   %
```

```
% Outputs: '[BASENAME]_t2-[VAL]_rt.mat' - rephasing response in time domain
%          '[BASENAME]_t2-[VAL]_rf.mat' - rephasing spectrum time domain
%          '[BASENAME]_t2-[VAL]_nrt.mat' - nonrephasing response in time domain
%          '[BASENAME]_t2-[VAL]_nrf.mat' - nonrephasing spectrum in frequency domain
%          '[BASENAME]_t2-[VAL]_abs.mat' - 2DIR correlation spectrum

% evaluate rephasing signals
rfn = buildEcho(sys, expr, opts.rwaThresh, opts.prefacThresh,...
opts.l_poprlx);
printRandY(rfn,strcat(basename,'_Echo_fn'));
for ii=1:length(tau2)
tau2str = num2str(tau2(ii));
%adds tau2 time and rt (rephasing signal in the time domain) to file name
tname = strcat(basename,'_t2-',tau2str,'_rt');
fname = strcat(basename,'_t2-',tau2str,'_rf'); %rephasing freq
eval2DImpulse(rfn, tname, fname, tau2(ii), opts);
if opts.l_cspecOnly %this is an option in sp_SetOptions to save file space
delete(tname)
end
end
clear rfn

% evaluate nonrephasing signals
```

```
39  rfn = buildVirtEcho(sys, expr, opts.rwaThresh, opts.prefacThresh,...
40  opts.l_poprlx);
41  printRandY(rfn,strcat(basename,'_VirtEcho_fn'));
42  for ii=1:length(tau2)
43  tau2str = num2str(tau2(ii));
44  tname = strcat(basename,'_t2-',tau2str,'_nrt'); %non-rephasing time
45  fname = strcat(basename,'_t2-',tau2str,'_nrf'); %non-rephasing freq
46  eval2DImpulse(rfn, tname, fname, tau2(ii), opts);
47  if opts.l_cspecOnly
48  delete(tname)
49  end
50  end
51  clear rfn
52
53  % Make correlation spectra
54  timeID = tic;
55  for ii = 1:length(tau2)
56  tau2str = num2str(tau2(ii));
57  fname1 = strcat(basename,'_t2-',tau2str,'_rf');
58  fname2 = strcat(basename,'_t2-',tau2str,'_nrf');
59  absname = strcat(basename,'_t2-',tau2str,'_abs');
60  fr = load(fname1);
61  a = size(fr.response_freq,1);
62  fr2 = zeros(a+1);
63  fr2(1:a,1:a) = fr.response_freq;
64  clear fr
65  fnr = load(fname2);
66  fnr2 = zeros(a+1);
67  fnr2(1:a,1:a) = fnr.response_freq;
```

```
68    abs1 = fliplr(fr2) + fnr2;
69    freq = fnr.freq;
70    freq2 = zeros(a+1,1);
71    freq2(1:a) = freq;
72    freq2(a+1) = -freq2(1);
73    freq = freq2;
74    save(absname, 'abs1','freq');
75    if opts.l_cspecOnly
76    delete(fname1)
77    delete(fname2)
78    end
79    end
80    time_elapsed = toc(timeID);
81    outstr = strcat('Time elapsed for correlation spectrum construction:
      ',...
82    num2str(time_elapsed));
83    disp(outstr)
84
85    end
```

A.5.5 *createHomLimlinEshape.m*

This function can be used to automate the creation of homogeneous lineshape functions (I think). It is not called by any of the other simulation functions as far as I can tell.

```
1     function hfun = createHomLimLineShape(dephasingTimes)
2     %CREATEHOMLIMLINESHAPE - Generate Lineshape function for Hom. Limit
3     %  Description: This function creates a cell containing the funciton
4     %  handles for linshape functions in the homogeneous limit given a
      vector
5     %  of dephasing times. It also create each individual function
```

```
6    %
7    %  Parameters: dephasingTimes - matrix of dephasing times for each
8    %  coherence the ij element of the matrix should be the dephasing time
     for
9    %  the coherence between states i and j. Function assumes ij and ji
10   %  dephasing times are the same
11   %
12   %  Output: hfun   - cell containing lineshape function handles
13   %          gij.m  - lineshape function files
14
15   % Create g0
16   fid = fopen('g0.m','w');
17   fprintf(fid,'function y = g0(t)\n');
18   fprintf(fid,'%%G0 - Linespape function to return 0');
19   fprintf(fid,'\ny = 0;');
20   fclose(fid);
21
22   shape = size(dephasingTimes);
23
24   if (shape(1) ~= shape(2))
25   error('specPack: createHomLimLineShape','Argument must be square
     matrix');
26   end
27
28   0 thresh = 1.0
29   for i=1:shape(1)
30   for j=i:shape(1)
31   if (dephasingTimes(i,j) > 0)
32   fname = strcat('g',num2str(j),num2str(j)
```

```
33    makeFunction(strcat('g',num2str(j),numstr(j)),dephasingTime(i,j));
```

A.5.6 *eval2DImpulse.m*

This evaluates the response function for the 2D IR spectrum, and saves files for the time and frequency domain responses.

```
1     function eval2DImpulse( rfn, tname, fname, tau2, opts )
2     %EVAL2DIMPULSE Evaluates response function for 2DIR with Delta Fn
      pulses
3     %   Description: This function evaluates a response function on a 2D
      time
4     %   matrix and performs the fourier transform. time and frequency
      domain
5     %   results are saved in .mat files
6     %
7     %   Parameters: rfn   - response function
8     %               tname - file name for time domain data
9     %               fname - file name for frequency domain data
10    %               tau2  - tau2 value
11    %               opts  - struct of calculation options
12    %
13    %   Output:     [tname].mat - time domain data
14    %               [fname].mat - frequency domain data
15    %
16
17    time_width = opts.nsteps+1;
18    time_end = opts.nsteps*opts.tstep;
19
20    t1 = 0:opts.tstep:time_end;
21    t1mat = repmat(t1, [time_width 1]);
```

```
22
23   t3 = (0:opts.tstep:time_end)';
24   t3mat = repmat(t3, [1 time_width]);
25
26   %If poprlx is 'true' in sp_SetOptions.m, call evalrfn_rlx, otherwise
     calls
27   %evalrfn.
28   if opts.l_poprlx
29   response_time = evalrfn_rlx(rfn, t1mat, tau2, t3mat);
30   else
31   response_time = evalrfn(rfn, t1mat, tau2, t3mat);
32   end
33
34   save(tname, 'response_time', 't1', 't3');
35
36   fsize = opts.fftsize;
37   fsizep1 = opts.fftsize+1;
38
39   response_freq = fft2(real(response_time), fsize, fsize);
40   response_freq = fftshift(response_freq);
41   clear response_time
42
43   % Make Frequency axes
44   N = opts.fftsize;
45   freq = (-N/2):(N/2-1);
46   freq = freq/(N*opts.tstep*2.998e-5);
47
48   save(fname, 'response_freq','freq');
49
```

```
50    end
```

A.5.7 *evalrfn.m*

```
1     function response = evalrfn( rfn, t1, t2, t3 )
2     %EVALRFN Evaluates a response function returning a matrix of values
3     %   Description: This function evaluates the response function for
4     %   that has been previously constructed
5     %
6     %   Parameters: rfn - response function struct
7     %               t1  - matrix of t1 times
8     %               t2  - t2 delay (usually scalar)
9     %               t3  - matrix of t3 times
10    %
11    %   Output: response - response in time domain
12    %
13
14    % Begin timing
15    ticID = tic;
16
17    %The type was decided in buildEcho.m, and each term is subsequently
      added
18    %to the same y value
19    response = zeros(size(t1));
20    for ii = 1:rfn.nterm
21    switch rfn.rtype(ii);
22    case 1
23    response = response + r1val(rfn, ii, t1, t2, t3);
24    case 2
25    response = response + r2val(rfn, ii, t1, t2, t3);
```

```
26   case 3
27   response = response + r3val(rfn, ii, t1, t2, t3);
28   case 4
29   response = response + r4val(rfn, ii, t1, t2, t3);
30   case 5
31   response = response - conj(r1val(rfn, ii, t1, t2, t3));
32   case 6
33   response = response - conj(r2val(rfn, ii, t1, t2, t3));
34   case 7
35   response = response - conj(r3val(rfn, ii, t1, t2, t3));
36   case 8
37   response = response - conj(r4val(rfn, ii, t1, t2, t3));
38   end
39   end
40
41   % End Timing
42   time_elapsed = toc(ticID);
43   outStr = strcat('Time elapsed during evalrfn: ',num2str(time_elapsed));
44   disp(outStr)
45
46   End
47
48   function y = r1val(rfn, ii, t1, t2, t3)
49   %R1VAL - returns value of r1
50   %Each part of the response function is added to each other here,
51   %orientational, dephasing, the prefactor, and the time component.  This
     is
52   %the same for each r value.
53
```

```
54   y = zeros(size(t1));
55
56   % Comput rotational contribution
57
58   %Most of these values were determined in buildRot.m
59   for j=1:rfn.nYTerms(ii)
60   y = y+yVal(t1,t2,t3, rfn.yType(ii,j),rfn.dor)*rfn.yFac(ii,j);
61   end
62
63   % First Calculate dephasing functions (JCP, 2001, 115, 9266)
64
65   y = y.*exp(-(conj(rfn.hfun{rfn.cc(ii),rfn.cc(ii)}(t3))...
66   +conj(rfn.hfun{rfn.bb(ii),rfn.bb(ii)}(t2))...
67   +rfn.hfun{rfn.dd(ii),rfn.dd(ii)}(t1+t2+t3)...
68   +conj(rfn.hfun{rfn.bb(ii),rfn.cc(ii)}(t3+t2)...
69   -rfn.hfun{rfn.bb(ii),rfn.cc(ii)}(-t3)...%conj?
70   -rfn.hfun{rfn.bb(ii),rfn.cc(ii)}(-t2))...%conj
71   +rfn.hfun{rfn.cc(ii),rfn.dd(ii)}(t1+t2)...
72   -rfn.hfun{rfn.cc(ii),rfn.dd(ii)}(t1+t2+t3)...
73   -rfn.hfun{rfn.cc(ii),rfn.dd(ii)}(-t3)...
74   +conj(rfn.hfun{rfn.bb(ii),rfn.dd(ii)}(-t1)...
75   -rfn.hfun{rfn.bb(ii),rfn.dd(ii)}(-t3-t2)...%conj, time ordering
76   -rfn.hfun{rfn.bb(ii),rfn.dd(ii)}(+t1+t2)...%conj
77   +rfn.hfun{rfn.bb(ii),rfn.dd(ii)}(-t3)))); %conj
78
79   % compute final values
80   y = rfn.prefac(ii)*exp(1i*(-rfn.w3(ii)*t3-rfn.w2(ii)*t2-
     rfn.w1(ii)*t1)).*y;
81   end
```

```
82  function y = r2val(rfn, ii, t1, t2, t3)
83  %R2VAL - returns value of r2
84
85  y = zeros(size(t1));
86
87  % Comput rotational contribution
88  for j=1:rfn.nYTerms(ii)
89  y = y+yVal(t1,t2,t3, rfn.yType(ii,j),rfn.dor)*rfn.yFac(ii,j);
90  end
91
92  % First calculate dephasing
93
94  y = y.*exp(conj(-(rfn.hfun{rfn.cc(ii),rfn.cc(ii)}(t3)...
95  +rfn.hfun{rfn.dd(ii),rfn.dd(ii)}(t1+t2)...
96  +conj(rfn.hfun{rfn.bb(ii),rfn.bb(ii)}(t2+t3))...
97  +rfn.hfun{rfn.bb(ii),rfn.cc(ii)}(-t2)...
98  -rfn.hfun{rfn.bb(ii),rfn.cc(ii)}(t3)...
99  -rfn.hfun{rfn.bb(ii),rfn.cc(ii)}(-t2-t3)...
100 +rfn.hfun{rfn.cc(ii),rfn.dd(ii)}(t1+t2+t3)...
101 -rfn.hfun{rfn.cc(ii),rfn.dd(ii)}(t1+t2)...
102 -rfn.hfun{rfn.cc(ii),rfn.dd(ii)}(t3)...
103 +rfn.hfun{rfn.bb(ii),rfn.dd(ii)}(t1)...
104 -rfn.hfun{rfn.bb(ii),rfn.dd(ii)}(t3+t1+t2)...
105 -rfn.hfun{rfn.bb(ii),rfn.dd(ii)}(-t2)...
106 +rfn.hfun{rfn.bb(ii),rfn.dd(ii)}(t3))));
107
108 y = rfn.prefac(ii)*exp(1i*(-
    rfn.w3(ii)*t3+rfn.w2(ii)*t2+rfn.w1(ii)*t1)).*y;
109
```

```
110  end
111  function y = r3val(rfn, ii, t1, t2, t3)
112  %R3VAL - returns value of r3
113
114  y = zeros(size(t1));
115
116  % Comput rotational contribution
117  for j=1:rfn.nYTerms(ii)
118  y = y+yVal(t1,t2,t3, rfn.yType(ii,j),rfn.dor)*rfn.yFac(ii,j);
119  end
120
121  % Calculate Dephasing
122
123  y = y.*exp(conj(-(conj(rfn.hfun{rfn.bb(ii),rfn.bb(ii)}(t3))...
124  +rfn.hfun{rfn.cc(ii),rfn.cc(ii)}(t2+t3)...
125  +rfn.hfun{rfn.dd(ii),rfn.dd(ii)}(t1)...
126  +rfn.hfun{rfn.cc(ii),rfn.dd(ii)}(t2+t3+t1)...
127  -rfn.hfun{rfn.cc(ii),rfn.dd(ii)}(t2+t3)...
128  -rfn.hfun{rfn.cc(ii),rfn.dd(ii)}(t1)...
129  -rfn.hfun{rfn.bb(ii),rfn.cc(ii)}(t3+t2)...
130  -rfn.hfun{rfn.bb(ii),rfn.cc(ii)}(-t3)...
131  +rfn.hfun{rfn.bb(ii),rfn.cc(ii)}(t2)...
132  -rfn.hfun{rfn.bb(ii),rfn.dd(ii)}(t3+t2+t1)...
133  +rfn.hfun{rfn.bb(ii),rfn.dd(ii)}(t3+t2)...
134  +rfn.hfun{rfn.bb(ii),rfn.dd(ii)}(t2+t1)...
135  -rfn.hfun{rfn.bb(ii),rfn.dd(ii)}(t2))));
136
137  y = rfn.prefac(ii)*exp(1i*(-
     rfn.w3(ii)*t3+rfn.w2(ii)*t2+rfn.w1(ii)*t1)).*y;
```

```
138  end
139  function y = r4val(rfn, ii, t1, t2, t3)
140  %R4VAL - returns value of r1
141  y = zeros(size(t1));
142
143  % Comput rotational contribution
144  for j=1:rfn.nYTerms(ii)
145  y = y+yVal(t1,t2,t3, rfn.yType(ii,j),rfn.dor)*rfn.yFac(ii,j);
146  end
147
148  %Calculate Dephasing
149
150  y = y.*exp(-(rfn.hfun{rfn.bb(ii),rfn.bb(ii)}(t3)...
151  +rfn.hfun{rfn.cc(ii),rfn.cc(ii)}(t2)...
152  +rfn.hfun{rfn.dd(ii),rfn.dd(ii)}(t1)...
153  +rfn.hfun{rfn.bb(ii),rfn.cc(ii)}(t3+t2)...
154  -rfn.hfun{rfn.bb(ii),rfn.cc(ii)}(t3)...
155  -rfn.hfun{rfn.bb(ii),rfn.cc(ii)}(t2)...
156  +rfn.hfun{rfn.cc(ii),rfn.dd(ii)}(t1+t2)...%time ordering
157  -rfn.hfun{rfn.cc(ii),rfn.dd(ii)}(t2)...
158  -rfn.hfun{rfn.cc(ii),rfn.dd(ii)}(t1)...
159  +rfn.hfun{rfn.bb(ii),rfn.dd(ii)}(t3+t2+t1)...
160  -rfn.hfun{rfn.bb(ii),rfn.dd(ii)}(t3+t2)...
161  -rfn.hfun{rfn.bb(ii),rfn.dd(ii)}(t2+t1)...
162  +rfn.hfun{rfn.bb(ii),rfn.dd(ii)}(t2)));
163
164  y = rfn.prefac(ii)*exp(1i*(-rfn.w3(ii)*t3-rfn.w2(ii)*t2-
     rfn.w1(ii)*t1)).*y;
165  end
```

A.5.8 *evalrfn_rlx.m*

This is the same as evalrfn.m except for a couple of lines of code. Refer to evalrfn.m for more detailed comments. The population relaxation portion of this simulation code needs more work.

```
1    function response = evalrfn( rfn, t1, t2, t3 )
2    %EVALRFN Evaluates a response function returning a matrix of values
3    %   Description: This function evaluates the response function for
4    %   that has been previously constructed.  Only comments on portions
5    %   different from evalrfn.m are included.
6
7    %!!Warning!!! - I think the population relaxation portion of this code
8    %needs further work.  Modes do not seem to decay properly when plotting
     a
9    %spectral feature over multiple tau2 points.
10   %
11   %   Parameters: rfn - response function struct
12   %               t1  - matrix of t1 times
13   %               t2  - t2 delay (usually scalar)
14   %               t3  - matrix of t3 times
15   %
16   %   Output: response - response in time domain
17   %
18
19   % Begin timing
20   ticID = tic;
21
22   response = zeros(size(t1));
23   for ii = 1:rfn.nterm
24   switch rfn.rtype(ii);
```

```
25   case 1
26   response = response + r1val(rfn, ii, t1, t2, t3);
27   case 2
28   response = response + r2val(rfn, ii, t1, t2, t3);
29   case 3
30   response = response + r3val(rfn, ii, t1, t2, t3);
31   case 4
32   response = response + r4val(rfn, ii, t1, t2, t3);
33   case 5
34   response = response - conj(r1val(rfn, ii, t1, t2, t3));
35   case 6
36   response = response - conj(r2val(rfn, ii, t1, t2, t3));
37   case 7
38   response = response - conj(r3val(rfn, ii, t1, t2, t3));
39   case 8
40   response = response - conj(r4val(rfn, ii, t1, t2, t3));
41   end
42   end
43
44   % End Timing
45   time_elapsed = toc(ticID);
46   outStr = strcat('Time elapsed during evalrfn: ',num2str(time_elapsed));
47   disp(outStr)
48
49   End
50
51   function y = r1val(rfn, ii, t1, t2, t3)
52   %R1VAL - returns value of r1
53
```

```
54   y = zeros(size(t1));
55
56   % Comput rotational contribution
57   for j=1:rfn.nYTerms(ii)
58   y = y+yVal(t1,t2,t3, rfn.yType(ii,j),rfn.dor)*rfn.yFac(ii,j);
59   end
60
61   % First Calculate dephasing functions
62   y = y.*exp(-(conj(rfn.hfun{rfn.cc(ii),rfn.cc(ii)}(t3))...
63   +conj(rfn.hfun{rfn.bb(ii),rfn.bb(ii)}(t2))...
64   +rfn.hfun{rfn.dd(ii),rfn.dd(ii)}(t1+t2+t3)...
65   +conj(rfn.hfun{rfn.bb(ii),rfn.cc(ii)}(t3+t2)...
66   -rfn.hfun{rfn.bb(ii),rfn.cc(ii)}(t3)...
67   -rfn.hfun{rfn.bb(ii),rfn.cc(ii)}(t2))...
68   +rfn.hfun{rfn.cc(ii),rfn.dd(ii)}(t1+t2)...
69   -rfn.hfun{rfn.cc(ii),rfn.dd(ii)}(t1+t2+t3)...
70   -rfn.hfun{rfn.cc(ii),rfn.dd(ii)}(-t3)...
71   +conj(rfn.hfun{rfn.bb(ii),rfn.dd(ii)}(-t1)...
72   -rfn.hfun{rfn.bb(ii),rfn.dd(ii)}(t2+t3)...
73   -rfn.hfun{rfn.bb(ii),rfn.dd(ii)}(-t1-t2)...
74   +rfn.hfun{rfn.bb(ii),rfn.dd(ii)}(t3))));
75
76   % Compute population relaxation component - this is the only difference
77   % from evalrfn.m
78   y = y.*exp(-rfn.T1_1(ii)*t1-rfn.T1_2(ii)*t2-rfn.T1_3(ii)*t3);
79
80   % compute final values
81   y = rfn.prefac(ii)*exp(1i*(-rfn.w3(ii)*t3-rfn.w2(ii)*t2-
     rfn.w1(ii)*t1)).*y;
```

```
82   end
83   function y = r2val(rfn, ii, t1, t2, t3)
84   %R2VAL - returns value of r2
85
86   y = zeros(size(t1));
87
88   % Comput rotational contribution
89   for j=1:rfn.nYTerms(ii)
90   y = y+yVal(t1,t2,t3, rfn.yType(ii,j),rfn.dor)*rfn.yFac(ii,j);
91   end
92
93   % First calculate dephasing
94   y = y.*exp(conj(-(rfn.hfun{rfn.cc(ii),rfn.cc(ii)}(t3)...
95   +rfn.hfun{rfn.dd(ii),rfn.dd(ii)}(t1+t2)...
96   +conj(rfn.hfun{rfn.bb(ii),rfn.bb(ii)}(t2+t3))...
97   +rfn.hfun{rfn.bb(ii),rfn.cc(ii)}(-t2)...
98   -rfn.hfun{rfn.bb(ii),rfn.cc(ii)}(t3)...
99   -rfn.hfun{rfn.bb(ii),rfn.cc(ii)}(-t2-t3)...
100  +rfn.hfun{rfn.cc(ii),rfn.dd(ii)}(t1+t2+t3)...
101  -rfn.hfun{rfn.cc(ii),rfn.dd(ii)}(t1+t2)...
102  -rfn.hfun{rfn.cc(ii),rfn.dd(ii)}(t3)...
103  +rfn.hfun{rfn.bb(ii),rfn.dd(ii)}(t1)...
104  -rfn.hfun{rfn.bb(ii),rfn.dd(ii)}(t3+t1+t2)...
105  -rfn.hfun{rfn.bb(ii),rfn.dd(ii)}(-t2)...
106  +rfn.hfun{rfn.bb(ii),rfn.dd(ii)}(t3))));
107
108  % Compute population relaxation component
109  y = y.*exp(-rfn.T1_1(ii)*t1-rfn.T1_2(ii)*t2-rfn.T1_3(ii)*t3);
110
```

```
111  y = rfn.prefac(ii)*exp(1i*(-
     rfn.w3(ii)*t3+rfn.w2(ii)*t2+rfn.w1(ii)*t1)).*y;
112
113  end
114  function y = r3val(rfn, ii, t1, t2, t3)
115  %R3VAL - returns value of r3
116
117  y = zeros(size(t1));
118
119  % Comput rotational contribution
120  for j=1:rfn.nYTerms(ii)
121  y = y+yVal(t1,t2,t3, rfn.yType(ii,j),rfn.dor)*rfn.yFac(ii,j);
122  end
123
124  y = y.*exp(conj(-(conj(rfn.hfun{rfn.bb(ii),rfn.bb(ii)}(t3))...
125  +rfn.hfun{rfn.cc(ii),rfn.cc(ii)}(t2+t3)...
126  +rfn.hfun{rfn.dd(ii),rfn.dd(ii)}(t1)...
127  +rfn.hfun{rfn.cc(ii),rfn.dd(ii)}(t2+t3+t1)...
128  -rfn.hfun{rfn.cc(ii),rfn.dd(ii)}(t2+t3)...
129  -rfn.hfun{rfn.cc(ii),rfn.dd(ii)}(t1)...
130  -rfn.hfun{rfn.bb(ii),rfn.cc(ii)}(t3+t2)...
131  -rfn.hfun{rfn.bb(ii),rfn.cc(ii)}(-t3)...
132  +rfn.hfun{rfn.bb(ii),rfn.cc(ii)}(t2)...
133  -rfn.hfun{rfn.bb(ii),rfn.dd(ii)}(t3+t2+t1)...
134  +rfn.hfun{rfn.bb(ii),rfn.dd(ii)}(t3+t2)...
135  +rfn.hfun{rfn.bb(ii),rfn.dd(ii)}(t2+t1)...
136  -rfn.hfun{rfn.bb(ii),rfn.dd(ii)}(t2))));
137
138  % Compute population relaxation component
```

```
139  y = y.*exp(-rfn.T1_1(ii)*t1-rfn.T1_2(ii)*t2-rfn.T1_3(ii)*t3);
140
141  % Compute final value
142  y = rfn.prefac(ii)*exp(1i*(-
     rfn.w3(ii)*t3+rfn.w2(ii)*t2+rfn.w1(ii)*t1)).*y;
143  end
144  function y = r4val(rfn, ii, t1, t2, t3)
145  %R4VAL - returns value of r1
146  y = zeros(size(t1));
147
148  % Comput rotational contribution
149  for j=1:rfn.nYTerms(ii)
150  y = y+yVal(t1,t2,t3, rfn.yType(ii,j),rfn.dor)*rfn.yFac(ii,j);
151  end
152
153  y = y.*exp(-(rfn.hfun{rfn.bb(ii),rfn.bb(ii)}(t3)...
154  +rfn.hfun{rfn.cc(ii),rfn.cc(ii)}(t2)...
155  +rfn.hfun{rfn.dd(ii),rfn.dd(ii)}(t1)...
156  +rfn.hfun{rfn.bb(ii),rfn.cc(ii)}(t3+t2)...
157  -rfn.hfun{rfn.bb(ii),rfn.cc(ii)}(t3)...
158  -rfn.hfun{rfn.bb(ii),rfn.cc(ii)}(t2)...
159  +rfn.hfun{rfn.cc(ii),rfn.dd(ii)}(t1+t2)...
160  -rfn.hfun{rfn.cc(ii),rfn.dd(ii)}(t2)...
161  -rfn.hfun{rfn.cc(ii),rfn.dd(ii)}(t1)...
162  +rfn.hfun{rfn.bb(ii),rfn.dd(ii)}(t3+t2+t1)...
163  -rfn.hfun{rfn.bb(ii),rfn.dd(ii)}(t3+t2)...
164  -rfn.hfun{rfn.bb(ii),rfn.dd(ii)}(t2+t1)...
165  +rfn.hfun{rfn.bb(ii),rfn.dd(ii)}(t2)));
166
```

```
167   y = y.*exp(-rfn.T1_1(ii)*t1-rfn.T1_2(ii)*t2-rfn.T1_3(ii)*t3);
168
169   y = rfn.prefac(ii)*exp(1i*(-rfn.w3(ii)*t3-rfn.w2(ii)*t2-
      rfn.w1(ii)*t1)).*y;
170   end
```

A.5.9 *initExperiment.m*

This function puts all of the experimental parameters together in one place so they are easily transferred to other functions.

```
1     function expr = initExperiment(fieldFreq, measurement, polarization)
2     % INITEXPERIMENT Creates object holding all experimental parameters
3     %   Description: Creates a structure that holds all of the parameters
4     %   relevant to experimental setup
5     %
6     %   Parameters: fieldFreq - 4 element vector containg center frequency
      of
7     %                           input and local oscillator fields
8     %               measurement - String denoting desired experiment, right
      now
9     %                           that is only a 2D IR experiment
10    %               polarization - field polarizations 'ZZZZ' or 'YYZZ'
11
12    expr.w = fieldFreq;
13    expr.measurement = measurement;
14    expr.polar = polarization;
15    end
```

A.5.10 *initSystem.m*

This function puts all of the system parameters together in one place so they can easily be transferred to other functions.

```
1    function sys = initSystem(nStates, energies, dipoles, dipoleDir,
     rho0,...
2    hfun, dor, popRlx)
3    % initSystem Create structure that hold parameters of molecular system
4    %   Description - Function creates a object that stores all parameter
5    %   of the material system required to calculate nonlinear signal
6    %
7    %   Parameters: nStates - Number of system eigenstates
8    %               energeis - vector containing eigenstate energies
     (cm^-1)
9    %               dipoles - matrix of transition dipole moments
10   %               dipoleDir - 3d Matrix defining projection of dipoles
     onto
11   %                 each of the cartesian axes in the molecular frame
12   %               hfun - matrix of lineshape functions
13   %               dor - orientational diffusion coefficient
14   %               popRlx - optional vector of population relaxation times
15   %
16   %   Returns: sys - structure containing system information
17   %            sys.nStates
18   %            sys.w - transition frequency matrix, the difference
     between
19   %                   frequencies of each energy level
20   %            sys.mu - transition dipole matrix
21   %            sys.muDir - directional cosine information
```

```
22    %            sys.hfun - lineshpe functions
23    %            sys.dor - orientational diffusion coefficient
24
25    sys.nStates = nStates;
26    sys.mu = dipoles;
27    sys.muDir = dipoleDir;
28    sys.rho0 = rho0;
29    sys.hfun = hfun;
30    sys.dor = dor;
31
32    % Optional poprlx
33    if nargin == 8
34    sys.popRlx = popRlx;
35    end
36
37    % Construct transition frequency matrix
38    sys.w = zeros(nStates,nStates);
39    for i = 1:nStates
40    for j = i:nStates
41    sys.w(i,j) = energies(i) - energies(j);
42    sys.w(j,i) = -sys.w(i,j);
43    end
44    end
45
46    end
```

A.5.11 *printRandY.m*

This function is most useful as a check of all parts of the orientational response function.

```
1     function printRandY( rfn, outFile )
```

```
2
3    %This function files prints out of the orientational components that
     are
4    %calculated.  This can be used to check and make sure that both your
5    %relative angles are correctly distributed to the x,y,z axes and that
     all
6    %terms for the orientational response have been calculated.
7
8    fid = fopen(strcat(outFile, '.txt'),'w');
9    for ii = 1:rfn.nterm
10   fprintf(fid,'R%d(%d,%d,%d,%d)\r\n',rfn.rtype(ii),rfn.aa(ii),...
11   rfn.bb(ii),rfn.cc(ii),rfn.dd(ii));
12   fprintf(fid,'Num Y Terms: %d\r\n',rfn.nYTerms(ii));
13   for jj = 1:rfn.nYTerms(ii)
14   fprintf(fid,'Type: %d       Prefactor: %d\r\n',rfn.yType(ii,jj),...
15   rfn.yFac(ii,jj));
16   end
17   end
18   fclose(fid);
19
20   end
```

A.5.12 *printTerms.m*

This function is also most useful for checking that all response functions are accounted for.

```
1    function printTerms( rfn, outFile )
2    %PRINTTERMS Prints the terms in the response function
3    %    Description: Prints the terms of a response function i.e.
4    %    'R1(0,1,0,1)'
5    %
```

```
6     %    Parameters: rfn - struct containing response function
7     %                outFile - baseName for output. response functions will
      be
8     %                  printed in the file '[outFile].txt'
9
10    fid = fopen(strcat(outFile, '.txt'),'w');
11    for ii = 1:rfn.nterm
12    fprintf(fid,'R%d(%d,%d,%d,%d)\r\n',rfn.rtype(ii),rfn.aa(ii),...
13    rfn.bb(ii),rfn.cc(ii),rfn.dd(ii));
14    end
15    fclose(fid);
16
17    end
```

A.5.13 *sp_SetOptions.m*

This file has to be manually updated for each system that 2D IR spectra are simulated for.

```
1     function opts = sp_SetOptions( varargin )
2     %SP_SETOPTIONS Returns a structure containing all options for
      simulation
3     %calculations
4     %   Description: For newer versions of MATLAB, updates the specified
5     %   paramenters automatically (here that is the commented out code),
      and then
6     %   returns the structuce containing all parameters.  For older
      versions of
7     %   MATLAB, this file has to be opened and modified by hand (as it is
8     %   written now).
9     %
10    %     Parameters: prefacThresh - Threshold for cut-off of terms with a
```

```
11  %                       prefactor less than this value.  (Should be set
    at
12  %                       least below the quadruple of lowest dipole
    moment, mu^4.)
13  %                 rwaThresh - Rotating wave approximation cut-off.
14  %                       Frequency differences larger than this value
    will
15  %                       show no coupling in simulation.
16  %                 tstep - Size of step used in the t1 and t3 time axes
17  %                 nsteps - Number of time steps taken in the t1 and t3
    time
18  %                       axes
19  %                 fftsize - Size that each time axis will be zero-
    padded to
20  %                       before Fourier transform.  This value should be
    a
21  %                       power of 2.
22  %                 l_poprlx - Option to turn on population relaxation.
23  %                       Either 'true' or 'false'.
24  %                 l_cspecOnly - Option to only save the correlation
25  %                       spectrum, and not the rephasing or non-
    rephasing
26  %                       spectra, to save disk space.  Either 'true' or
    'false'.
27
28  if (mod(nargin,2) ~= 0)
29  error('sp_SetOptions:ArgChk',...
30  'Odd number of parameters. Arguments must be in key-value pairs')
31  End
```

```
32
33  % % Set defaults
34  % keySet = {'prefacThresh','rwaThresh','tstep', 'nsteps','fftsize',...
35  %   'l_poprlx','l_cspecOnly'};
36  % defaults = [0.2, 80, 10.0, 2000,2048,false,false];
37  %
38  % optMap = containers.Map(keySet, defaults);
39  %
40  % for ii = 1:(nargin/2)
41  %   if optMap.isKey(varargin{2*ii-1})
42  %     optMap('isKey') = varargin{2*ii};
43  %   else
44  %     error('sp_SetOptions:KeyVal',...
45  %       'Option not found')
46  %   end
47  % end
48  %
49  % opts.prefacThresh = optMap('prefacThresh');
50  % opts.rwaThresh = optMap('rwaThresh');
51  % opts.tstep = optMap('tstep');
52  % opts.nsteps = optMap('nsteps');
53  % opts.fftsize = optMap('fftsize');
54  % opts.l_poprlx = optMap('l_poprlx');
55  % opts.l_cspecOnly = optMap('l_cspecOnly');
56
57  opts.prefacThresh = 0.08;
58  opts.rwaThresh = 120;
59  opts.tstep = 8;
60  opts.nsteps = 2000;
```

```
61    opts.fftsize = 2048;
62    opts.l_poprlx = false;
63    opts.l_cspecOnly = false;
64
65
66    end
```

A.5.14 *yVal.m*

This function is called by buildRot.m.

```
1     function y = yVal(t1, t2, t3, type, dor)
2     %YVAL returns the value of the dynamic orientational contribution term
3     %   Description: This function calculates the time-dependent
      contribution
4     %   to the orientational response.  Again these terms come from Table
      II in
5     %   JCP, 2001, 115, 297.
6     %
7     %   Parameters: t1 - t1 time
8     %               t2 - t2 time
9     %               t3 - t3 time
10    %               type - type of Y response
11    %               dor - Orientational diffusion coefficient
12    %
13    % Case is determined in buildRot.m
14    switch type
15    case 1
16    y = c1(t1,dor).*c1(t3,dor).*(1.0+0.8*c2(t2,dor));
17    case 2
18    y = c1(t1,dor).*c1(t3,dor).*(1.0-0.4*c2(t2,dor));
```

```
19   case 3
20   y = c1(t1,dor).*c1(t3,dor).*c2(t2,dor);
21   case 4
22   y = c1(t1,dor).*c1(t3,dor).*(1.0+0.2*c2(t2,dor));
23   case 5
24   y = c1(t1,dor).*c1(t3,dor).*c2(t2,dor);
25   otherwise
26   error('specPack:yVal', 'No response function of this type')
27   end
28
29   end
30
31   function y = c1(tt,dor)
32   %C1 Function for orientational response
33   y = exp(-2.0*dor*tt);
34   end
35
36   function y = c2(tt,dor)
37   %C2
38   y = exp(-6.0*dor*tt);
39   end
```

Appendix B:

Matlab Code for Determining Coupling Constants

This was the code used to determine the coupling constants reported in Chapter 2. There is the main file that defines the initial guesses for the energies and coupling constants in the local site basis.

B.1 Main File – First Guesses for Local Site Basis Parameters

The function file, coupling_fit_main.m, is where the first guesses for local site basis parameters are entered and where the experimental energies are compared to the diagonalized values from the local mode Hamiltonian.

```
1    clear all
2    global tren1
3
4    %These sections have the experimental values for the energies of each
     mode
5    %and their anharmonicities (in the normal mode basis)
6
7    % % % %trimer
8    % % % r = 2050;
9    % % % t = 2067;
10   % % % a = 2074;
11   % % % b = 2116;
12   % % % an_r = 21;
13   % % % an_t = 24;
```

```
14    % % % an_a = 19;
15    % % % an_b = 24;
16    % % % an_rt = 2;
17    % % % an_ra = 9;
18    % % % an_rb = 3;
19    % % % an_ta = 15;
20    % % % an_tb = 9;
21    % % % an_ab = 10;
22
23    %dimer in D2O
24    r = 2048;
25    t = 2026;
26    a = 2061;
27    b = 2096;
28    an_r = 19;
29    an_t = 15;
30    an_a = 28;
31    an_b = 28;
32    an_rt = 14;
33    an_ra = 10;
34    an_rb = 9;
35    an_ta = 2;
36    an_tb = 6;
37    an_ab = 11;
38
39    % % % %dimer in FA
40    % % % r = 2051;
41    % % % t = 2002;
42    % % % a = 2065;
```

```
43  % % % b = 2090;
44  % % % an_r = 18;
45  % % % an_t = 14;
46  % % % an_a = 27;
47  % % % an_b = 19;
48  % % % an_rt = 2;
49  % % % an_ra = 6;
50  % % % an_rb = 5;
51  % % % an_ta = 2;
52  % % % an_tb = 4;
53  % % % an_ab = 13;
54
55  %computes the energies of the two-quantum states
56  r2 = 2*r-an_r;
57  t2 = 2*t-an_t;
58  a2 = 2*a-an_a;
59  b2 = 2*b-an_b;
60  rt = r+t-an_rt;
61  ra = r+a-an_ra;
62  rb = r+b-an_rb;
63  ta = a+t-an_ta;
64  tb = b+t-an_tb;
65  ab = a+b-an_ab;
66
67  tren = [0 r t a b r2 t2 a2 b2 rt ra rb ta tb ab]';
68  tren1 = sort(tren); %puts them in terms of increasing energy (no longer
    following the format in line 67)
69
70
```

```
%Here is where the first guesses for energies in the local mode basis
are
%made

% Er1=2032;
% Et1=2004;
% Ea1=2053;
% Eb1=2089;
% Vrt1=25;
% Vra1=18;
% Vrb1=19;
% Vta1=17;
% Vtb1=22;
% Vab1=35;
% Dr1=21;
% Dt1=18;
% Da1=20;
% Db1=21;

%dimer in D2O
Er1=2037;
Et1=2003;
Ea1=2064;
Eb1=2090;
Vrt1=6;
Vra1=6;
Vrb1=6;
Vta1=6;
Vtb1=6;
```

```
99   Vab1=6;
100  Dr1=21;
101  Dt1=21;
102  Da1=21;
103  Db1=22;
104
105
106  %Constraints on the values
107
108  %trimer
109  lb = [2000,2000,2000, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2];
110  ub = [2080,2104,2146,30,30,30,30,30,30,40,40,40,40];
111
112  % dimer/D2O
113  % % lb = [2000,2000,2000,2000, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2];
114  % % ub = [2080,2090,2104,2146,30,30,30,30,30,30,40,40,40,40];
115
116  f0=[Er1  Ea1  Eb1 Vrt1 Vra1 Vrb1 Vta1 Vtb1 Vab1  Dr1  Dt1  Da1  Db1];
117  options=optimset('MaxFunEvals', 500000, 'MaxIter', 100000, 'TolFun',
     1e-200, 'TolX', 1e-350);
118  [f,resnorm,residual] = lsqnonlin('fit_coupling',f0,[],[],options);
119
120  load y
121
122  compmat=zeros(15,4);
123  % compmat(2:15,1)=f;
124  compmat(3:15,1)=f;
125  compmat(2,1)=f(1);
126  compmat(:,2)=tren1;
```

```
127   compmat(:,3)=y;
128   compmat(:,4)=y-tren1;
129
130   %Sum of the difference between the fit and the experimental.
131   residual=sum(abs(compmat(:,4)));
```

B.2 Calculating Difference Between the Experimental and Fit Values

This function file, fit_coupling.m, calls a function that diagonalizes the Hamiltonian, and then it calculates the difference between the experimental and fit values. The sum of these differences are minimized.

```
1     function s = fit_coupling(f)
2     global tren1
3
4     %Calls another function here that diagonalizes the local site basis
      matrix
5     y=fifteenLevHam(f);
6
7     %Difference btwn experimental values and the values from the
      diagonalized
8     %matrix
9     s1=abs(y(1)-tren1(1));
10    s2=abs(y(2)-tren1(2));
11    s3=abs(y(3)-tren1(3));
12    s4=abs(y(4)-tren1(4));
13    s5=abs(y(5)-tren1(5));
14    s6=abs(y(6)-tren1(6));
15    s7=abs(y(7)-tren1(7));
16    s8=abs(y(8)-tren1(8));
```

```
17    s9=abs(y(9)-tren1(9));
18    s10=abs(y(10)-tren1(10));
19    s11=abs(y(11)-tren1(11));
20    s12=abs(y(12)-tren1(12));
21    s13=abs(y(13)-tren1(13));
22    s14=abs(y(14)-tren1(14));
23    s15=abs(y(15)-tren1(15));
24
25    %Trying to minimize the sum of the absolute value of the differences,
      with
26    %more emphasis on some of the differences
27    s=s1+4*s2+4*s3+4*s4+4*s5+s6+s7+s8+s9+s10+s11+s12+s13+s14+s15;
```

B.3 EIGENSTATES OF THE LOCAL SITE BASIS HAMILTONIAN

This function file, fifteenLevHam.m, diagonalizes the Hamiltonian after the local mode energies are entered.

```
1     function y = fifteenLevHam(f)
2
3     % Er=f(1);
4     % Et=f(2);
5     % Ea=f(3);
6     % Eb=f(4);
7     % Vrt=f(5);
8     % Vra=f(6);
9     % Vrb=f(7);
10    % Vta=f(8);
11    % Vtb=f(9);
12    % Vab=f(10);
```

```
13   % Dr=f(11);
14   % Dt=f(12);
15   % Da=f(13);
16   % Db=f(14);
17   Er=f(1);
18   Et=f(1);
19   Ea=f(2);
20   Eb=f(3);
21   Vrt=f(4);
22   Vra=f(5);
23   Vrb=f(6);
24   Vta=f(7);
25   Vtb=f(8);
26   Vab=f(9);
27   Dr=f(10);
28   Dt=f(11);
29   Da=f(12);
30   Db=f(13);
31
32   %Placing guesses in the hamiltonian matrix
33   Htr=zeros(15,15);
34   Htr(2,2)=Er;
35   Htr(3,3)=Et;
36   Htr(4,4)=Ea;
37   Htr(5,5)=Eb;
38   Htr(6,6)=2*Er-Dr;
39   Htr(7,7)=2*Et-Dt;
40   Htr(8,8)=2*Ea-Da;
41   Htr(9,9)=2*Eb-Db;
```

```
42   Htr(10,10)=Er+Et;
43   Htr(11,11)=Er+Ea;
44   Htr(12,12)=Er+Eb;
45   Htr(13,13)=Et+Ea;
46   Htr(14,14)=Et+Eb;
47   Htr(15,15)=Eb+Ea;
48   Htr1=zeros(15,15);
49   Htr1(2,3)=Vrt;
50   Htr1(2,4)=Vra;
51   Htr1(2,5)=Vrb;
52   Htr1(3,4)=Vta;
53   Htr1(3,5)=Vtb;
54   Htr1(4,5)=Vab;
55   Htr1(6,10)=sqrt(2)*Vrt;
56   Htr1(6,11)=sqrt(2)*Vra;
57   Htr1(6,12)=sqrt(2)*Vrb;
58   Htr1(7,10)=sqrt(2)*Vrt;
59   Htr1(7,13)=sqrt(2)*Vta;
60   Htr1(7,14)=sqrt(2)*Vtb;
61   Htr1(8,11)=sqrt(2)*Vra;
62   Htr1(8,13)=sqrt(2)*Vta;
63   Htr1(8,15)=sqrt(2)*Vab;
64   Htr1(9,12)=sqrt(2)*Vrb;
65   Htr1(9,14)=sqrt(2)*Vtb;
66   Htr1(9,15)=sqrt(2)*Vab;
67   Htr2=Htr1';
68   Htr3=Htr1+Htr+Htr2;
69
70   %Diagonalizing the Hamiltonian
```

```
71    [~,D] = eig(Htr3);
72    y=diag(D);
73
74    save y y
```

Appendix C:

Matlab Code for Analyzing t-HDVE Raw Data

The raw data from the transient heterodyne detected vibrational echo (t-HDVE) experiments has to be treated before system information can be extracted. For the data presented in Lynch et al., JPCA, 2012, 116, 7023 and Lynch et al., JCP, 2012, 136, 241101, this is how the data was treated:

- Interpolate the ω_3 axis to equal frequency steps
- Fourier transform into the time domain
- Remove the negative time points
- Apply a window that multiplies the data points before the signal by zero, and multiplies all of the other time points by one
- Fourier transform back into the time domain, zero-padding to 64 points
- To obtain the correct frequency axis, compare it to an analysis where both Fourier transforms have been heavily zero-padded (8192 points), and add the needed offset to the frequency axis

The function files following include two that show how all the data files that LabView saves are loaded into the same matrix for easy use in the function file in Section C.3 where the raw data is actually treated as above.

C.1 Loading 3rd Order Data Files into Single Variable

The function file, pepsmat_3rd.m, contains the method used to loadt he data files for the 3rd order signal into a single variable.

```
1    clear all;
```

```
%This file loads all points saved in different files by LabView and puts
%them together in the same variable.  This version is for the 3rd order
%signal.

file='CN_mat_avg_3rd151001.dat';
file1='CN_mat_avg_3rd15101.dat';
filet='CN_time151001.dat';
filet1='CN_time15101.dat';


firstfile=1;
i=firstfile;
k=i;

lastfile=5;
m=lastfile;
l=m;


for t=k:l;
if t<10
sth=num2str(t);
file1(18)=sth;
filet1(11)=sth;
data=load(file1);
time=load(filet1);
matrixd11(:,:,t)=data;
matrixt11(:,:,t)=time;
```

```
30   else
31   tstr=num2str(t);
32   file(18)=tstr(1);
33   file(19)=tstr(2);
34   filet(11)=tstr(1);
35   filet(12)=tstr(2);
36   data=load(file);
37   time=load(filet);
38   matrixd11(:,:,t)=data;
39   matrixt11(:,:,t)=time;
40   end
41   end
42
43   file='CN_mat_avg_3rd152001.dat';
44   file1='CN_mat_avg_3rd15201.dat';
45   filet='CN_time152001.dat';
46   filet1='CN_time15201.dat';
47
48
49   firstfile=1;
50   i=firstfile;
51   k=i;
52
53   lastfile=28;
54   m=lastfile;
55   l=m;
56
57
58   for t=k:l;
```

```
59   if t<10
60   sth=num2str(t);
61   file1(18)=sth;
62   filet1(11)=sth;
63   data=load(file1);
64   time=load(filet1);
65   matrixd21(:,:,t)=data;
66   matrixt21(:,:,t)=time;
67   else
68   tstr=num2str(t);
69   file(18)=tstr(1);
70   file(19)=tstr(2);
71   filet(11)=tstr(1);
72   filet(12)=tstr(2);
73   data=load(file);
74   time=load(filet);
75   matrixd21(:,:,t)=data;
76   matrixt21(:,:,t)=time;
77   end
78   end
79
80
81   file='CN_mat_avg_3rd153001.dat';
82   file1='CN_mat_avg_3rd15301.dat';
83   filet='CN_time153001.dat';
84   filet1='CN_time15301.dat';
85
86   firstfile=1;
87   i=firstfile;
```

```
88   k=i;
89
90   lastfile=5;
91   m=lastfile;
92   l=m;
93
94
95   for t=k:l;
96   if t<10
97   sth=num2str(t);
98   file1(18)=sth;
99   filet1(11)=sth;
100  data=load(file1);
101  time=load(filet1);
102  matrixd31(:,:,t)=data;
103  matrixt31(:,:,t)=time;
104  else
105  tstr=num2str(t);
106  file(18)=tstr(1);
107  file(19)=tstr(2);
108  filet(11)=tstr(1);
109  filet(12)=tstr(2);
110  data=load(file);
111  time=load(filet);
112  matrixd31(:,:,t)=data;
113  matrixt31(:,:,t)=time;
114  end
115  end
116
```

```
117  [a b c]=size(matrixd11);
118  [a1 b1 c1]=size(matrixd21);
119  [a2 b2 c2]=size(matrixd31);
120  [at bt c]=size(matrixt11);
121  [at1 bt1 c1]=size(matrixt21);
122  [at2 bt2 c2]=size(matrixt31);
123
124
125  pepsmatd15=zeros(a,b,c+c1+c2);
126  pepsmatt15=zeros(at,bt,c+c1+c2);
127
128
129  pepsmatd15(:,:,1:c)=matrixd11;
130  pepsmatd15(:,:,c+1:c+c1)=matrixd21;
131  pepsmatd15(:,:,c+c1+1:end)=matrixd31;
132
133  pepsmatt15(:,:,1:c)=matrixt11;
134  pepsmatt15(:,:,c+1:c+c1)=matrixt21;
135  pepsmatt15(:,:,c+c1+1:end)=matrixt31;
136
137  save pepsmatt15 pepsmatt15
138  save pepsmatd15 pepsmatd15
```

C.2 LOADING 5TH ORDER DATA FILES INTO SINGLE VARIABLE

The function file, pepsmat_5th.m, contains the method used to load the data files for the 3rd order signal into a single variable.

```
1    clear all;
```

```
2    %This file loads all points saved in different files by LabView and
     puts
3    %them together in the same variable.  This version is for the 5th order
4    %signal.
5
6
7    file='CN_mat_avg_5th151001.dat';
8    file1='CN_mat_avg_5th15101.dat';
9
10
11   firstfile=1;
12   i=firstfile;
13   k=i;
14
15   lastfile=5;
16   m=lastfile;
17   l=m;
18
19
20   for t=k:l;
21   if t<10
22   sth=num2str(t);
23   file1(18)=sth;
24   data=load(file1);
25   matrixd11(:,:,t)=data;
26   else
27   tstr=num2str(t);
28   file(18)=tstr(1);
29   file(19)=tstr(2);
```

```
30   data=load(file);
31   matrixd11(:,:,t)=data;
32   end
33   end
34
35   file='CN_mat_avg_5th152001.dat';
36   file1='CN_mat_avg_5th15201.dat';
37
38   firstfile=1;
39   i=firstfile;
40   k=i;
41
42   lastfile=28;
43   m=lastfile;
44   l=m;
45
46
47   for t=k:l;
48   if t<10
49   sth=num2str(t);
50   file1(18)=sth;
51   data=load(file1);
52   matrixd21(:,:,t)=data;
53   else
54   tstr=num2str(t);
55   file(18)=tstr(1);
56   file(19)=tstr(2);
57   data=load(file);
58   matrixd21(:,:,t)=data;
```

```
59   end
60   end
61
62
63   file='CN_mat_avg_5th153001.dat';
64   file1='CN_mat_avg_5th15301.dat';
65
66   firstfile=1;
67   i=firstfile;
68   k=i;
69
70   lastfile=5;
71   m=lastfile;
72   l=m;
73
74
75   for t=k:l;
76   if t<10
77   sth=num2str(t);
78   file1(18)=sth;
79   data=load(file1);
80   matrixd31(:,:,t)=data;
81   else
82   tstr=num2str(t);
83   file(18)=tstr(1);
84   file(19)=tstr(2);
85   data=load(file);
86   matrixd31(:,:,t)=data;
87   end
```

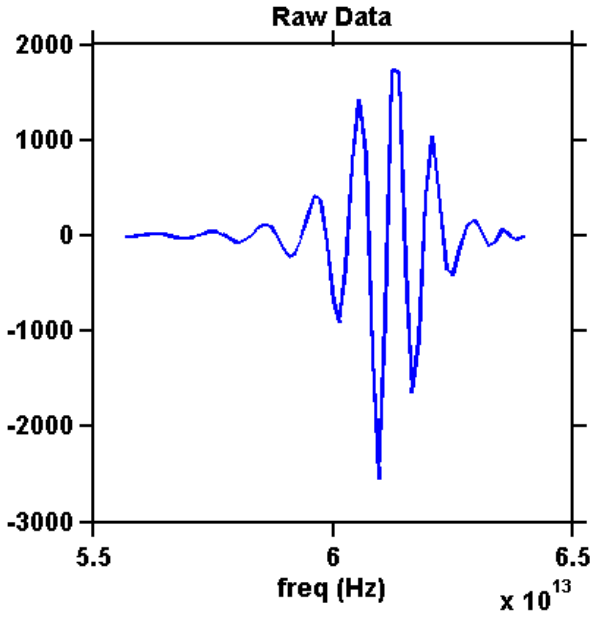
```
88    end
89
90    [a b c]=size(matrixd11);
91    [a1 b1 c1]=size(matrixd21);
92    [a2 b2 c2]=size(matrixd31);
93
94
95    pepsmatd15_5th=zeros(a,b,c+c1+c2);
96
97
98    pepsmatd15_5th(:,:,1:c)=matrixd11;
99    pepsmatd15_5th(:,:,c+1:c+c1)=matrixd21;
100   pepsmatd15_5th(:,:,c+c1+1:end)=matrixd31;
101
102   save pepsmatd15_5th pepsmatd15_5th
```

C.3 TREATING RAW T-HDVE DATA

This function file, pepsmat_analysis_0pad_time.m, contains the method used to treat the raw data from a t-HDVE experiment.

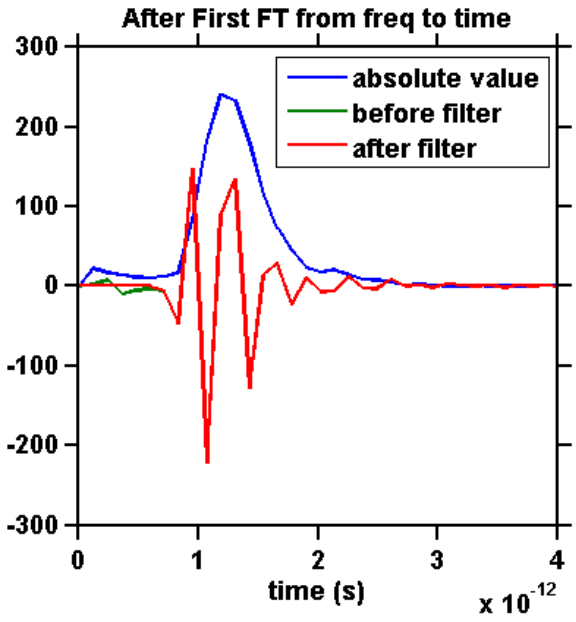
```
1     clear all
2     tic
3
4     %from pepsmat_3rd.m or pepsmat_5th.m
5     load pepsmatd2
6     dat=pepsmatd2;
7     load pepsmatt2
8     load w3
9     t1=-pepsmatt2(3,:,:);
```

```
10
11    [a1,b1,c1]=size(dat);
12
13    c=2.99792458E10; %speed of light in cm/s
14    f3=w3.*c; %converting from cm-1 to s-1
15    data2=sortrows(f3);
16    datan=f3;
17    xstep=f3(2)-f3(1);
18    x=round(datan(1)):xstep:round(datan(end));
19    px=length(x);
20
21    for m=1:c1; %m => tau2
22    for n=1:b1;  % n => tau1, 11 is tau1=0 for t-HDVE_02
23    %interpolates so there is equal spaces in the freq domain
24    y(:,n,m)=spline(f3,dat(:,n,m),x);
```




```
25    newx=x;
26    newy=zeros(length(newx),n,m);
27    newy(:,n,m)=y(:,n,m);
```

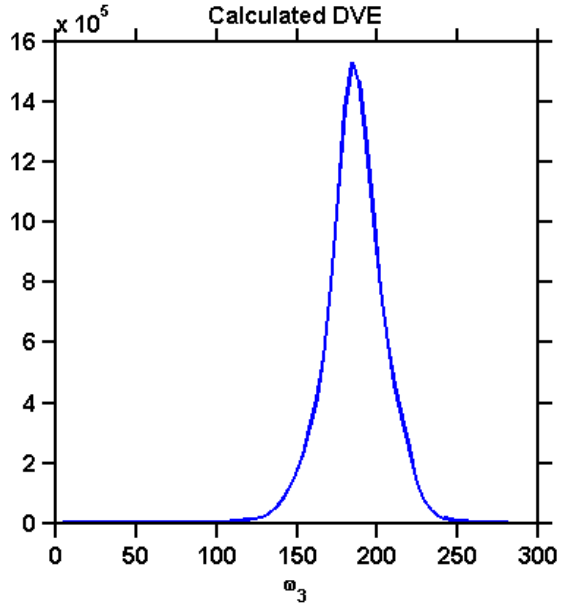
```
28
29  %first FT from freq to time domain (no zero padding)
30  ffm(:,n,m)=ifft(newy(:,n,m));
31  ffm1(:,n,m)=fftshift(ffm(:,n,m));
```

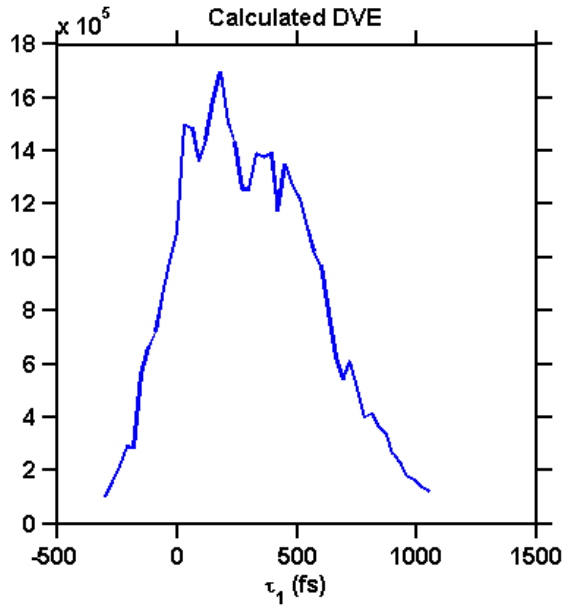



```
32  a=length(ffm(:,1,1));
33
34  %Defining the time axis after the first FT
35  time1=zeros(1,a);
36  for k=1:a
37  time1(k)=k/(a*xstep);
38  end
39  time2=time1(1:a/2);
40  time3=flipdim(time2,2).*(-1);
41  time=zeros(1,a+1);
42  time(1:a/2)=time3;
43  time(a/2+2:a+1)=time2;
44
45  %Putting first FT into the same points as new time axis
```

```
ffm3=zeros(a+1,n,m);
ffm3(1:a/2,n,m)=ffm1(1:a/2,n,m);
ffm3(a/2+2:end,n,m)=ffm1(a/2+1:end,n,m);
ffm2(:,n,m)=ffm3(a/2+1:end,n,m);
time(1:a/2+1)=[];

%This is the filter - a shifted heaviside function
fil=ones(length(ffm2(:,1)),n,m);
fil(1:6,n,m)=0; %This is where the filter is shifted, change '1:6'
gtime=time;

%The filter is applied to the data in the time domain here
filt(:,n,m)=ffm2(:,n,m).*fil(:,n,m);

%Second FT - zero padded to 64 to give the same number of points in
%the freq domain that we started with
stuff(:,n,m)=fft(filt(:,n,m),64);
fl=length(stuff(:,1,1));

%Defining the new freq axis - this needs to be compared to a
%version where the second FT was heavily zero-padded and the
%frequency is more accurately determined
newfreq=zeros(size(stuff(:,1,1)));
time_step=gtime(3)-gtime(2);
for p = 1:fl;
newfreq(p)=p/(fl*time_step);
end
w3new=newfreq./c;

```

```
75    %calculation of the pump-probe
76    dvepp(:,n,m)=stuff(:,n,m);
77
78    %calculation of the dispered vibrational echo
79    dve(:,n,m)=(abs(dvepp(:,n,m))).^2;
```






```
80
81    end
```

```
82   end
83
84   figure
85   plot(f3,dat(:,21,1),'linewidth',2)
86   axis square
87   set(gca, 'Tickdir', 'out');
88   set(gca, 'FontWeight','bold')
89   set(gca,'FontSize',14)
90   set(gca, 'FontName','Arial')
91   set(gca,'TickLength', [0.03, 0.01])
92   set(gca,'LineWidth', 2)
93   xlabel('freq (Hz)')
94   title('Raw Data')
95
96   figure
97
     plot(time,abs(ffm2(:,21,1)),time,ffm2(:,11,1),gtime,filt(:,11,1),'linew
     idth',2)
98   xlim([0 4e-12])
99   axis square
100  set(gca, 'Tickdir', 'out');
101  set(gca, 'FontWeight','bold')
102  set(gca,'FontSize',14)
103  set(gca, 'FontName','Arial')
104  set(gca,'TickLength', [0.03, 0.01])
105  set(gca,'LineWidth', 2)
106  xlabel('time (s)')
107  title('After First FT from freq to time')
108  legend('absolute value','before filter','after filter')
```

```
109
110  figure
111  plot(w3new,dve(:,21,1),'linewidth',2)
112  axis square
113  set(gca, 'Tickdir', 'out');
114  set(gca, 'FontWeight','bold')
115  set(gca,'FontSize',12)
116  set(gca, 'FontName','Arial')
117  set(gca,'TickLength', [0.03, 0.01])
118  set(gca,'LineWidth', 2)
119  xlabel('\omega_{3}')
120  title('Calculated DVE')
121
122  figure
123  plot(t1(1,:,1),dve(44,:,1),'linewidth',2)
124  axis square
125  set(gca, 'Tickdir', 'out');
126  set(gca, 'FontWeight','bold')
127  set(gca,'FontSize',12)
128  set(gca, 'FontName','Arial')
129  set(gca,'TickLength', [0.03, 0.01])
130  set(gca,'LineWidth', 2)
131  xlabel('\tau_1 (fs)')
132  title('Calculated DVE')
133
134
135  dvepp2_0pad_time_sfthv=dvepp;
136  dvecalc2_0pad_time_sfthv=dve;
137  w3new_0pad_time=w3new;
```

```
138
139 %These are the files that are saved in the 'raw' data file folder.
    This is
140 %the data Mike started with for the analysis in
141 %The Journal of Physical Chemistry A 116 (2012): 7023-7032. and
142 %The Journal of chemical physics 136 (2012): 241101.
143 save dvepp2_0pad_time_sfthv dvepp2_0pad_time_sfthv
144 save dvecalc2_0pad_time_sfthv dvecalc2_0pad_time_sfthv
145 save w3new_0pad_time w3new_0pad_time
146
147
148 toc % 1 min
```

VITA

Karla M. Slenkamp was born in Seattle, WA in 1987. After completing her work at Henry M. Jackson High School in 2005, she entered Western Washington University where she studied chemistry. There she receieved her Bachelor of Science, *Magna Cum Laude*, in June 2009, and entered the Graduate School at the University of Washington in September 2009. She completed her Doctor of Philosophy in Chemistry at UW in June 2015.