MIXED QUANTUM-CLASSICAL SIMULATIONS ON MODEL DONOR-BRIDGE-ACCEPTOR TRIADS

A Dissertation Presented to

the Faculty of the Department of Chemistry

University of Houston

In Partial Fulfillment

of the Requirements for the Degree

Doctor of Philosophy

By

Kush Patel

August 2019

MIXED QUANTUM-CLASSICAL SIMULATIONS ON MODEL DONOR-BRIDGE-ACCEPTOR TRIADS

Kush Patel

APPROVED:

Dr. Eric R. Bittner, Chairman

Dr. Vassiliy Lubchenko

Dr. Judy Wu

Dr. Shoujun Xu

Dr. Oomman Varghese

Dean, College of Natural Sciences and Mathematics

Acknowledgments

First and foremost, I would like to thank Dr. Eric Bittner. These six years have been challenging but an incredible period of growth for me, both professionally and personally. Thank you for continuing to push me forward despite my personal apprehensions. Thank you for helping me rekindle my lost love for mathematics and computer science. Above all, thank you for all the shenanigans. They have certainly made this whole experience entertaining. Your zest for life is contagious, and I hope to carry that forward. At the end of all this, I am still not sure if this path was correct for me. But, if I had to do this all over again, no other advisor would suffice. I also want to thank my other group members, Dr. Hao, Dr. Vlad, Dr. Allen, Robert, and Nosheen. Navigating the world of Bittner would not have been possible without you all.

I want to thank my wonderful mom, dad, and brother. Your support means the world to me. I have always looked to you three for guidance, and you have seldom let me down. I am sorry I am not a medical doctor, but I refuse to be just another average "Dr. Patel". I especially want to thank my brother. Bhai, you always knew what I needed in my best and my worst days.

I want to especially thank Dr. Vladimir Lankevich and Dr. Erin Finley, for their incredible support, both scientifically and personally. This last year has been exceptionally difficult for me. Thank you both for seeing me through this.

I would also like to thank my closest friends, NK, JB, JK, and AP. You boys have been by my side through thick and thin, and I could not have asked for more.

Finally, I want to thank Wikipedia. It certainly does not get the credit it deserves.

MIXED QUANTUM-CLASSICAL SIMULATIONS ON MODEL DONOR-BRIDGE-ACCEPTOR TRIADS

An Abstract of a Dissertation

Presented to

the Faculty of the Department of Chemistry

University of Houston

In Partial Fulfillment

of the Requirements for the Degree

Doctor of Philosophy

By

Kush Patel

August 2019

Abstract

Marcus theory predicts electron transfer rates between donor and acceptor systems. Since its inception in the 1950's, this theory has been widely applied to topics in many disciplines and has become an instrumental model in describing the efficiencies of photovoltaics. A key aspect of this theory requires knowledge of the nuclear reorganization of system and solvent alike and computation of energies associated with such nuclear motions is often difficult or impossible for systems with large number of degrees of freedom. We develop here a mixed quantum mechanical/molecular dynamics model to investigate charge transfer dynamics in a set of large organic Donor-Bridge-Acceptor triad molecules. Specifically, we are interested in the differences in electron and nuclear behavior relating to small changes in the molecular makeup of Carotenoid-Porphyrin-Fullerene triads.

Our model approximates excitation energies on the order of 1.9 eV which agrees with absorption spectra for these triads and isolated porhyrins. Via electron population analysis, we monitor charge migration to the acceptor in time. Approximations of the charge transfer rates reveal ultrafast (picosecond scale) electron dynamics consistent with experimental literature. We then correlate nuclear dynamics with the charge transfer process using the Short-Time Fourier Transform technique. Broadly, the porphyrins undergo higher energy vibrations, whereas the fullerenes see low energy modes. Aryl side groups exhibit torsional motions relative to the porphyrin. Aryl linkers between bridge and acceptor are restricted from such motions and therefore express ring distortion modes. Finally, we find an amide linker mode that is directionally sensitive to electron motion.

This work supports the notion of vibrationally coupled ultrafast charge transfer found in both experimental and theoretical studies and lays a foundational method for identifying key vibrational modes for parameterizing future theoretical models.

Contents

List of Figures

Chapter 1

Introduction

In the mid-1950s, Rudolph Marcus published what would become the standard model for predicting rate constants in electron transfer (ET) mechanisms[1, 2]. In his originating work, he considers the self-exchange reaction between pairs of small Iron and Magnesium complexes. In particular, he wanted to reconcile the Franck-Condon Principle (FCP) and energy conservation in the context of these nonradiative reduction-oxidation reactions[3]. Under the FCP, an electron would jump from one ion to another instantaneously without an immediate response from slower nuclei. The resulting products are the same two ions ($Fe^{2+} + Fe^{3+} \rightleftharpoons Fe^{3+} + Fe^{2+}$), but each ion is now in a foreign, non-equilibrium environment. With larger ionic complexes, the change in the local electric fields is smaller after electron transfer, eliciting less response from the solvent. However, the nature of the bond between metal ion and ligand changes, and this becomes the foreign environment. Nuclear coordinates and their coupling to the electronic state are, therefore, an integral part of the potential energy description.

While the original work focused on small metal ions and complexes, the resulting

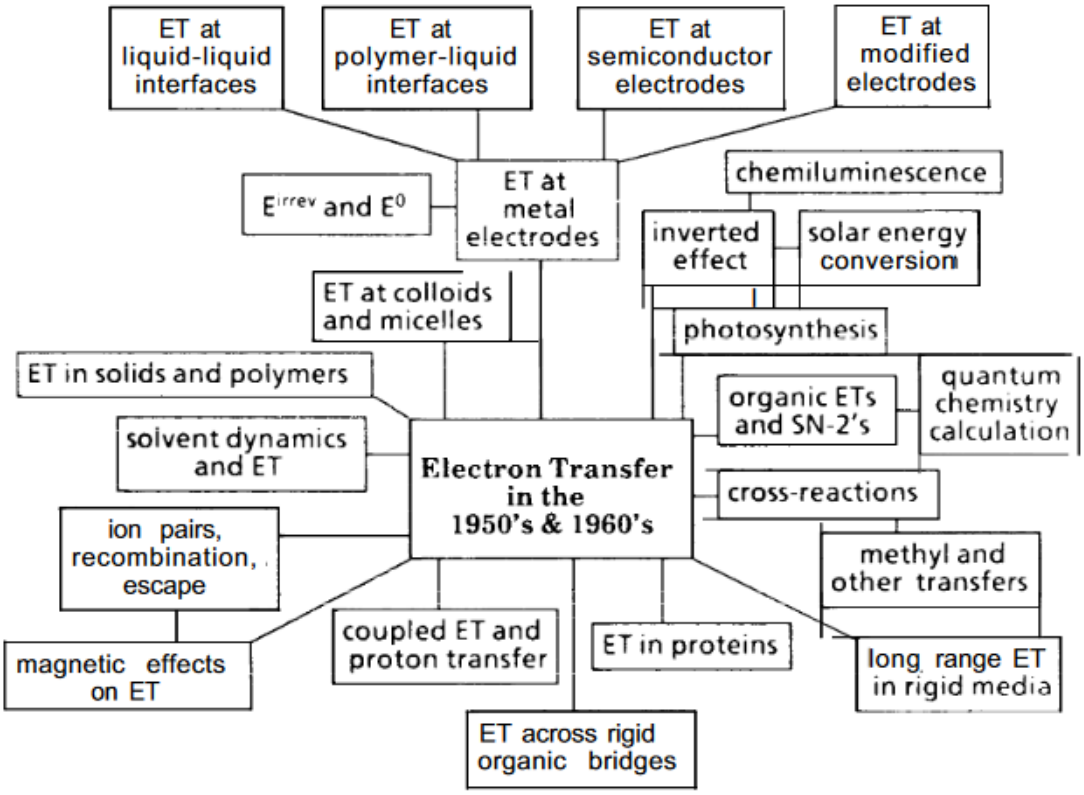

Figure 1.1: Selection of topics related to electron transfer. Reproduced with permission from Springer Science and Bus Media B V.

theory is widely predictive in many current topics as electron transfer is at the heart of the vast majority of chemical and biochemical processes. Figure 1.1 depicts a small subset of related topics and illustrates the ubiquity of this theory.

Marcus aimed to describe the rate of ET in the Arrhenius (kinetic) sense, $k \propto e^{-\Delta G^{\ddagger}/RT}$. Here, $\Delta G^{\ddagger}$ is the activation energy between an initial and final electronic state. Figure 1.2 plots the parabolic form (2^{nd} order Taylor expansion) of the free energy surfaces along a single normal coordinate for such states. With this approximation, one can relate the activation energy to two parameters, λ and ΔG^0. With exponential prefactor from the Fermi Golden Rule, one can then elegantly write the

transition rate as

$$k_{ET} = \frac{2\pi |V|^2}{\hbar} \frac{1}{\sqrt{4\pi\lambda k_B T}} \exp\left[-\frac{(\Delta G^0 + \lambda)^2}{4\lambda k_B T}\right] \tag{1.1}$$

where V is the electronic coupling between the initial and final states; λ is the reorganization energy associated with the nuclear relaxation from the minima of the first state to the minimum of the final state along the surface of the final state; and ΔG^0 is the free energy offset (driving force) between the states. The reorganization energy term includes both the rearrangement of the ions of interest and solvent molecule reorientation as a result of the redistribution of charge. Many experimental comparisons highlight the predictive power of this theory[4], and it remains as a comparison point for new theoretical models[5, 6, 7].

The Gaussian-like exponential term dictates that the rate is maximal, for a given V, when $-\Delta G^0 = \lambda$. Most electron transfer reactions lie within the normal Marcus regime, where $-\Delta G^0 < \lambda$, as depicted in Figure 1.2. For $-\Delta G^0 > \lambda$, this is the inverted regime. In this region, while the driving force becomes increasingly favorable, the wavefunction overlap between the considered states vanishes, and therefore reaction rate drops[8]. Evidence of this case was found 30 years later by Miller and coworkers[9].

While it is possible to compute ΔG, λ, and V in smaller or ordered systems, the task becomes much more formidable when considering large or non-rigid systems. Computation of ΔG^0 and λ require not only geometry optimization and energy calculations for the ground state, but also those for multiple excited states, which are significantly more involved. One can approximate such parameters from diabatic surfaces, but they are not uniquely defined. One can uniquely define the adiabatic

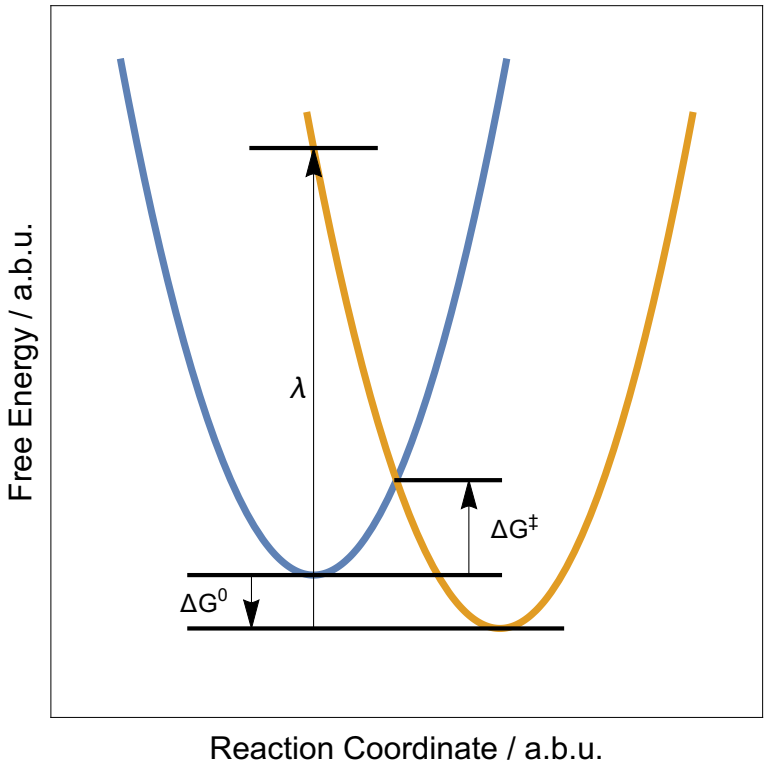

Figure 1.2: Schematic representation of Marcus electron transfer theory. Parabolas represent the energy surfaces of the initial (blue) and final (orange) states with respect to an arbitrary reaction coordinate, where the minima correspond to an equilibrium geometry. Terms depicted are ΔG^0, the free energy offset, and λ, the reorganization energy.

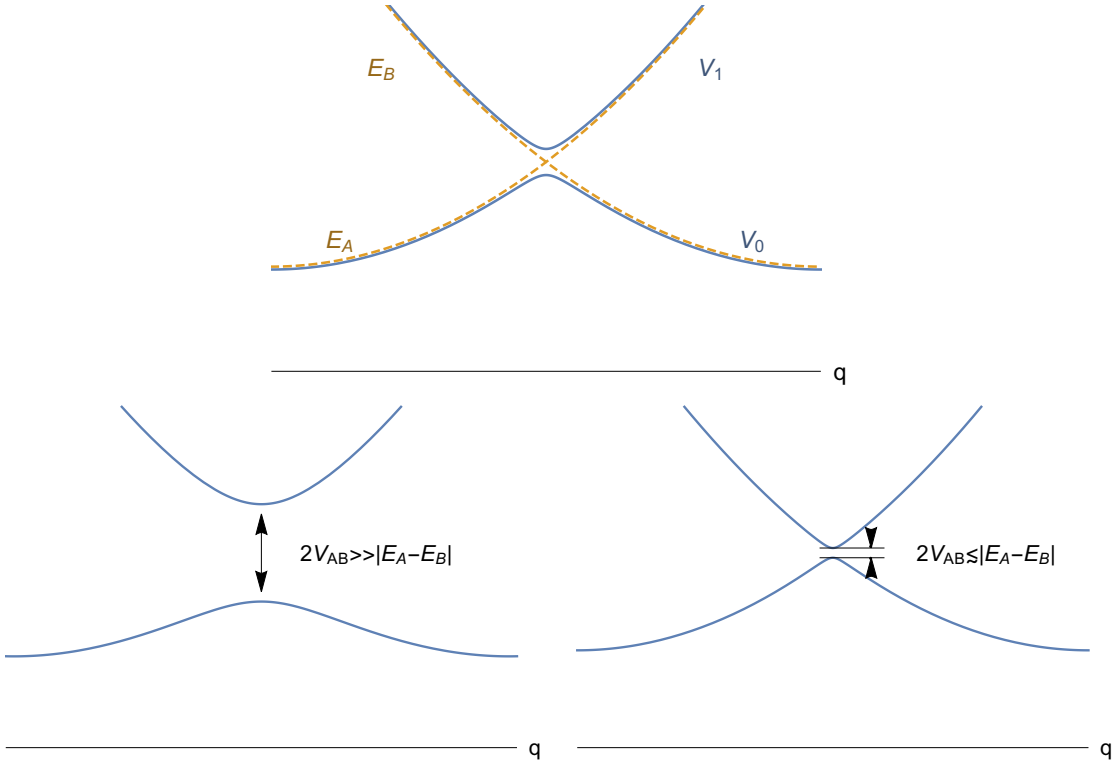

Figure 1.3: Schematic figure of adiabatic (blue) potentials surfaces (arbitrary energy unit) as a function of arbitrary coordinate q. These are one dimensional cross-sections of an otherwise high dimensional coordinate space. (Top) Adiabatic surfaces plotted with resulting diabatic surfaces (orange). (Bottom, left) Adibatic potentials under strong coupling regime. (Bottom, right) Adiabatic potentials in weak coupling regime.

potentials along a given nuclear coordinate via Born-Oppenheimer Approximation

$$V_n(R) = \langle \psi_n(q) | \, \mathcal{H}_{el}(q) \, | \psi_n(q) \rangle \tag{1.2}$$

giving rise to a series of potentials $V_n(q)$ visualized in Figure 1.3. Such potentials may come arbitrarily close but will not cross, unless allowed by symmetry.

Diabatization involves assuming that away from the avoided crossing, each adiabatic state can be written as a linear combination of some other states such that in this basis

$$\mathcal{H} = \begin{pmatrix} E_A(q) & V_{AB} \\ V_{AB} & E_B(q) \end{pmatrix} \tag{1.3}$$

and the eigenvalues of $\mathcal{H}$ are the same as those of the Born-Oppenheimer Hamiltonian. $E_A(q)$ and $E_B(q)$ are the "diabatic" potentials. So long as $|E_A - E_B| > 2V_{AB}$, the diabatic potentials provide a robust and accurate description. In this limit, a perturbative expansion of the states in terms of $\frac{V_{AB}}{|E_A - E_B|}$ converges rapidly. However, close to the avoided crossing, the diabatic states become strongly mixed and perturbation theory is no longer valid.

Furthermore, free energy surfaces are not dependent on merely one dimension or normal coordinate as pictured; they are multivariate with respect to the system under study. Nevertheless, if these obstacles can be overcome, this theory remains not only viable for determining electron transfer rates for inorganic, organometallic, and organic molecules but stands as a comparison point for new models[4, 5, 6, 7]. The reorganization energy encapsulates any nuclear dynamics that occur as a result of changes in the electronic state. As the term is a significant factor this description of electron transfer, identifying which normal coordinates of a given system are significant to the transition will not only give insight into better molecular construction but can also reduce the number of degrees of freedom considered in models.

1.1 Vibronic Transitions

Each electronic state contains numerous vibrational levels, which in turn contain rotational levels. Often, these rotational degrees of freedom are ignored as their energy scale is negligible in the context of electronic energies. A description of these vibrational states can be derived from first assuming the Born-Oppenheimer approximation (electronic and nuclear degrees of freedom can be separated).

$$|\Psi_{total}\rangle = |\Psi_{elec}\rangle \otimes |\Psi_{nuc}\rangle \tag{1.4}$$

The Franck-Condon Principle then assumes nuclear motion is slow compared to electron motion and therefore photoexcitations are instantaneous relative to nuclear motion. Thus, a transition probability can be calculated:

$$P_{f\leftarrow i} = \left|\langle\Psi^f_{total}|\,\hat{\mu}\,|\Psi^i_{total}\rangle\right|^2 = \left|\langle\Psi^f_{nuc}|\Psi^i_{nuc}\rangle\right|^2 \left|\langle\Psi^f_{elec}|\,\hat{\mu}\,|\Psi^i_{elec}\rangle\right|^2 \tag{1.5}$$

where $\hat{\mu}$ is the dipole operator. The probability has dependence on the nuclear wavefunction and only with significant overlap can the transition occur.

Figure 1.4 plots a pair of displaced harmonic oscillators. In this example, the ground state vibrational mode has overlap with the fourth vibrational overtone of the excited state. The spectral amplitude for electronic transitions depends on this amplitude of the overlap. Note, that between the two electronic states, the energy minima are at two different normal mode coordinates. The final state resulting from the electronic transition begins at a nonequilibrium geometry with some vibrational energy, hence, the portmanteau "vibronic".

Following electronic excitation, the system can then vibrationally relax. When the wavefunction of a lower vibrational state has significant overlap with another in the electronic ground state, the system can emit a photon and electronically relax. The outgoing photon is of less energy than the initial excitation input energy (fluorescence). However, the system need not always electronically relax with photoemission. If there is significant wavefunction overlap with a different excited electronic state, the system can relax nonradiatively (Figure 1.2). The new excited state will likely have a different equilibrium nuclear configuration and electronic wavefunction distribution such that the electron density may be localized in a different spatial region (charge transfer), as discussed previously. This nonradiative transition is very much in line with the work of Marcus and an accurate description faces the same challenges. Where two electronic states coincide, electronic coupling becomes large and

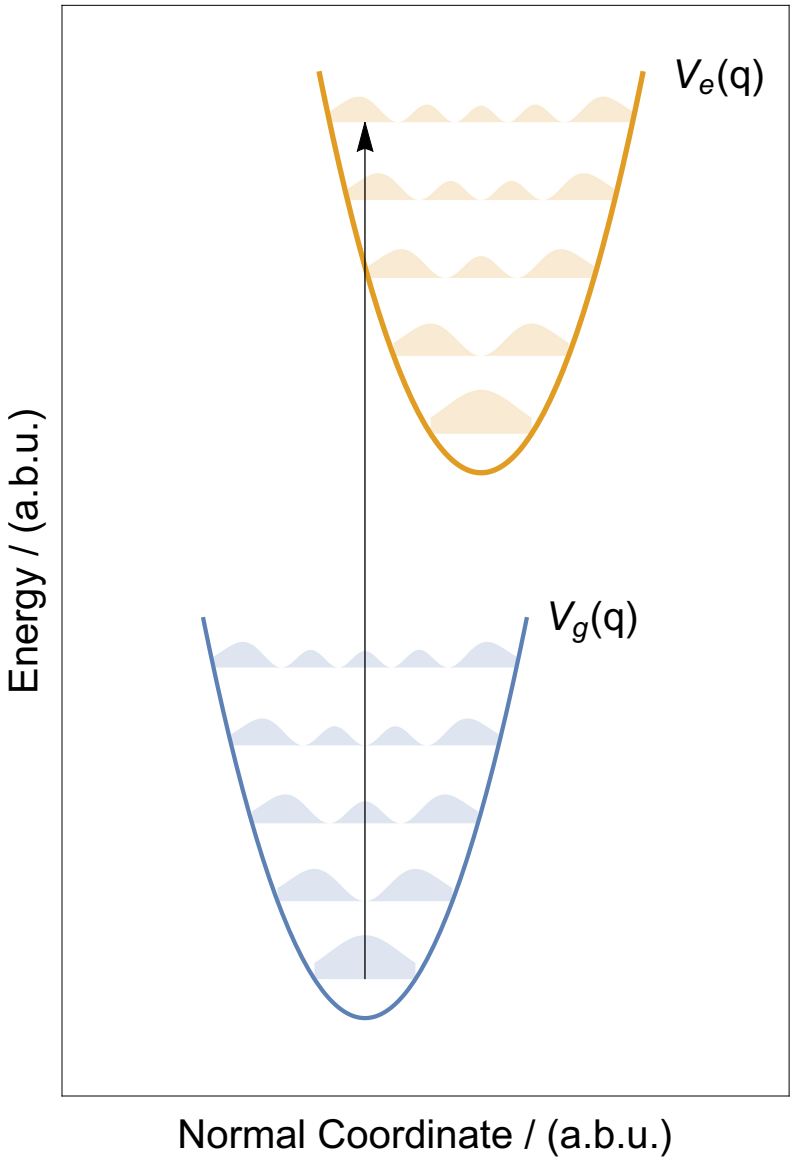

Figure 1.4: Schematic diagram of a vertical transition between a ground state (blue) and excited state (orange). Parabolic curves represent the potential energy surface of the respective electronic states. Strata within the parabolic curves are the amplitudes of the vibrational wavefunctions.

the Born-Oppenheimer approximation breaks down.

The Fewest Switches Surface Hopping (FSSH) method approaches electronic transitions from a dynamics perspective[10]. Within this method, multiple adiabatic states are propagated via the time-dependent Shrödigner equation and the system is in exactly one of these states at any given time. The system can hop to another adiabatic state with a probability computed from the electronic coupling between its current state and some other adiabatic state. An ensemble of propagated trajectories averages out the random aspect of "choosing" to hop against a probability. Adjusting nuclear velocities following a hop conserves angular momenta and total energy. Nuclei are purely classical, responding to mean-field potential (Ehrenfest[11] dynamics), and therefore no vibrational wavefunction overlap is considered for the purposes of electronic transitions. The original proposition of this method accurately described a collision reaction between H^+ and D_2[12]. Further developments apply more rigorous considerations, namely towards the ad hoc nuclear momentum rescaling and lack of quantum decoherence treatment[13, 14, 15, 16, 17]. FSSH and related methods have been applied to model electronic transitions in larger systems reproducing experimental results and, in some cases, with better accuracy than exact quantum mechanical calculations[18, 19]. FSSH is not entirely without faults, however. Large coupling value relative to the energy difference between two electronic states drives up the switching probability. In systems where low-lying states are densely packed, trajectories will rapidly hop about these surfaces, leading to unphysical results.

1.2 Charge Transfer in Organic Semiconductors

As the looming threat of global warming and climate change begins to manifest, the need to undo global energy dependence on fossil fuels and non-renewable energy sources becomes increasingly apparent. Solar energy is a viable alternative as there is enough radiation to power human energy consumption many-fold[20]. It becomes a crucial task, then, to understand the optoelectronic properties of photoactive materials and optimize their construction. In particular, comprehensive knowledge of electron/energy transfer in such materials lays a strong foundation towards this end. As such, the charge transfer problem has been the subject of an enormous amount of effort in well over the last half-century, including the works of many Nobel laureates.

Figure 1.5 shows the progress towards high photoconversion efficiency for various kinds of photovoltaic materials. The current record holders are multi-junction inorganic photovoltaics that now boast up to 46% photoconversion efficiency. These materials, however, often require caustic components and are costly to produce. On the lower side of the efficiency chart, the organic semiconductors have enjoyed two decades of growth, reaching just over 16% efficiency[21]. Despite this difference in efficacy, organic photovoltaics (OPV's) have many desirable properties. Namely, they are significantly cheaper and easier to produce, and the amorphous nature of the active photovoltaic layer lends to flexible devices suitable for applications on even non-rigid surfaces. Furthermore, vast knowledge of synthetic organic chemistry affords polymer scientists an enormous variation in cell construction. The efficiency of OPV's relies on the successful transfer of photoexcited electrons across the active layer. Complex morphology of most OPV's and their low dielectric constants make this a non-trivial phenomenon.

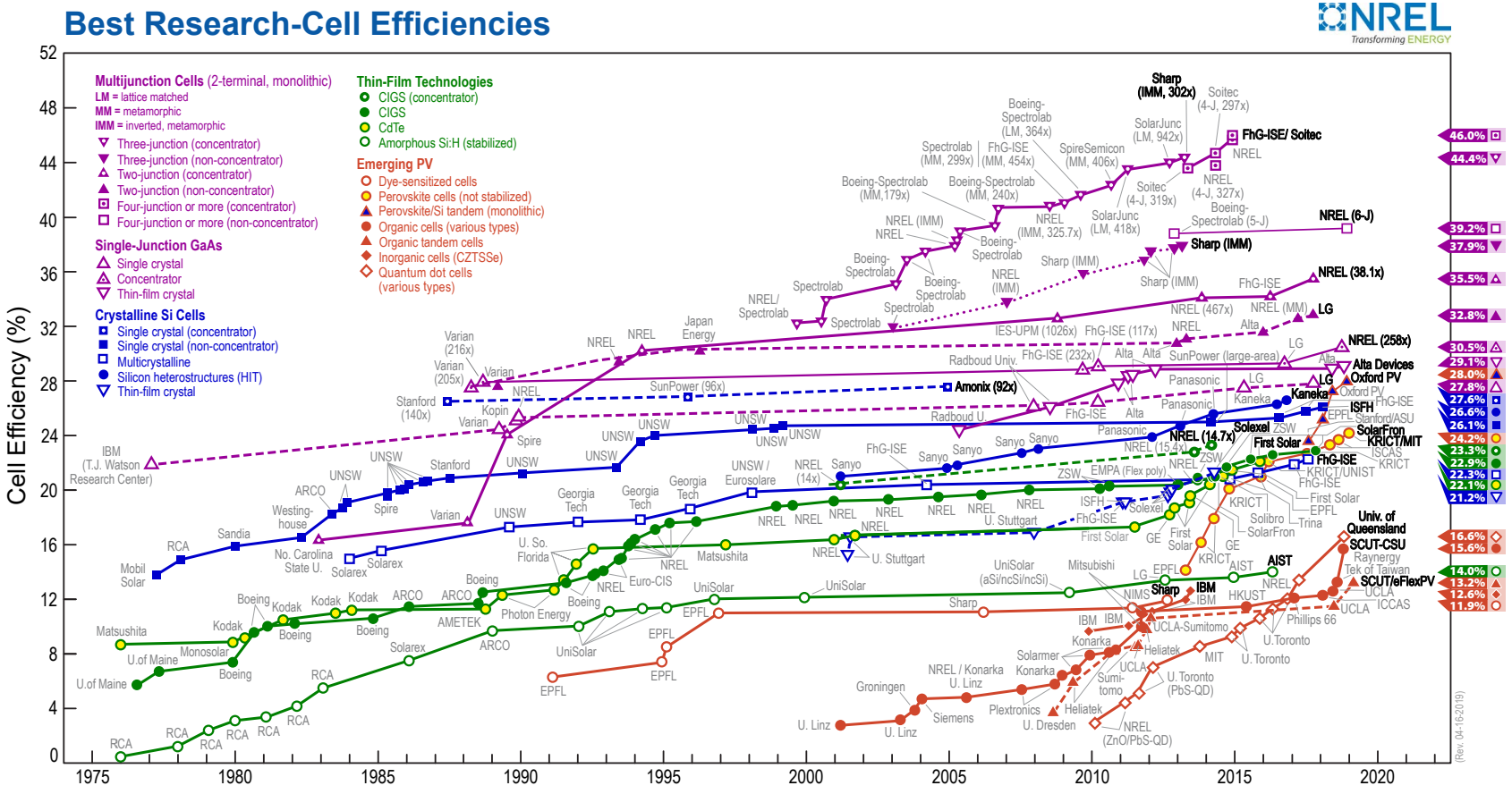

Figure 1.5: Photovoltaic efficiencies by material class in time collected by National Renewable Energy Laboratory. Figure current as of June 2019. Image courtesy of NREL[22].

1.2.1 The Bulk Heterojunction

Figure 1.6 depicts the cross-section of an OPV cell. At the top sits a glass layer which allows light to travel in. Sandwiched between two electrode layers (gray) is the active layer consisting of separated donor and acceptor phases. In an OPV of this type, a photon will excite an electron in the donor or acceptor phase generally forming a Frenkel exciton (a tightly bound electron and hole pair on the same molecule) or charge transfer exciton (electron and hole are on different molecules, yet Coulombically bound). This exciton will either undergo recombination (radiative for singlet excitons, nonradiative for triplet) back to the ground state or dissociate into a loosely bound electron-hole pair wherein the electron finds itself in the acceptor phase and its positively charged vacancy in the donor phase. These separated charges eventually migrate to opposite electrodes generating current. Binding energies for these excitons range between $\sim$ 0–1 eV[23]. This interaction is particularly strong relative to inorganic semiconductors as a result of the low dielectric constant of organic media, making the separation process all the more difficult. This binding energy must be overcome by the combination of ambient thermal energy and energetic offset of the acceptor and donor materials.

Characterizing these electron dynamics provides insight into the photophysical properties of the semiconducting material and may lead to the design of improved or novel materials. However, it is exceptionally difficult to atomistically simulate the optoelectronic properties of a full-scale bulk heterojunction given that modern quantum computational techniques increase exponentially with system size[1]. The energetic offset between acceptor and donor materials constitutes the driving force for charge

[1]Hartree-Fock (HF) treatments scale as $O(n^4)$ and Density Functional Theory (DFT) as $O(n^3)$ for n basis functions.

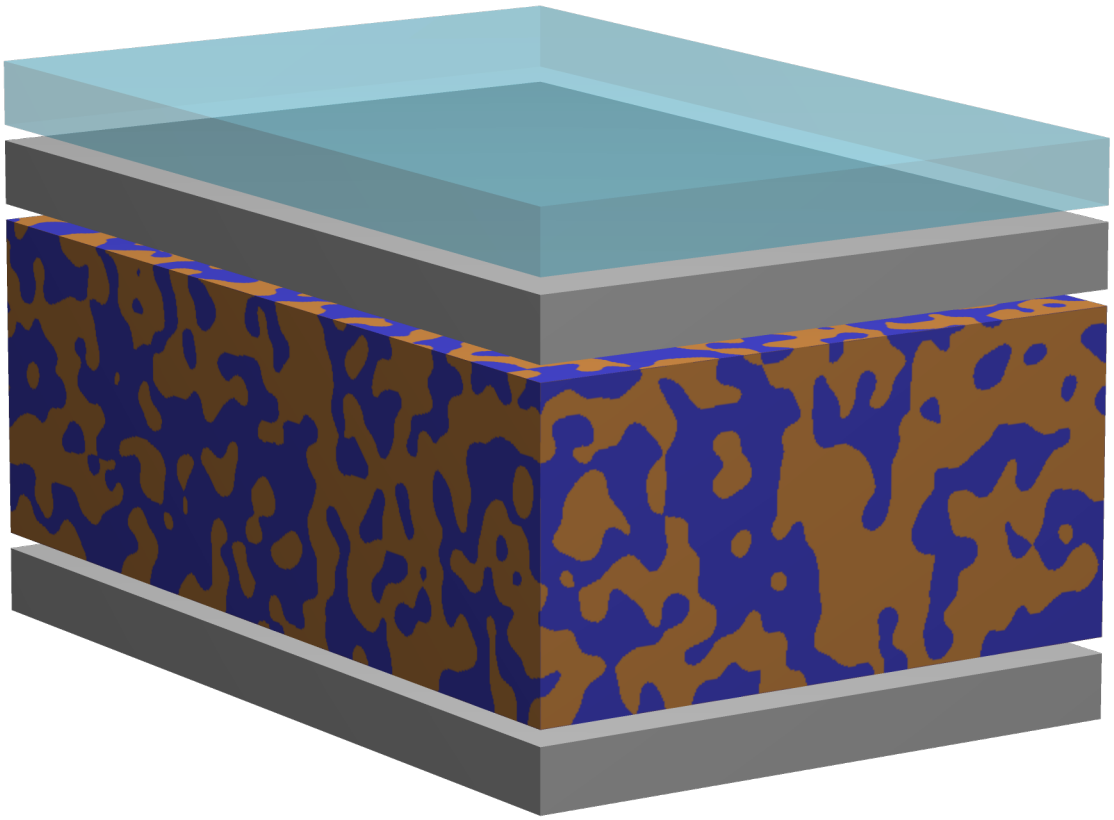

Figure 1.6: Depiction of an organic photovoltaic device. Blue and orange represent donor and acceptor phases of the bulk heterojunction.

separation and, therefore, one can simplify the problem by studying interactions at the interface between semiconductor phases.

Previous work by Kelley, Patel, and Bittner used a mixed quantum mechanical/-molecular mechanics approach model a PPV:PCMB bulk heterojunction in which preselected interfacial molecules were determined to be "quantum active" leaving the remaining molecules treated as a classical solvent[24]. By "quantum active", we mean π electrons of this subset of molecules are treated quantum mechanically and the rest treated with force fields. Our simulations describe that low lying excited states form a dense band spanning about 0.5 eV; as previously discussed, FSSH under this scenario undergoes rapid switching between propagated surfaces producing unphysical results. We show, via Fourier Transform of bandgap time-series, that excitation energy is modulated by lower energy torsional modes, mid-range C=C bond stretching, and, to a lesser extent, high energy C-H modes. Such vibrations, though perhaps only weakly coupled to electronic states, provide enough energetic overlap between electronic states such that rapid transitions occur between excitonic, charge transfer, and (long-range) charge-separated state.

In general, an excitation can occur in any molecular phase of the system. Photoexcitation of the donor requires subsequent migration of the excited electrons to the acceptor along LUMO orbitals to form the CT state. In contrast, photoexcitation of the acceptor should be followed by hole migration to the donor through HOMO orbitals. In certain model systems, charge transfer can be mediated by a bridge moiety. Large energetic offset decreases overlap between donor and acceptor states. However, bridge states provide both intermediate energetic and spatial pathways for much more efficient electron transfer. Distortions arising from vibrations, solvent effects, or other

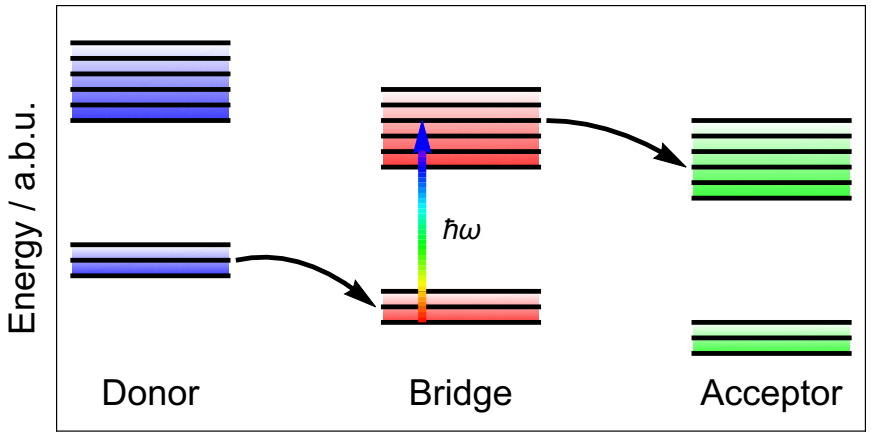

Figure 1.7: Schematic representation of orbital energies for each moiety. Initial excitation is represented as a promotion of an electron from bridge HOMO to LUMO. Dark arrows represent subsequent electron transfer processes resulting in the final charge transfer state. Vibrational strata provide accessible higher energy states the energetic proximity helps facilitates electronic transitions.

noise fluctuate the energy differences between these states and alter the charge transfer rate as a consequence. Figure 1.7 depicts an alternative sequence of the charge transfer mechanism wherein a bridge electron is excited, followed by two localized transitions: bridge to acceptor along LUMO's and donor to bridge along HOMO's. We discussed such models in the proceeding subsection.

1.2.2 Artificial Photosynthesis

Additionally, one can look to nature for inspiration for another solar energy conversion method. For billions of years, phototrophic lifeforms have used light as the primary source of energy to drive anabolic processes. Photosynthesis is initiated by photoabsorption in the light-harvesting complex (LHC). Figure 1.8 depicts the structure of light-harvesting complex I (LH-I) and the reaction center (RC) of purple bacteria[25]. Within LH-I, 32 bacteriochlorophyll (BChl, green square with yellow carotenoid tail) absorb light and funnel the excitation energy to the RC encompassed within.

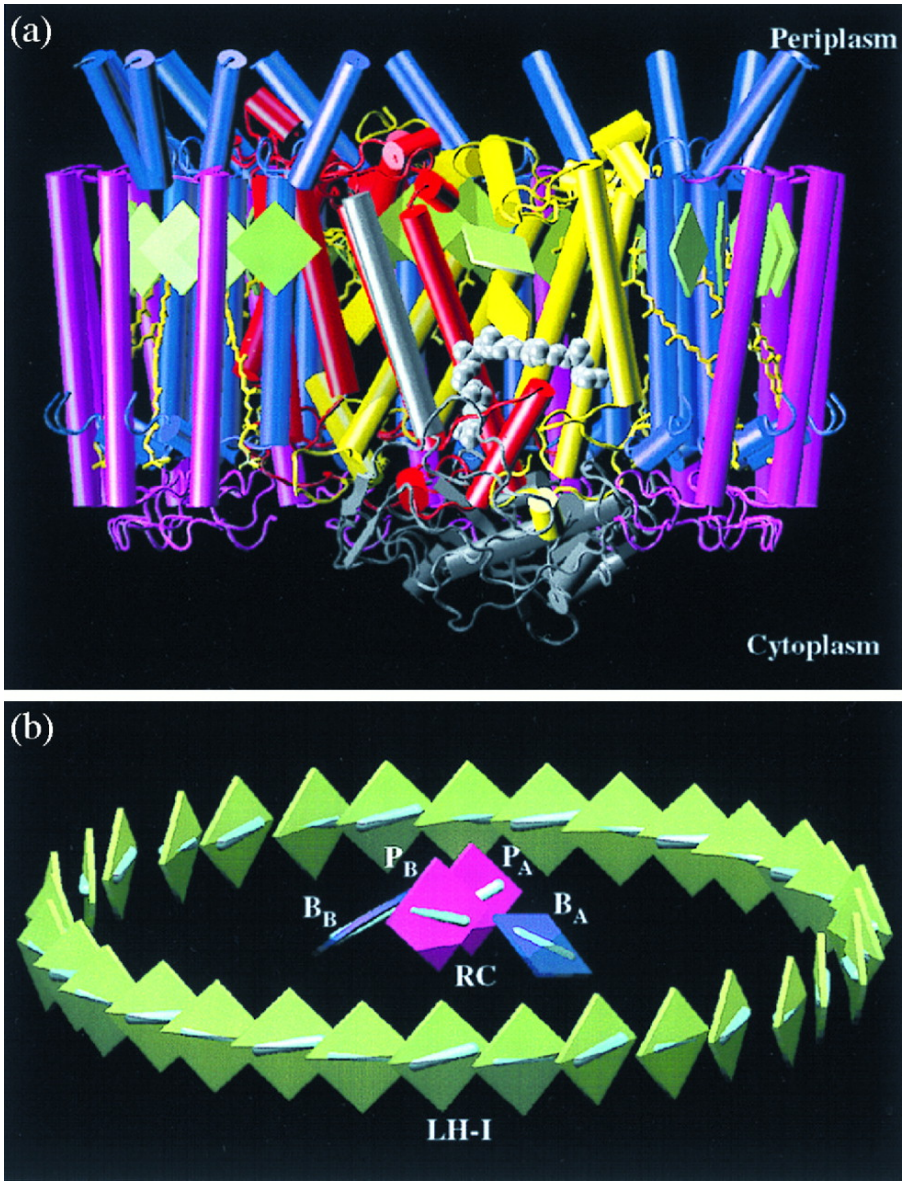

Figure 1.8: Structure of the LH-I–RC complex. (a) Side view of the LH-I–RC complex with three LH-I $\alpha\beta$-heterodimers on the front side removed to expose the RC in the interior. The α-helices are represented as cylinders with the L, M, and H subunits of the RC in yellow, red, and gray, and the α-apoprotein and the β-apoprotein of the LH-I in blue and magenta. BChls and bacteriopheophytins are represented as green and yellow squares, respectively. Carotenoids (spheroidenes) in a yellow licorice representation, and quinone Q_B is rendered by gray van der Waals spheres. Q_B shuttles in and out (as Q_BH_2) of the LH-I–RC complex. (b) Arrangement of BChls in the LH-I–RC complex. The BChls are represented as squares with B875 BChls of LH-I in green, and the special pair (P_A and P_B) and the accessory BChls (B_A and B_B) of the RC in red and blue, respectively; cyan bars represent the Q_y transition moments of BChls.[25] Image courtesy of National Academy of Sciences. Copyright (1998) National Academy of Sciences, U.S.A.

However, this complex does not sufficiently collect enough light alone and requires antennae in the form of light-harvesting complex II (LH-II) and, in some bacteria, light-harvesting complex III (LH-III)[26, 27, 28, 29]. LH-II and LH-III are structurally similar to LH-I but contain less BChl units and have no reaction center. Each LH-I complex can be surrounded by up to ten LH-II antennae with the exact ratio depending on the bacteria itself and growth conditions[30]. Absorption studies of these LH complexes have found that LH-III absorbs at higher energies than LH-II, which in turn absorbs at higher energy than LH-I. The succession of energies drives photoexcitation energy from LH-III through LH-II to LH-I[31] and finally ending at a pair of strongly interacting BChls (red squares in Figure 1.8(b)) in the RC. This final pair, termed the special pair, drives the high energy electron into quinone acceptors. In purple bacteria, the system recovers the lost electron from the oxidation of H_2, elemental sulfur, or sulfur compounds, completing the electron processes in Photosystem II. In cyanobacteria and higher plants, the process concludes with the oxidation and splitting of water.

Scientists have long since attempted to understand and replicate the photophysical processes occurring within photosynthetic bacteria. Efficient splitting of water provides a viable means of hydrogen production for energy storage. Three key factors have been identified for developing artificial reaction centers. Systems must (1) have quantum yield near unity, (2) have slow recombination rates, and (3) store large amounts of energy[32].

The final electron transfer reaction in Photosystem II (special pair to quinone transfer) precedes the water-splitting step and is therefore an important precursor process to study. The photosynthetic complex fixes the special pair into optimal positions for electron transfer to bound quinones. For experimental study, porphyrin

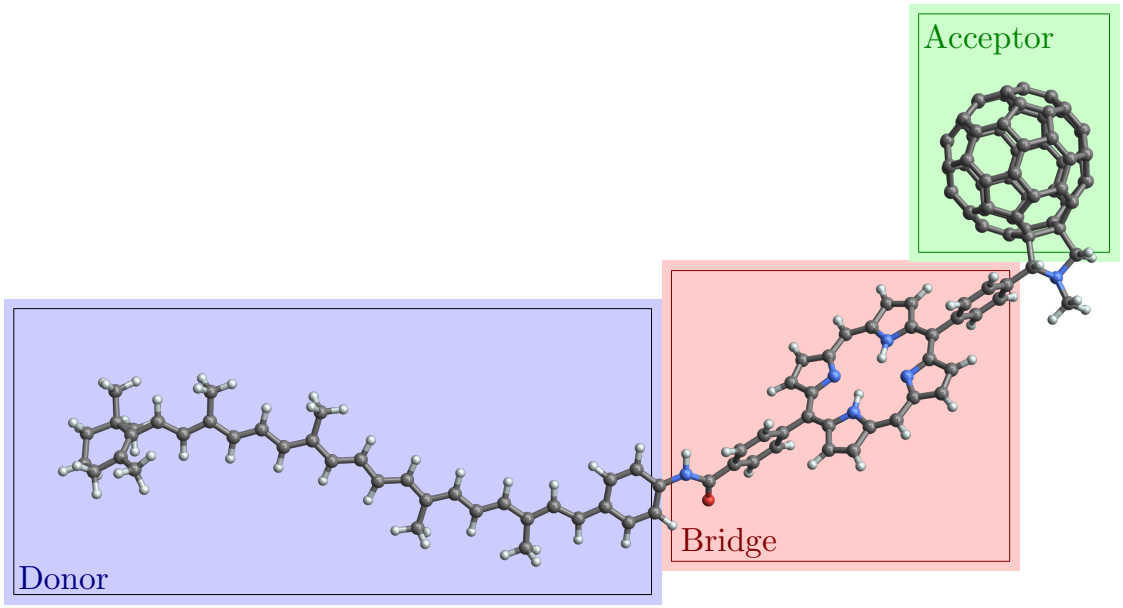

Figure 1.9: Basic construction of the Donor-Bridge-Acceptor triad composed of a carotenoid donor (blue), porphyrin bridge (red), and fullerene (green) derivative acceptor (green). Alternative structures include addition of side groups to the porphyrin in β or *meso* positions, increasing distance between subunits via methylene units, or a different fullerene derivative.

molecules are photophysically similar to Bchl, but have the added benefit of facile synthesis and stability. Thus, early models considered covalently bound porphyrin and quinone[33, 34].

Tom and Anna Moore and coworkers have experimentally investigated a wide array of dyads (donor/acceptor) and triads (donor/bridge/acceptor) that mimic the photosynthetic processes of bacterial light harvesters[35, 36, 37, 38, 39, 40, 41, 42, 43]. They have shown that porphyrin-quinone and porphyrin-C_{60} dyad systems can maintain charge separation for time scales up to hundreds of picoseconds[37, 40]. The addition of a carotenoid polyene moiety increases lifetimes, in some cases by three orders of magnitude, by increasing the physical separation of the charges. A readily detectable transient absorption band of the carotenoid radical cation[44] and large detected dipole moment[42] support this claim.

Triadic systems of this design undergo a multistep electron transfer process (depicted in Figure 1.10). First, the porphyrin bridge, retaining its chromophoric function from BChls and the dyads, is photoexcited into a singlet excited state. Following this, the excited electron migrates to the acceptor moiety (k_2), driven by the energetic offset between the exciton and first charge transfer state ($CP^+C_{60}^-$). An electron from the donor then relaxes (k_4) into the vacancy in the bridge forming the final CT state. This state can undergo a decay process into the carotenoid triplet state[45] (k_6) and finally relax down to the ground state. Incidentally, this triplet state, which provides photoprotection from singlet oxygen, is not seen in many other biomimetic materials but is seen in natural photosynthetic systems[40].

Each charge transfer process competes with a corresponding recombination process (not shown) that relaxes the system to the ground state. By comparing the rate constants (k_x) of each forward process to other competing relaxation processes, one can ascribe a quantum yield value to each step

$$\Phi = \frac{k_{\text{forward}}}{k_{\text{forward}} + k_{\text{others}}} \tag{1.6}$$

Large k_{forward} values (faster process) compared to competing processes push Φ towards unity. Experimental studies of the formation of first CT state in Carotenoid-Porphyrin-Fullerene (CPF) triads indicate the quantum yield for this process lies around $\Phi > 0.85$).

Rozzi and corkers have combined experimental and theoretical techniques on a study of one CPF triad[46]. Their experimental work studies the triad in Figure 1.9 with the addition of mesitylene molecules at the porphyrin *meso* positions. Their theoretical work considers only the basic construction of the triad. In this work, they claimed coherent ultrafast charge transfer occurs at the sub-100 femtosecond scale. Differential transmission spectra showed the rise of a few bands following

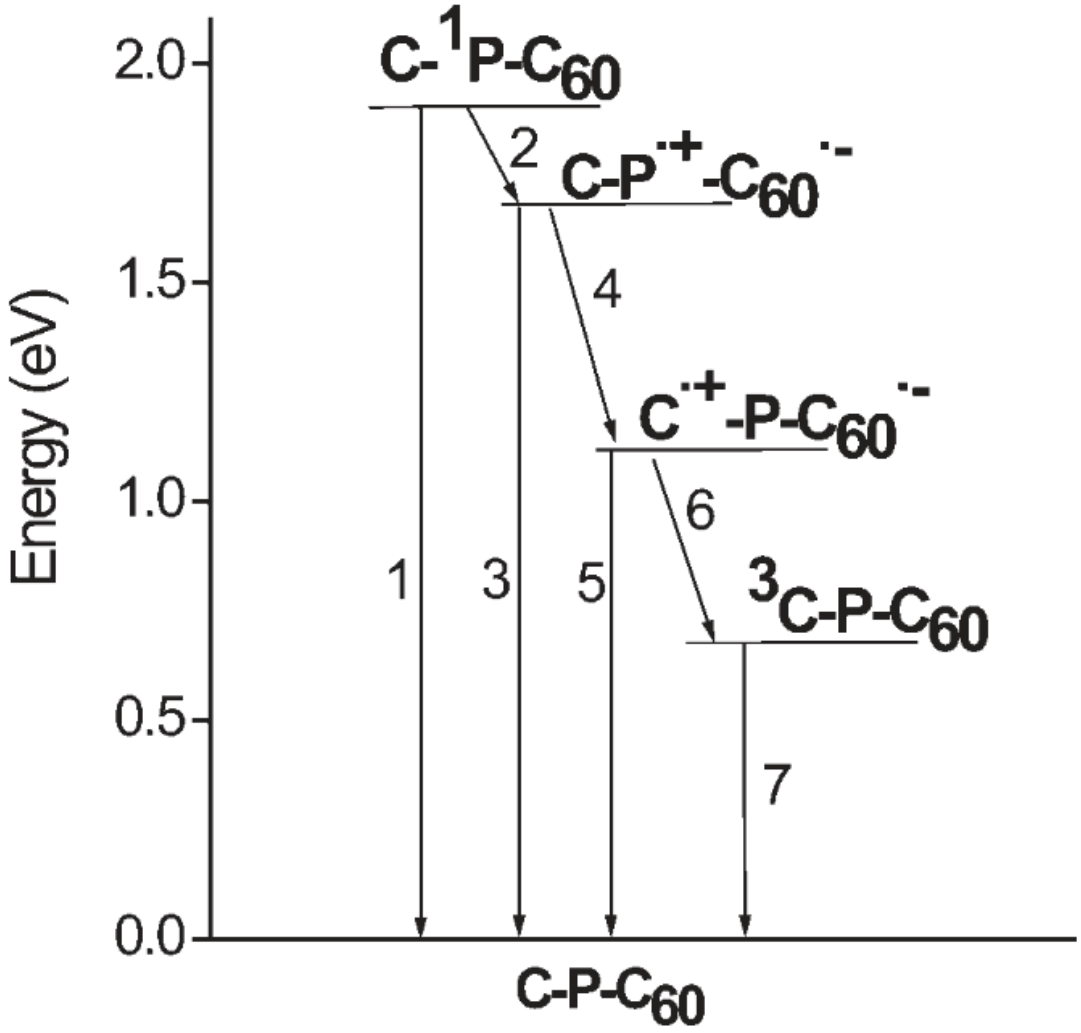

Figure 1.10: Energies of transient states for **M1**. Energies levels will vary with triads **M2-4** but are relatively similar. Arrows indicate electronic transitions with even numbers labeling charge transfer processes and odd numbered rates labeling relaxation processes. Republished with permission from Wiley and Sons.

system photoexcitation in the neighborhood of 550 nm. Each of these bands exhibited oscillatory behavior in intensity with a period of about 30 fs.

Their theoretical study involved a Time-Dependent Density Functional Theory (TDDFT)[47] excited state dynamics simulation of a similarly structured triad. By spatially integrating the electron density around the fullerene moiety, they measured the charging of the acceptor in time. A full electron of charge migrated to the fullerene with the charging shape also having some oscillatory behavior (roughly 30 fs period). As this is close to the conjugated carbon-carbon bond stretching period, they concluded that such nuclear vibrations are fundamental to CT. They performed additional TDDFT simulations freezing various subsets of nuclei. Each of these dynamics simulations showed mitigated charge transfer at some level and freezing all nuclei suppressed CT entirely.

Note, Ref. [46] only considered the results of a *single trajectory* which may not represent the actual CT dynamics within these triads. Furthermore, the molecule presented in the supplementary media of simulations does not reflect the molecule presented for their experimental work. Differences include lack of pyrolle methyl group on fullerene derivative (which extends the π conjugation towards acceptor), a reversal of the amide linker, and *para* versus *ortho* connectivity on an aryl linker, all of which have known effects on electronic current and charge transfer[48, 49].

Cheung *et. al.* have approached computation of CT rates of the basic CPF triad (Figure 1.9) via the Fermi Golden Rule expression[50]. They applied the linearized semiclassical (LSC) approximation, developed by Geva[51], which has been shown to reproduce rate constants so long as the donor and acceptor potential energy surfaces are sufficiently parabolic and otherwise identical save for some difference in equilibrium coordinate and energy. The FGR expression reduces to a form reminiscent of

the Marcus expression via second-order cumulant approximation. Here, the various parameters in the Marcus expression can be related to the dynamical statistics of the donor-acceptor energy bandgap. With this approach and parameterization from molecular dynamics simulations with explicit solvent, their results indicate the formation of the first CT state at the picosecond scale and second state at far longer times with little difference resulting from rigid or flexible triad nuclei. They further identify a fundamental amide mode (1700 cm^{-1}, C=O stretching) that is highly sensitive to the electronic state and triad conformation. This mode exhibits a blue shift of about 25 cm^{-1} between the π-π^* and first CT states. Despite this sensitivity, there seems no imminent correlation between the mode position and the local electric field. Thereby, this mode is more likely to be affected by changes in the conjugations over other effects. With the dramatic shift in frequency, they propose that this mode can be probed to visualize or monitor the CT process.

1.2.3 Charge Transfer Rate Dependence on Molecular Construction

Charge transfer rates depend on the molecular system. It is surprising, however, just how sensitive they are to seemingly minor changes in moiety linkers and even side substituent groups. For example, in triads with donor-bridge units linked by an amide group, reversal of the amide direction changes the rate of formation of the final CT state by a factor of ~ 30[37]. A number of studies on the rectification behavior of small molecules and linkers substantiate this directional bias[52, 48, 53] and primarily attribute this effect to asymmetry in electronic structure.

Choice of side groups on the bridge can affect the electronic coupling between

the bridge and the acceptor. Several studies of CPF triads consider the porphyrin bridge linked to fullerene acceptor via an aryl group. Fluctuations of the dihedral angle between this linker and porphyrin not only modulate the electronic coupling to the acceptor but also perhaps affect the energy difference between excited and charge transfer states. Large aliphatic side groups in the β positions sterically hinder the accessible angles the linker can sample. Small dihedral angles increase the p_z orbital overlap where angles closer to $\pi/2$ close the overlap. Therefore bridges with smaller side groups should allow better electronic overlap and result in faster CT rates. However experimental measurements prove otherwise: triads with aliphatic groups proved an order of magnitude faster[39, 43]. The absorption spectrum of the triads is remarkably close to a linear combination of spectra of the individual moieties [39, 46] indicating that when covalently linked, the moieties are nearly electronically independent; covalent bonding in the triad form only weakly perturbs the electronic structure of each moiety. Bahr and coworkers argue, in the framework of Marcus theory, that the thermodynamic driving force, ΔG^0, has a much more significant effect on the CT rate than the electronic coupling.

1.3 Dissertation Overview

We are interested in identifying the principle vibronic motions that couple and modulate electron transfer processes in a class of molecules that model photosynthetic reaction centers in biological systems. In agreement with the literature, the present work confirms the crucial role vibrational dynamics play in the process of electron transfer in Donor-Bridge-Acceptor triads. Specifically, we identify particular vibrational modes of various Carotenoid-Porphyrin-Fullerene (CPF) systems that strongly

correlate to charge transfer following photoexcitation. Our investigations employ a density matrix formalism in the time-dependent Hartree-Fock (TDHF) schema in conjunction with molecular dynamics simulations to analyze vibrational modes of the system and assess the dependence of charge transfer on the smallest details of molecular structure.

This dissertation is structured in the following way. Chapter 2 details the derivation of the methodology with which we perform our investigations, including the density matrix formalism and its connection to TDHF through the Liouville-von Neumann equation. Further, we explain structure preparation and simulation setup. Chapter 3 presents, first, the results of the electron dynamics and charge transfer rate approximation and, second, an in-depth vibrational mode analysis of the CPF triads and detailed discussions of what charge-transfer depends on in these systems. We conclude by outlining immediate areas of improvement as well as speculations on the possible directions this project can take. The reader can find in Appendix A a multitude of `Fortran` and `Python` codes used in the development and analysis of this work.

Chapter 2

Methods

2.1 Dynamical Simulations

In highly conjugated organic systems, the electronic properties of interest primarily happen within the closely packed π and π^* molecular orbitals. We suspect that treating π-electrons quantum mechanically should sufficiently reproduce the charge transfer dynamics while significantly reducing computational cost. Therefore we assume negligible orbital overlap between σ- and π-states and separate their degrees of freedom. The semiempirical Pariser-Parr-Pople (PPP)[54, 55] model readily reflects this paradigm and is implemented in the TINKER software[56].

The TINKER code employs a modified version of the PPP Hamiltonian wherein a specified number of atoms (sites), predesignated in the input, contribute a p_z orbital and an appropriate number of electrons to the quantum system depending on expected atomic orbital hybridization. These site localized orbitals form the basis set for the quantum calculations and π molecular orbitals. All other electronic interactions (σ-system) are then treated with molecular mechanics (force fields). The ground state

is calculated via self-consistent field (SCF) iterations of the restricted Hartree-Fock (HF) equations

$$\mathcal{F}_{ij} = H_{ij} + \sum_{k}^{N/2} \sum_{lm}^{N} c_{lk} c_{mk}^{*} \left[2\left(ij|ml\right) - \left(il|mj\right) \right] \tag{2.1}$$

H_{ij} is the Hartree-core (one particle) term which encapsulates the electron's kinetic energy and Coulombic interaction with the nuclei (in atomic units):

$$H_{ij} = \langle i| - \frac{1}{2}\nabla^2 - \sum_{A} \frac{Z_A}{r_{1A}} |j\rangle \tag{2.2}$$

where A iterates through the nuclei. The second term is the two particle matrix and corresponds to Coulomb and exchange terms[57]. Indices $i, j, l,$ and m iterate through the basis (site) orbitals centered on the nuclei; the matrix element c_{lk} is the contribution of site orbital l to molecular orbital k (the eigenvector matrix). The form of Equation 2.1 can be simplified with two steps. First, define a density matrix in the usual sense under closed-shell restricted (all electrons are spin paired and molecular orbitals are doubly occupied) Hartree Fock

$$\rho_{ij} = \sum_{k}^{N/2} \sum_{lm} 2 c_{lk} c_{mk}^{*} \tag{2.3}$$

Here, the coefficient of 2 denotes doubly occupied molecular orbital k. We generalize this expression as

$$\rho_{ij} = \sum_{k}^{N} \sum_{lm} \eta_k c_{lk} c_{mk}^{*} \tag{2.4}$$

where η_k is some population count (2 for closed-shell, 1 for open-shell). In this form, the density matrix is simply a unitary transform of the state occupations (a diagonal matrix) into site basis.

$$\rho_{site} = C \cdot \rho_{state} \cdot C^{\dagger} \tag{2.5}$$

Here, the matrix C is the modal matrix (molecular orbital) of the Fock matrix. Second, apply Complete Neglect of Differential Overlap (CNDO) approximation to

the two-particle integrals.

$$\begin{aligned}(ij|kl) &= \int d\mathbf{r}_1 d\mathbf{r}_2 \chi_i^*(\mathbf{r}_1)\chi_j(\mathbf{r}_1) r_{12}^{-1} \chi_l^*(\mathbf{r}_2)\chi_l(\mathbf{r}_2) \\ &= J_{ik}\delta_{ij}\delta_{kl}\end{aligned} \tag{2.6}$$

(for atomic orbitals χ). This reduces the HF equations to

$$\mathcal{F}_{ij} = H_{ij} + \sum_k^{N/2} \rho_{ij} J_{im} \left[\delta_{ij}\delta_{ml} - \frac{1}{2}\delta_{il}\delta_{mj}\right] \tag{2.7}$$

In this expression, terms H_{ij} and J_{im} are approximated using semi-empirical parameters and are also dependent on the geometry of the system.

2.1.1 Nuclear-Electron Coupling

Electronic degrees of freedom are coupled to the molecular dynamics calculation by adjusting equilibrium bond lengths, bond force constants, and torsional barriers as a function of the site representation density matrix. The density matrix here is interpreted as a *bond-charge matrix* where diagonal elements represent site populations, and off-diagonal terms represent the π-bond order between two sites. The equilibrium bond length is calculated using a linear interpolation between idealized single and double bond lengths. Thus, for bonded atoms i and j, the equilibrium bond length is

$$l = l_1 + \delta l(1 - \rho_{ij}) \tag{2.8}$$

where l_1 is the idealized bond length for the double bond $\rho_{ij} = 1$. The bond force constant is similarly interpolated

$$k = k_1 - \delta k(1 - \rho_{ij}) \tag{2.9}$$

as is the torsional barrier

$$t = \rho_{ij} t_1 \tag{2.10}$$

In general off-diagonal elements of the density matrix have non-zero imaginary value, but the matrix must remain Hermitian ($\rho_{ij} = \rho^*_{ji}$). For molecular mechanics parameters, we drop the imaginary value and use only the real part. It is clear to see that increasing the bond order (π-bond character), the equilibrium length decreases while bond force constant and torsional barrier increases.

At each time step, SCF iterations recalculate the ground state Fock matrix, generating a new density matrix, and the above parameters are adjusted accordingly. Nuclear integration follows the new forces and the process repeats. In the context of harmonic oscillators, the vibrational energy of a normal mode depends on the mode's reduced mass and the force constant. In units of wavenumbers (cm^{-1}):

$$\tilde{\nu} = \frac{1}{2\pi c}\sqrt{\frac{k}{\mu}} \tag{2.11}$$

where c is the speed of light, μ is the reduced mass, and k is the force constant taking into account the above computation. As a system undergoes thermal fluctuation, the electronic structure, and therefore elements of the density matrix, should not fluctuate wildly. However, an impulsive excitation causes significant deviation from the ground state and the collective changes in these force constants will change the overall dynamical behavior of the nuclei. In short, as the electronic state undergoes any form of transition, there will be a corresponding change in the vibrational modes.

2.1.2 Excited States from Configuration Interaction-Singles

Our first modification to this framework comes in the form of implementing an electronically excited state using Configuration Interaction-Singles (CI-S). In general, the CI method uses the HF states as input and generates a wavefunction composed of a

large number of elementary excitation states.

$$
\begin{aligned}
|\Phi\rangle = \\
& C_0 |\Psi_0\rangle \quad + \sum_{ar} C_a^r |\Psi_a^r\rangle \quad + \sum_{\substack{a<b \\ r<s}} C_{ab}^{rs} |\Psi_{ab}^{rs}\rangle \quad + \sum_{\substack{a<b<c \\ r<s<t}} C_{abc}^{rst} |\Psi_{abc}^{rst}\rangle \quad + \quad \dots \\
& \text{Ground State} \qquad \text{Singles} \qquad \text{Doubles} \qquad \text{Triples} \qquad \dots
\end{aligned}
\tag{2.12}
$$

In this notation, coefficients C represent the contribution of the particular elementary excitation and $|\Psi_a^r\rangle$ implies moving an electron from occupied HF state a to unoccupied HF state r (b to s, c to t,...).

Even for a reasonable number of electrons, the full formulation quickly becomes intractable. We, therefore, dismiss double excitations and higher as well as truncate the input to the highest 20 occupied HF states and lowest 20 unoccupied HF states (amounting to 400 single excitations considered). Then, diagonalizing the CI Hamiltonian produces the excitation energies and excited states. We only consider singlet excitations and therefore restrict the possible excitations to the promotion of an α-spin electron to another α-orbital.

2.1.3 Density Matrix Formalism

For increasingly larger systems, repeating the SCF iterations at each time-step becomes computationally cumbersome (despite already ignoring non-π electrons). The same holds for diagonalizing the CI Hamiltonian. The next modification changes the method by which we propagate a system in time. We take a time-dependent Hartree-Fock (TDHF) approach choose to evolve the density matrix in time via the Liouville-von Neumann equation. We briefly revisit the derivation from Ref [58] here.

We define the time-dependent density matrix by the outer product:

$$\rho(t) = |\psi(t)\rangle \langle\psi(t)| \tag{2.13}$$

The time derivative of this expression gives:

$$i\hbar\frac{\partial}{\partial t}\rho(t) = \left(i\hbar\frac{\partial |\psi(t)\rangle}{\partial t}\right)\langle\psi(t)| + |\psi(t)\rangle\left(i\hbar\frac{\partial \langle\psi(t)|}{\partial t}\right) \tag{2.14}$$

We apply the time-dependent Schrödinger equation and its adjoint

$$\begin{aligned} i\hbar\tfrac{\partial}{\partial t}|\psi(t)\rangle &= \hat{\mathcal{H}}|\psi(t)\rangle \\ i\hbar\tfrac{\partial}{\partial t}\langle\psi(t)| &= -\langle\psi(t)|\hat{\mathcal{H}} \end{aligned} \tag{2.15}$$

to equation 2.14, giving:

$$\begin{aligned} i\hbar\tfrac{\partial}{\partial t}\rho(t) &= \hat{\mathcal{H}}|\psi(t)\rangle\langle\psi(t)| - |\psi(t)\rangle\langle\psi(t)|\hat{\mathcal{H}} \\ &= \left[\hat{\mathcal{H}}, \rho\right] \end{aligned} \tag{2.16}$$

The above equation is often compactly written as:

$$\frac{\partial}{\partial t}\rho(t) = -\frac{i}{\hbar}\mathcal{L}\rho(t) \tag{2.17}$$

where $\mathcal{L}$ is the Liouvillian super-operator that commutates the Hamiltonian and density matrices. In the time-dependent Hartree-Fock schema, we replace the Hamiltonian operator with the Fock operator

$$i\hbar\frac{\partial\rho}{\partial t} = [\mathcal{F}, \rho] = i\mathcal{L}_{\mathcal{F}}\rho \tag{2.18}$$

We choose to expand the time propagator as a series of imaginary Chebyshev polynomials:

$$\left(e^{-i\mathcal{L}dt}\right)\rho(t) = \left(\sum_{k=0}^{N} a_k T_k(-i\mathcal{L}dt/R)\right)\rho(t) \tag{2.19}$$

where T_k is the k^{th} Chebyshev polynomial of the first kind. The value

$$R = dt(E_{max} - E_{min})/2 \tag{2.20}$$

normalizes the super-operator eigenspectrum to range [-1,1] as this is the region where the Chebyshev polynomials are defined. The coefficient a_k is defined as such:

$$a_k = e^{i(R+G)} C_k J_k(R) \tag{2.21}$$

where $C_k = 1$ for $k = 1$ and $C_k = 2$ for $k > 1$. J_k are the Bessel functions of the first kind of order k.

We use the Chebyshev expansion of the exponential is generally considered to be stable and converges rapidly. More importantly, this expansion scheme is norm preserving[59, 60]: the total magnitude of ρ is conserved. This property ensures that the trace of the density matrix ($\mathrm{Tr}[\rho]$ = number of electrons) is also preserved. All of these traits are highly beneficial for maintaining accuracy and stability during long-time propagation. Finally, the recursive definition of the polynomials makes their computation trivial, even for a large number of expansion terms. For a normalized operator $\hat{X}$, the Chebychev polynomials are generated using the recursion relation

$$\begin{aligned} T_0(\hat{X}) &= \hat{1} \\ T_1(\hat{X}) &= \hat{X} \\ T_k(\hat{X}) &= 2\hat{X}T_{k-1}(\hat{X}) + T_{k-2}(\hat{X}) \end{aligned} \tag{2.22}$$

Within our implementation of this scheme, we truncated the recursion at 9th order giving a typical numerical error of $1 : 10^{-10}$ (unitless) which we consider sufficiently accurate.

The Fock matrix (Eq. 2.7) is calculated at each time-step based on the current nuclear geometry and the previous electronic density matrix.

$$\mathcal{F} = \mathcal{F}\left(\mathbf{R}(t + dt), \rho(t)\right) \tag{2.23}$$

The density matrix is propagated with the new Fock matrix, force field parameters are adjusted based on new bond orders, nuclei are iterated according to classical

mechanics, and the iteration process repeats. At time $t = 0$, the ground state SCF computation and CI excited states must still be computed; implementing the time evolution removes the need for subsequent diagonalizations. However, it is perhaps wise to occasionally diagonalize the Fock matrix and collect the eigenspectrum to ensure the Liouvillain remains within the limit where the Chebychev polynomials are well behaved.

2.1.4 Hartree Core Coupling Term

The implementation of the PPP Hamiltonian is a nearest neighbors model: for the calculation of the Hartree core term, only matrix elements of bonded π-active atoms are non-zero. Then for systems with multiple molecules (e.g., stacked benzene rings) or sp^3 hybridized atoms between π-systems (e.g., diphenylmethane), the matrix is block diagonal. Despite the two-particle matrix, J_{im}, being densely populated, the shape of H_{ij} dominates in the SCF procedure. As a result, the final Fock matrix is block diagonal leading to localized molecular orbitals, and therefore, a block diagonal density matrix, as shown in figure 2.1. What this implies is that there is no electronic communication between these separated systems and, during the time evolution described previously, no charge can migrate between these systems.

We remedy this by lifting the nearest neighbors restriction and allow interaction between non-bonded sites. However, the original formulation of the one-particle term is calculated by the Whitehead and Lo formula[61]:

$$H_{ij} = \frac{3}{2}\left(E_b - E_e\right) - \frac{3}{8}\gamma_{11} + \frac{5}{12}\gamma_{12} - \frac{1}{24}\gamma_{14} \tag{2.24}$$

where E_b (E_e) is the bond energy of carbon-carbon bond in benzene (ethylene) calculated using a semiempirically parameterized Morse potential, γ_{11} is the average of the

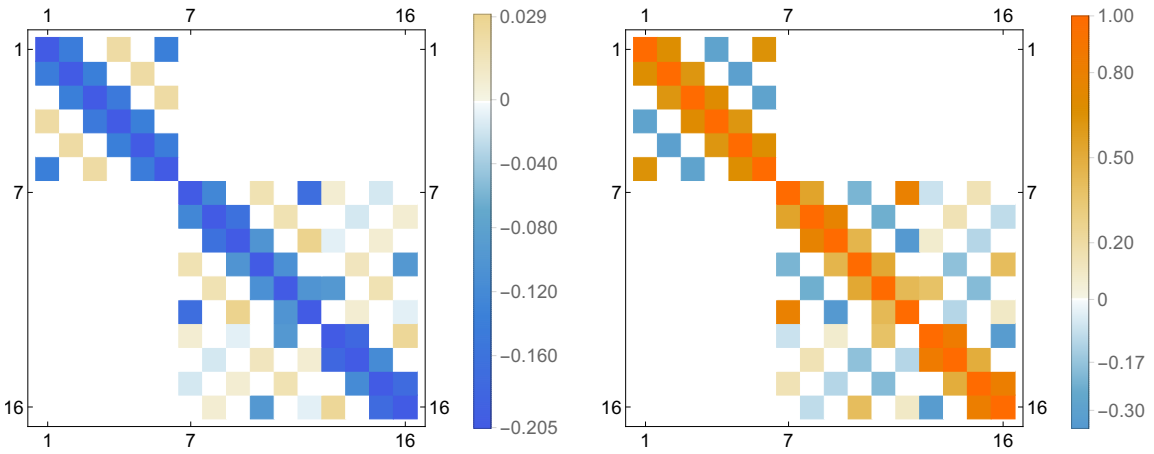

Figure 2.1: Matrix plots of the Fock (left, units in eV) and density (right, unitless) matrices for a ground state calculation of stacked benzene and naphthalene. Indices 1-6 correspond to benzene carbons and indices 7-16 correspond to naphthalene carbons. Here, interactions are limited to nearest neighbors.

atomic repulsion energies of atomic orbitals i and j, $\gamma_{12} = \left(r_{ij}^2 + \gamma_{11}^2\right)^{-1/2}$ (for atomic orbital center distance r_{ij}), and $\gamma_{14} = \left(4r_{ij}^2 + \gamma_{11}^2\right)^{-1/2}$. The formula does not decay to zero at long distances due to the γ_{11} term being distance independent. In the nearest neighbors approximation, this is acceptable as interaction distances between bonded sites are expected to remain within normal covalent bond lengths (1-2 Å). Therefore, we introduce a Yukawa-like screening factor to the one particle calculation between non-bonded sites:

$$H'_{ij} = \begin{cases} H_{ij} & i,j \text{ bonded} \\ H_{ij} \times \exp\left[-\frac{r_{ij}}{r_{cr}}\right] \frac{a_0}{r_{ij}} & i,j \text{ non-bonded} \end{cases} \tag{2.25}$$

where r_{cr} is an adjustable coupling radius and a_0 is the Bohr radius. At large r_{ij}, the interaction energy between sites i and j vanish and the SCF-minimized Fock matrix, and consequently density matrix, are no longer block diagonal (as seen in figure 2.2). As a result, discrete π-systems now interact. Coupling values are between non-bonded atoms are small relative to that of bonded atoms, but importantly non-zero. As a consequence of enabling this interaction, the π orbitals now overlap and the electronic

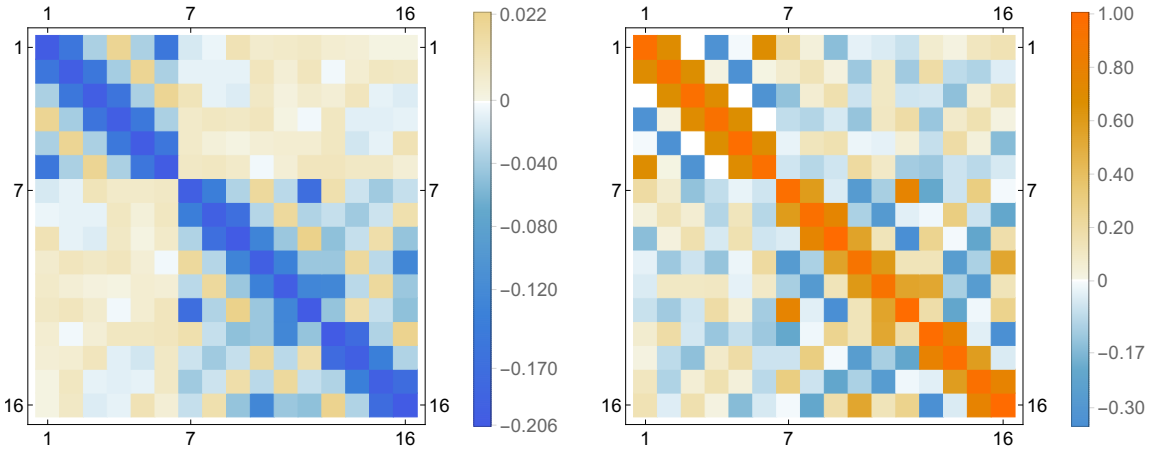

Figure 2.2: Matrix plots of the Fock (left, units in eV) and density (right, unitless) matrices for a ground state calculation of stacked benzene and naphthalene. Indices 1-6 correspond to benzene carbons and indices 7-16 correspond to naphthalene carbons. Here the adjustable coupling parameter is turned on ($r_{cr} = 2.0$Å).

energy is lowered in the given example.

2.1.5 Excited State Density Matrix

Thus far, our formulation is exact, but in principle does not provide a means for propagating the system in an excited state. We now formulate an expression for the density matrix in the excited state. First, consider the population matrix in equation 2.5. Naïvely, we might simply move some population from a lower orbital into a higher one. For example, in the following six state system, we can create an excitation by promoting an electron from occupied orbital #3 into unoccupied orbital #4.

$$\rho_{state}^{(ex)} = \begin{pmatrix} 2 & & & & & \\ & 2 & & & & \\ & & 1 & & & \\ & & & 1 & & \\ & & & & 0 & \\ & & & & & 0 \end{pmatrix} \tag{2.26}$$

$$\rho_{site}^{(ex)} = C\ \rho_{state}^{(ex)}\ C^{\dagger} \tag{2.27}$$

where we transform $\rho_{state}^{(ex)}$ to the site basis via unitary transform C. Note, in the state basis, both the Fock and density matrices are diagonal (all off-diagonal elements are

zero). This construction proves problematic in the context of equation 2.19. Recalling that the C diagonalizes the Fock matrix and $CC^\dagger = C^\dagger C = \hat{1}$:

$$\begin{aligned} i\hbar\frac{\partial \rho_{site}}{\partial t} &= [\mathcal{F}_{site}, \rho^{(ex)}_{site}] \\ &= \mathcal{F}_{site}\rho^{(ex)}_{site} - \rho^{(ex)}_{site}\mathcal{F}_{site} \\ &= C\mathcal{F}_{state}C^\dagger C\rho^{(ex)}_{state}C^\dagger - C\rho^{(ex)}_{state}C^\dagger C\mathcal{F}_{state}C^\dagger \\ &= C\mathcal{F}_{state}\rho^{(ex)}_{state}C^\dagger - C\rho^{(ex)}_{state}\mathcal{F}_{state}C^\dagger \\ &= C\left(\mathcal{F}_{state}\rho^{(ex)}_{state} - \rho^{(ex)}_{state}\mathcal{F}_{state}\right)C^\dagger \\ &= C\left[\mathcal{F}_{state}, \rho^{(ex)}_{state}\right]C^\dagger \\ &= 0 \end{aligned} \tag{2.28}$$

where, in the final step, two diagonal matrices necessarily commute. Since the time derivative of the density matrix is zero, the matrix is stationary, will not evolve, and no electrodynamics will occur.

We, instead, define an excitation operator

$$A^\dagger_N = \sum_{p,h} z^*_{ph}\phi^\dagger_p\phi_h \tag{2.29}$$

where z is the contribution of the elementary excitation from $h \to p$ for excitation N. The operators ϕ are state operators which are further broken down as a linear combination of site operators, a

$$\phi^\dagger_n = \sum_j C^*_{jn}a^\dagger_j \tag{2.30}$$

Again, matrix C is the unitary transform that changes from state to site basis. Operator $A^\dagger_N$ acts on the HF ground state to produce the N^{th} excited state.

$$A^\dagger_N\left|HF\right\rangle = \left|N\right\rangle \tag{2.31}$$

The ground state reduced density matrix is given by

$$\rho_{rs} = \langle a^\dagger_s a_r\rangle = \left\langle HF\right| a^\dagger_s a_r \left|HF\right\rangle \tag{2.32}$$

where the HF ground state is generated by populating the lowest energy states from the vacuum state

$$|HF\rangle = \phi_1^\dagger \dots \phi_n^\dagger |0\rangle \tag{2.33}$$

for n number of electrons. Using equation 2.30,

$$\begin{aligned}
\rho_{rs} &= \sum_{ij} \langle HF| \phi_i^\dagger \phi_j |HF\rangle c_{si}^* c_{rj} \\
&= \sum_{ij} \delta_{ij} \langle HF| \phi_i^\dagger \phi_i |HF\rangle c_{si}^* c_{ri} \\
&= \sum_{i \leq n} c_{si}^* c_{ri}
\end{aligned} \tag{2.34}$$

where we have now restricted i to only the occupied orbitals. We now write the density matrix for singly excited state N in the same fashion.

$$\begin{aligned}
\gamma_{rs}^N &= \langle N| a_s^\dagger a_r |N\rangle \\
&= \sum_{nm} \langle N| \phi_n^\dagger \phi_m |N\rangle c_{sn}^* c_{rm} \\
&= \sum_{nm} \sum_{p'h'} \sum_{ph} c_{sn}^* c_{rm} z_{p'h'}^* z_{ph} \langle HF| \phi_{h'}^\dagger \phi_{p'} \phi_n^\dagger \phi_m \phi_p^\dagger \phi_h |HF\rangle
\end{aligned} \tag{2.35}$$

Indices p and p' span over unoccupied orbitals; indices h and h' span over occupied orbitals in the ground state; and indices m and n span all state orbitals, occupied and unoccupied. We now turn our efforts to simplifying the integral

$$I = \langle HF| \phi_{h'}^\dagger \phi_{p'} \phi_n^\dagger \phi_m \phi_p^\dagger \phi_h |HF\rangle \tag{2.36}$$

Note, fermion orbitals operators must anti-commute

$$\{\phi_n, \phi_m^\dagger\} = \phi_n \phi_m^\dagger + \phi_m^\dagger \phi_n = \delta_{nm} \tag{2.37}$$

Rearrangement gives

$$\phi_n \phi_m^\dagger = \delta_{nm} - \phi_m^\dagger \phi_n \tag{2.38}$$

Furthermore, we can use the properties of $|HF\rangle$. Namely, a creation on an occupied orbital and annihilation on an unoccupied orbital both destroy the ket.

We start by moving $\phi_{p'}$ to the right.

$$\begin{aligned}
I &= \langle HF| \phi^\dagger_{h'}\phi_{p'}\phi^\dagger_n\phi_m\phi^\dagger_p\phi_h |HF\rangle \\
&= \delta_{np'} \langle HF| \phi^\dagger_{h'}\phi_m\phi^\dagger_p\phi_h |HF\rangle - \langle HF| \phi^\dagger_{h'}\phi^\dagger_n\phi_{p'}\phi_m\phi^\dagger_p\phi_h |HF\rangle \\
&= \delta_{np'} \langle HF| \phi^\dagger_{h'}\phi_m\phi^\dagger_p\phi_h |HF\rangle + \langle HF| \phi^\dagger_{h'}\phi^\dagger_n\phi_m\phi_{p'}\phi^\dagger_p\phi_h |HF\rangle \\
&= \delta_{np'} \langle HF| \phi^\dagger_{h'}\phi_m\phi^\dagger_p\phi_h |HF\rangle + \delta_{pp'} \langle HF| \phi^\dagger_{h'}\phi^\dagger_n\phi_m\phi_h |HF\rangle - \langle HF| \phi^\dagger_{h'}\phi^\dagger_n\phi_m\phi^\dagger_p\phi_{p'}\phi_h |HF\rangle \\
&= \delta_{np'} \langle HF| \phi^\dagger_{h'}\phi_m\phi^\dagger_p\phi_h |HF\rangle + \delta_{pp'} \langle HF| \phi^\dagger_{h'}\phi^\dagger_n\phi_m\phi_h |HF\rangle + \langle HF| \phi^\dagger_{h'}\phi^\dagger_n\phi_m\phi^\dagger_p\phi_h\phi_{p'} |HF\rangle \\
&= \delta_{np'} \langle HF| \phi^\dagger_{h'}\phi_m\phi^\dagger_p\phi_h |HF\rangle + \delta_{pp'} \langle HF| \phi^\dagger_{h'}\phi^\dagger_n\phi_m\phi_h |HF\rangle \\
&= \delta_{np'} I_1 + \delta_{pp'} I_2
\end{aligned} \tag{2.39}$$

In I_1 move $\phi_{h'}$ to the right.

$$\begin{aligned}
I_1 &= \langle HF| \phi^\dagger_{h'}\phi_m\phi^\dagger_p\phi_h |HF\rangle \\
&= \delta_{h'm} \langle HF| \phi^\dagger_p\phi_h |HF\rangle - \langle HF| \phi_m\phi^\dagger_{h'}\phi^\dagger_p\phi_h |HF\rangle \\
&= \delta_{h'm} \langle HF| \phi^\dagger_p\phi_h |HF\rangle + \langle HF| \phi_m\phi^\dagger_p\phi^\dagger_{h'}\phi_h |HF\rangle \\
&= \delta_{h'm} \langle HF| \phi^\dagger_p\phi_h |HF\rangle + \delta_{h'h} \langle HF| \phi_m\phi^\dagger_p |HF\rangle - \langle HF| \phi_m\phi^\dagger_p\phi_h\phi^\dagger_{h'} |HF\rangle \\
&= \delta_{h'm} \langle HF| \phi^\dagger_p\phi_h |HF\rangle + \delta_{h'h} \langle HF| \phi_m\phi^\dagger_p |HF\rangle
\end{aligned} \tag{2.40}$$

$\phi^\dagger_p$ can be interpreted as annihilating to the left. Therefore the first term vanishes and I_1 reduces to

$$I_1 = \delta_{h'h}\delta_{mp} \tag{2.41}$$

I_2 follows similarly.

$$\begin{aligned} I_2 &= \langle HF| \phi^\dagger_{h'}\phi^\dagger_n\phi_m\phi_h |HF\rangle \\ &= -\langle HF| \phi^\dagger_n\phi^\dagger_{h'}\phi_m\phi_h |HF\rangle \\ &= \langle HF| \phi^\dagger_n\phi_m\phi^\dagger_{h'}\phi_h |HF\rangle - \delta_{h'm}\langle HF| \phi^\dagger_n\phi_h |HF\rangle \\ &= \delta_{h'h}\langle HF| \phi^\dagger_n\phi_m |HF\rangle - \langle HF| \phi^\dagger_n\phi_m\phi_h\phi^\dagger_{h'} |HF\rangle - \delta_{h'm}\langle HF| \phi^\dagger_n\phi_h |HF\rangle \\ &= \delta_{h'h}\langle HF| \phi^\dagger_n\phi_m |HF\rangle - \delta_{h'm}\langle HF| \phi^\dagger_n\phi_h |HF\rangle \end{aligned} \tag{2.42}$$

Take ϕ_h to create to the left. Then I_2 reduces to

$$I_2 = \delta_{h'h}\langle HF| \phi^\dagger_n\phi_m |HF\rangle - \delta_{h'm}\delta_{nh} \tag{2.43}$$

Since the HF ground state only has the lowest orbitals occupied, the remaining integral reduces to

$$\langle HF| \phi^\dagger_n\phi_m |HF\rangle = \delta_{n_{occ},m_{occ}} \tag{2.44}$$

where n and m here span only the occupied orbitals. Finally, the integral simplifies to

$$\begin{aligned} I &= \langle HF| \phi^\dagger_{h'}\phi_{p'}\phi^\dagger_n\phi_m\phi^\dagger_p\phi_h |HF\rangle \\ &= \delta_{hh'}\delta_{np'}\delta_{mp} - \delta_{pp'}\delta_{mh'}\delta_{nh} + \delta_{pp'}\delta_{hh'}\delta_{n_{occ},m_{occ}} \end{aligned} \tag{2.45}$$

As expressed in the above equation, the density matrix is in the state space. That is, h, h', p, p', m, and n iterate through the system's molecular orbitals. We examine these terms independently to interpret their physical meaning. The first term iterates m and n through the unoccupied orbitals and adds magnitude to the population and coherence terms. The second term iterates m and n through the occupied orbitals and subtracts magnitude. The third term accounts for the otherwise unaffected populations of the ground state.

Figure 2.3 plots these three terms for a six-state system (for which the first three

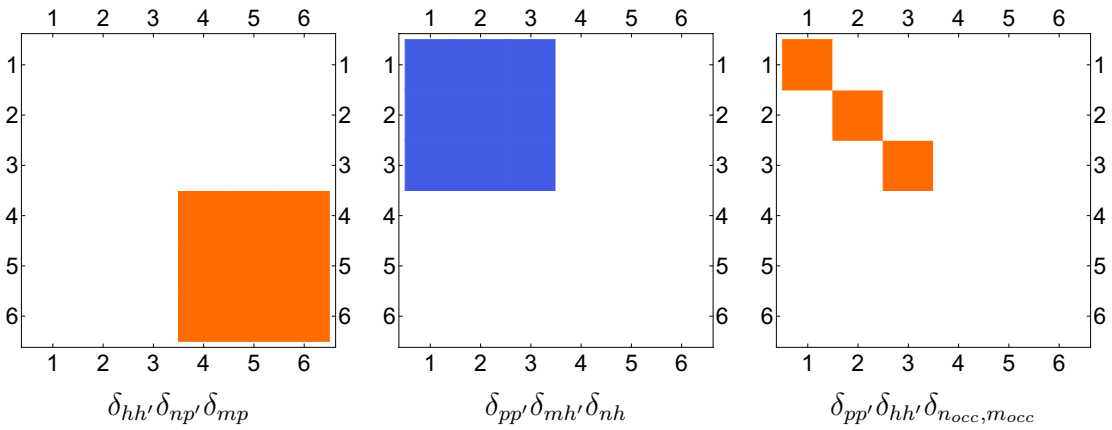

Figure 2.3: Matrix plots of each term in equation 2.45 for a sample six state system in the HF state basis. Plots are normalized to unity where orange represents an added quantity and blue represents a subtracted quantity. Values are unitless.

states are occupied in the ground state). The third term resembles the state population matrix of the naïve approach. The former two terms, however, not only shift populations to the excited states but also introduce coherences in the form of off-diagonal matrix elements. The final excited density matrix is not diagonal in this formulation but represents a pure state. The commutation of the generating Fock matrix and this excited state matrix is non-zero and, therefore, electron dynamics can occur.

2.1.6 Simulation Parameters

We are interested in studying the charge transfer properties of four DBA triad systems studied by Moore and coworkers[40, 43]. Figure 2.4 shows the breakdown of each molecule. Each molecule has the same acceptor (C_{60} derivative) and polymeric donor (carotenoid) is are differentiated by their side groups on porphyrin bridges. In Molecule 1 (**M1**), the porphyrin ring is dressed by tri-substituted aryl groups. Molecule 2 (**M2**) is identical to **M1** except for the two methyl substitutions on the aryl groups. Molecule 3 (**M3**) has methyl and *n*-butyl groups on the pyrrole sites.

Molecule 4 (**M4**), similar to **M3**, replaces the longer chain with ethyl groups. Additionally, the amide group is reversed on **M4** relative to those of **M1-3**.

The four molecules are constructed and pre-optimized using external software, AVOGADRO[62]. Note, one can easily convert a standard `PDB` file to TINKER's version of `XYZ` via the provided routine `PDBXYZ`. Each optimized structure thermalized with a Nosé-Hoover thermostat set to $T = 300$ K in vacuum for 10 ps at a time step of 0.01 fs (10^6 time steps). This temperature was chosen as an expected operating temperature for these systems in a real environment. The time-step must be chosen to capture electron dynamics adequately and should be shorter than the oscillation period of the time evolution operator. We therefore want $\delta t < \hbar/E$ or $E < \hbar/\delta t$, and our choice allows for the Liouvillian eigenspectrum to range between $\pm$ 1.2 Hartrees. The adjustable coupling parameter was set to 1.5 Å as it is slightly larger than the carbon-carbon bond distance of benzene. Figure 2.5 shows the molecules quickly achieve the target temperature and stay within a range of 20 K. From the last 5 ps, we sample 50 geometries (every 100 fs) as the basis for the ensemble. (Note, we retain the coordinates and velocities of each sample.) We parameterize the molecular mechanics and semiemperical quantum mechanics with the MM3 parameter set[63].

We perform a single point calculation on each of these samples to determine the correct excited state to choose. For these triads, the excitation should resemble an exciton localized to the porphyrin bridge. We determine the correct excitation using the eigenvectors of the Fock and CI matrices in the following way. Let $C_{i,j}$ be the contribution of atomic orbital i to molecular orbital j and z_{hp}^N be the contribution of elementary excitation (from molecular orbital h to p) to the N^{th} excited state. Then, for any CI excitation, N, we calculate the percentage of the hole density localized to

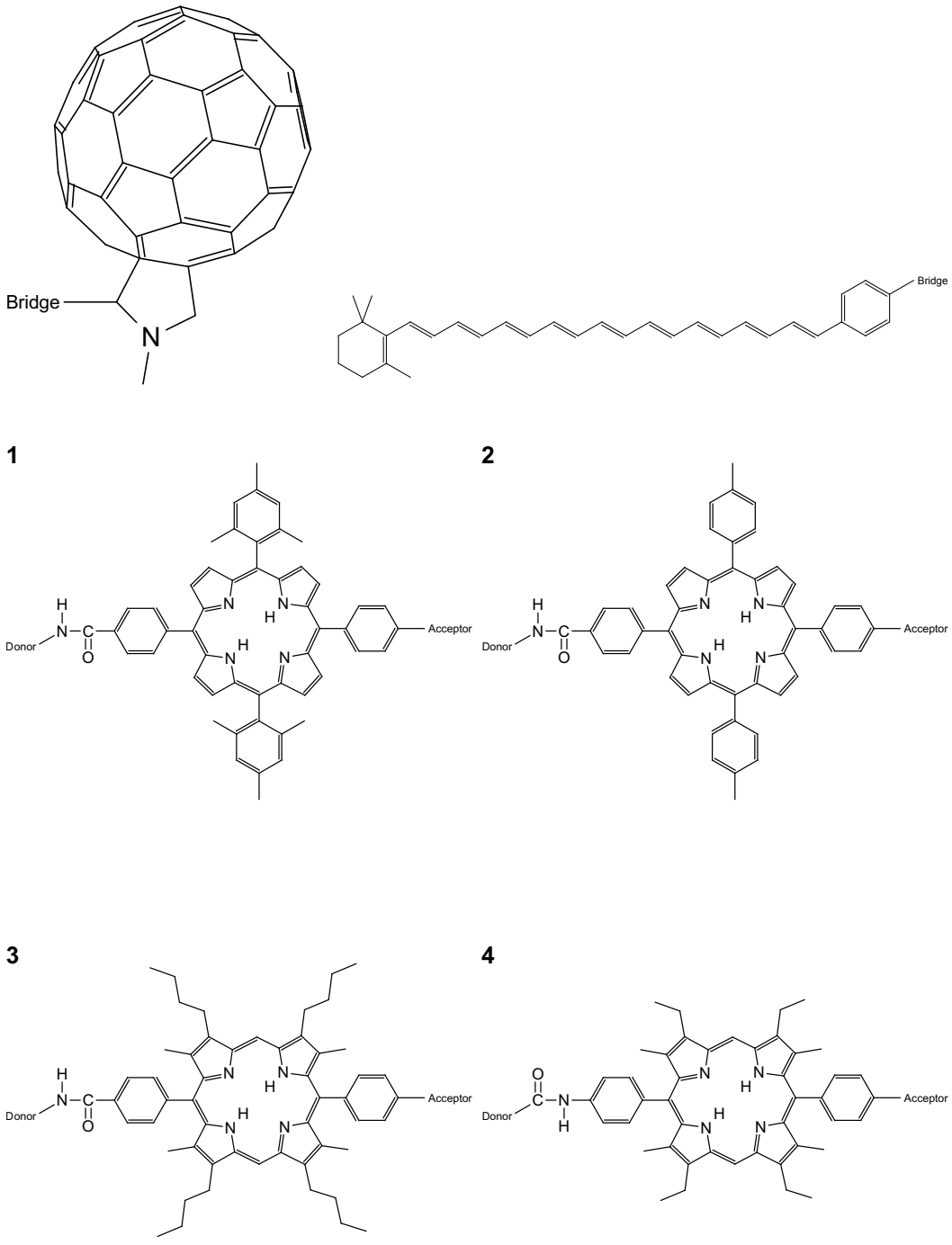

Figure 2.4: Breakdowns of the four molecules simulated in this study. Each molecule have a common acceptor (Fullerene derivative, top left) and donor (carotenoid, top right) moiety. The bridging moieties differentiate the molecules (labeled **1-4**).

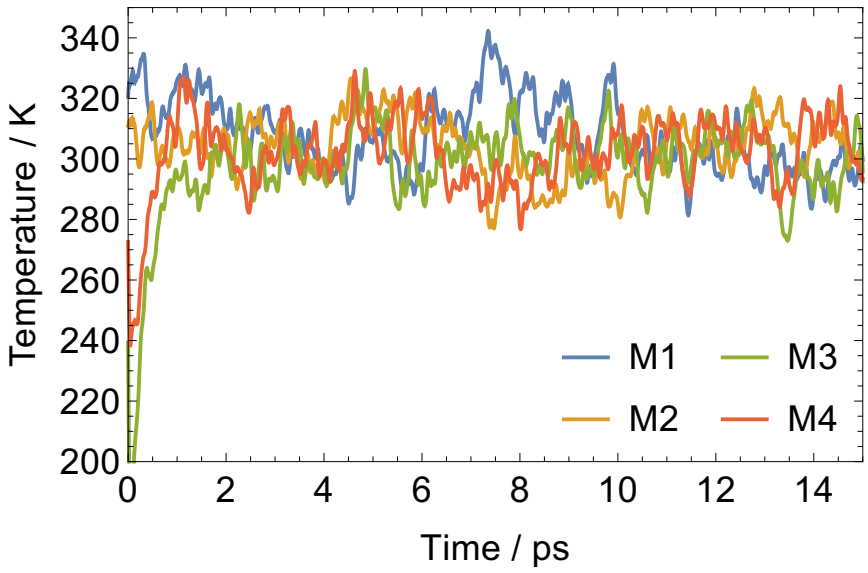

Figure 2.5: Temperature vs time plot of the thermalization of each molecule.

the bridge as

$$P_{\text{hole}}(N) = \sum_{i \in \text{Brg}} \sum_{h \in \text{occ}} |C_{i,h}|^2 \left| z_h^N \right|^2 \tag{2.46}$$

where i sums over atoms in the bridge moiety and h over the occupied molecular orbitals. Correspondingly, the percentage of electron density can be calculated as

$$P_{\text{elec}}(N) = \sum_{i \in \text{Brg}} \sum_{p \in \text{unocc}} |C_{i,p}|^2 \left| z_p^N \right|^2 \tag{2.47}$$

for p over the unoccupied molecular orbitals. We choose the lowest energy excitation which has P_{hole} and P_{elec} larger than 0.75 (at least 75% of the exciton density is localized to the bridge). For most configurations, this corresponds to the first excited state ($\approx$ 2.0 eV). Each geometry is excited at $t = 0$ and then freely relaxes given an energy kick for 10 ps. The thermostat and time step are retained at 300 K and 0.01 fs respectively.

This procedure essentially recreates the scenario of a molecule undergoing thermal fluctuations at operational temperatures, becoming photoexcited, and then undergoing electronic and nuclear relaxation.

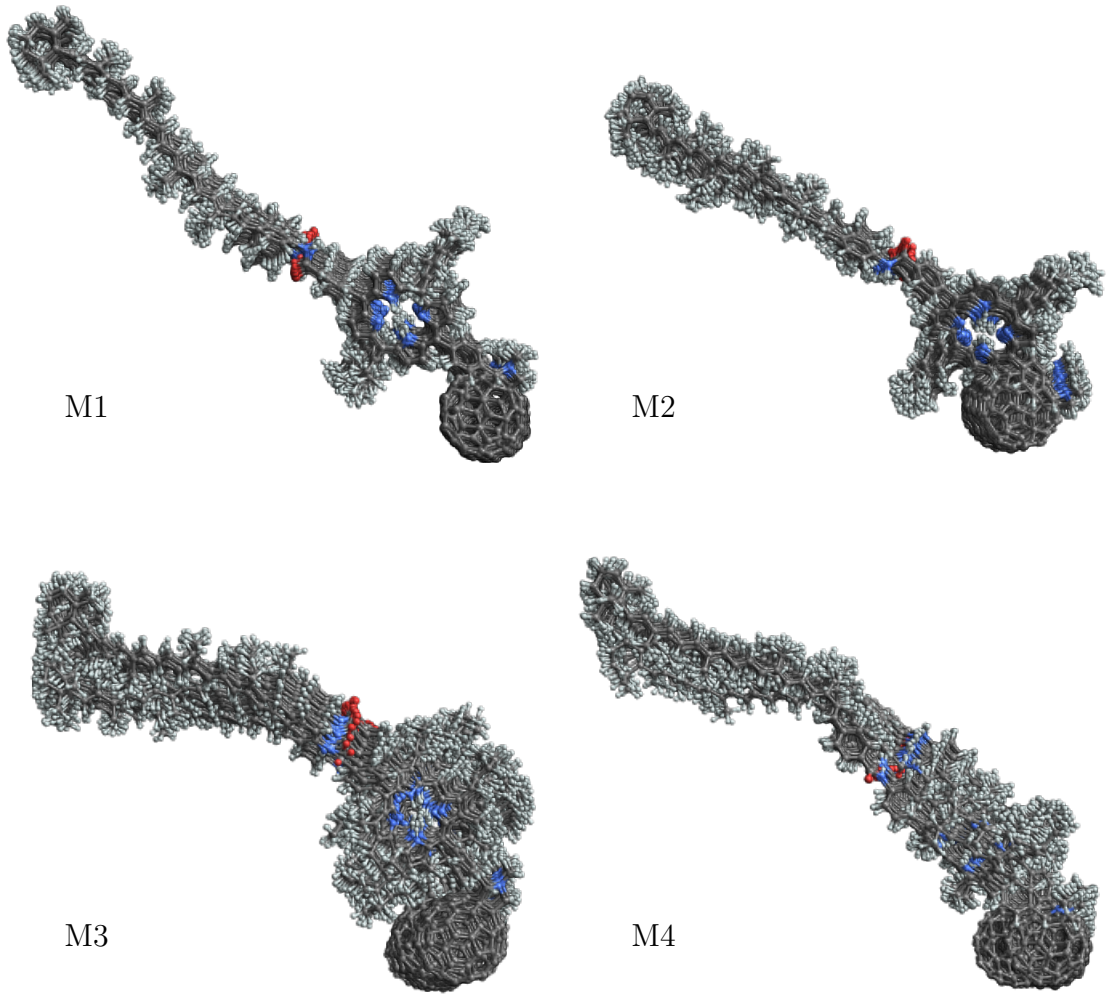

Figure 2.6: Overlay of starting geometries for molecules M1-M4.

2.2 Analysis

2.2.1 Rate Approximation

The diagonal terms of the density matrix (bond-charge matrix) give the occupation number for an orbital. Thus in the site-basis, ρ_{ii} represents an electron count on orbital i centered on some π-active nucleus. Then, a partial trace of the density matrix, choosing for atoms assigned to a particular moiety, accounts for the total number of electrons localized to that moiety; the full trace of the density matrix reflects the total electron count of the system. The effective charge (population gain) is this partial trace of the density matrix at time t subtracted by the ground state charge. For the acceptor moiety,

$$q_{\mathrm{Acc}}(t) = \sum_{i \in \mathrm{Acc}} (\rho_{ii}(t) - \rho_{ii}^{gs}) \tag{2.48}$$

For multi-molecular systems, this calculation can be extended to consider the charging of whole molecules. We measure charge transfer by following the excess population in acceptor moiety in time for each trajectory. Fitting the charge transfer plots from each trajectory to a logistic growth function provides an approximate form:

$$q(t) = \frac{q_{max}}{1 + \exp[-k(t - t_c)]} + q_0 \tag{2.49}$$

where q_{max} is the long-time maximal charge transferred, k is a steepness factor, t_c is the function center (50% completion time), and q_0 is an initial charge amount. Non-zero q_0 represents some partial charge transfer character in the initial excited state. From these regressions, the fractional completion time can be computed as

$$t_f = t_0 + \frac{1}{k} \ln \left[\frac{f}{1 - f} \right] \tag{2.50}$$

We take the 95% ($f = 0.95$) completion time to represent a charge transfer time-scale; the inverse of this is interpreted as the reaction rate.

2.2.2 Time-Frequency Analysis

We are interested in evaluating the nuclear dynamics that occur during the charge transfer process. For this, we use the velocity autocorrelation function

$$C_{vv}(\tau) = \sum_{t=0}^{T-\tau} \sum_{i=1}^{N} \vec{v}_i(t) \cdot \vec{v}_i(t+\tau) \tag{2.51}$$

where, τ is the lag time, T is the total time, N is the number of particles considered, and $\vec{v}_i(t)$ is the velocity of particle i at time t. Fourier transform of the correlation function will give a power spectrum, a distribution of energy across frequencies which here is analogous to a vibrational spectrum. However, this only provides a time-averaged description and is only well suited for a stationary signal; it does not decode information about transient signals. We suspect that certain nuclear modes activate and deactivate during the charge transfer process. A time-frequency spectrogram of a nonstationary time-series will resolve evolution of the vibrational power spectrum[64]. We apply the Short-Time Fourier Transform (STFT) method on the velocity autocorrelation function to visualize the evolution of the nuclear dynamics in time:

$$S(\tau, \omega) = \int_{-\infty}^{\infty} f(t) g(t-\tau) \mathrm{e}^{-\mathrm{i}2\pi\omega t} dt \tag{2.52}$$

$$g(t-\tau) = \begin{cases} \frac{1}{2} - \frac{1}{2} cos(\frac{\pi(\tau - t + \mathcal{T})}{\mathcal{T}}) & |\tau - t| \leq \mathcal{T} \\ 0 & |\tau - t| > \mathcal{T} \end{cases} \tag{2.53}$$

where $\mathcal{T}$ is the window radius of the Hanning window function $g(t)$, defined in equation 2.53. Nakai and coworkers have used this technique for the analysis of *ab initio* dynamics simulations of collision reactions on small molecules[65, 66, 67, 68, 69, 70]. STFT resolution is restricted to the Heisenberg-Gabor limit such that wider window functions resolve frequencies better, but sacrifice time resolution whereas thinner windows give better time resolution but sacrifice frequency resolution[71, 72]. For a given

window radius, uncertainty in frequency can be computed as $\Delta\omega \approx (2cT)^{-1}$. STFT was performed with a Hanning window of width $\mathcal{T} = 600fs$ ($\Delta\omega \approx 28$ cm^{-1}). At this resolution, we may not be able resolve finer details of the vibrational spectrograph, but it sufficiently suitable for our needs. Edge effects occur as a result of this analysis on finite time-series. At the temporal extrema, less information is available resulting in an overall smoothing and diminishing of the spectral intensity. To mitigate this, we include dynamics from 2 ps prior to the photoexcitation.

2.2.3 Screening Vibrational Modes

The discrete STFT process returns a 2-dimensional array in time and frequency. By plotting the spectrogram, we can visually inspect for and pick out modes that seem to correspond to the charge transfer time. Figure 2.7 follows the time-series of three modes $(\omega_1, \omega_2, \omega_3)$ from an example spectrogram where the vertical axis is the power spectrum intensity of a particular mode. In this example, we say the system undergoes charge transfer between $t = 1ps$ and $t = 5ps$ (gray region). The first mode (blue) has intensity only during the CT period, but no intensity before or after. The second mode (orange) behaves exactly opposite. It has intensity (activity) before and after CT, but not during CT. The third mode (green) is relatively active during the entire example simulation.

We find a correlation value between the spectral intensity of a mode and the charge transfer regime through the following steps. First, shift the intensities down for a mode's (ω_k) spectral time series such that its average lies at 0,

$$\tilde{S}(\tau, \omega_k) = S(\tau, \omega_k) - s_k \tag{2.54}$$

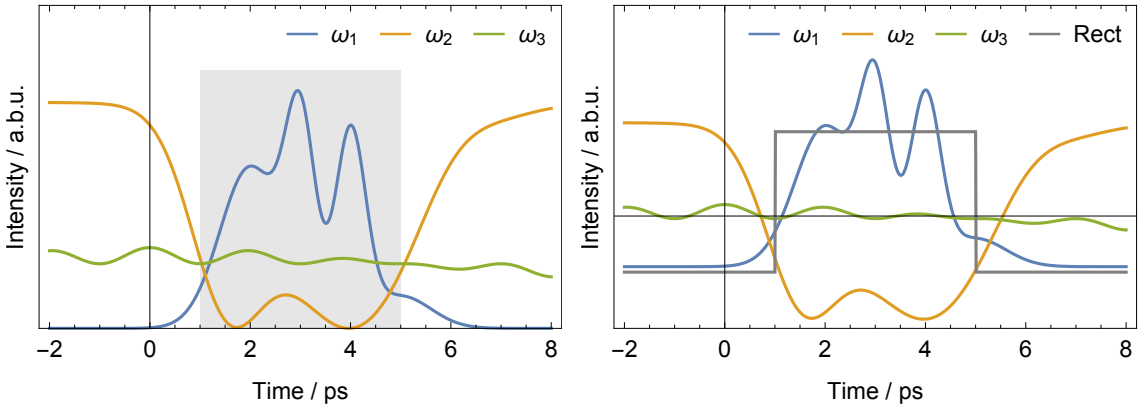

Figure 2.7: (Left) Spectral Intensity vs Time of three example modes. The gray region represents the time span where charge actively migrates to the acceptor. (Right) Down shifted intensities with shifted rectangle function.

for some s_k

$$\int_{t_1}^{t_2} \tilde{S}(\tau, \omega_k) d\tau = 0 \tag{2.55}$$

where t_1 and t_2 are the temporal limits of the vibrational time considered. Then define a rectangle function that is unity in the gray region and zero otherwise and again shift it down so that its average value is 0 (fig 2.7 (right)).

$$\int_{t_1}^{t_2} \text{Rect}(\tau) d\tau = 0 \tag{2.56}$$

The result of these shifts is plotted in Figure 2.7 (right). The inner product of the shifted spectral time-series, $\tilde{S}$, and the shifted rectangle function

$$C(\omega_k) = \int_{t_1}^{t_2} \tilde{S}(\tau, \omega_k) \text{Rect}(\tau) d\tau \tag{2.57}$$

gives a correlation strength between the mode and the charge transfer process. A large positive value indicates direct correlation: the mode is specifically active during the charge transfer. A large negative value indicates the mode is inactive during charge transfer and it is active otherwise. Small correlation values indicate either poor correlation or low spectral amplitude, either of which implies the mode is of no interest. Thus, this analysis gives a compact value that relates the activity of a mode to the charge transfer process.

Chapter 3

Charge Transfer Dynamics in Model Triad Systems

3.1 Ensemble Configurations

Given the non-trivial construction of the molecules of interest and their high number of vibrational degrees of freedom, we must introduce a statistical component to our investigations. Generally, a large selection of random configurations will result in a distribution around an average case. However, given constraints in time and resources, we are limited to a finite number of configurations to simulate. To produce these initial configurations, we uniformly sample ground state thermal fluctuations. Despite this uniform sampling, we can show the configurations are sufficiently random. The time correlation function, $C_{qq}(t) = \langle q(t)q(0)\rangle$, of the geometries shows the correlation strength between two configurations separated in time. We plot in Figure 3.1 both the correlation function and exponential decay fit. The correlation time can be found

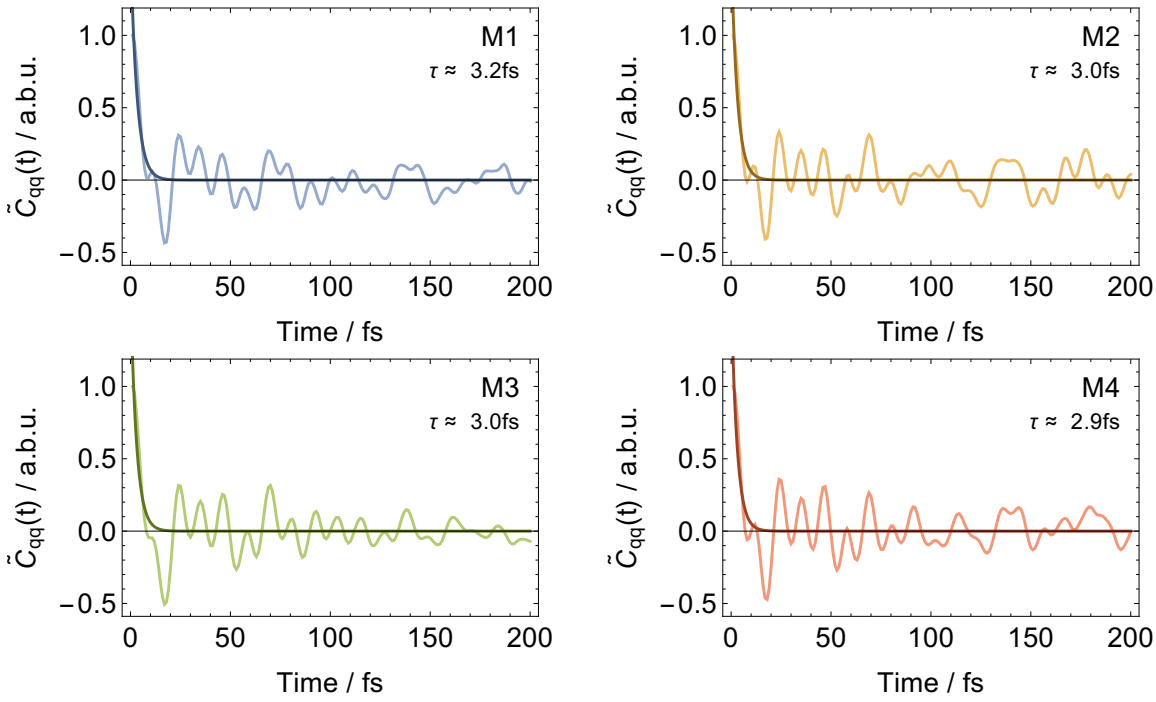

Figure 3.1: Geometry correlation plot (light) and exponential decay fit (dark) of the ground state dynamics for each molecule. Correlation functions are normalized to the initial value: $\tilde{C}_{qq}(t) = C_{qq}(t)/C_{qq}(0)$.

from the integral

$$\tau_c = \int_0^\infty \frac{C_{qq}(t)}{C_{qq}(0)} dt \tag{3.1}$$

The results from these ground state dynamics reveal a correlation time of about 3 fs. Consequently, the system does not retain memory from a previous configuration and samplings separated in time by the correlation time, τ_c, or longer can be considered uncorrelated. Our sampling time is $t_{samp} \sim 30\ \tau_c$ and we can consider the initial geometries to represent random thermal configurations.

Thermal fluctuations in the molecular configuration modulate the excitation energy and we need to check that the distribution of excitation energies of our initial geometries match that of the full configurational space. We assume in the last 5

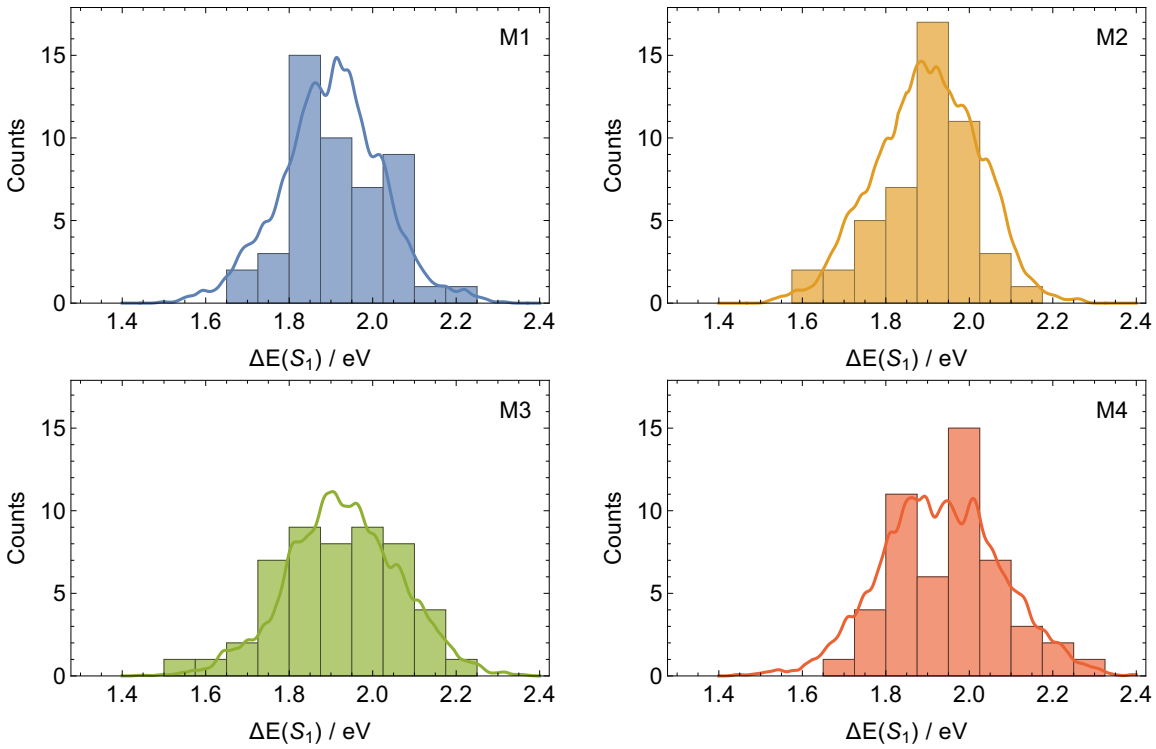

Figure 3.2: Energy distributions for the first excited singlet state for each molecule. Smooth distributions represent the distribution of excitation energies from 5000 ground state samplings each. Discrete histograms represent the 50 initial configurations chosen for the ensemble.

ps of the ground state dynamics, each system samples operational temperature configurations and that a selection of 5000 geometries adequately represents the full configurational space. Figure 3.2 compares the energy distributions of the selected initial geometries and subset of 5000 geometries. We find that not only the excitation energies not only match the distribution of the broader configuration space, but the average excitation energies also are in excellent agreement with experimental absorption spectra of the lowest excitation energy for the porphyrin moieties (∼1.9 eV)[40, 43] as well as spectra computed using higher theory levels[73, 74].

3.2 Approximated Charge Transfer Rates

Recently, coherent dynamics have been used to explain charge transfer at the sub-picosecond time scale[75, 76, 77]. In this strong electronic coupling regime, constructive interference of CT pathways opens up the possibility of rapid transfer despite large physical separation between chromophores in photosynthetic proteins[78, 79, 80]. Conversely, low electronic coupling typically entails electron hopping from site to site and results in a much slower transfer process. Figure 3.3 shows the excess electron population in (charge transferred to) the C_{60} moieties for each trajectory. In most cases, CT does not begin until after 2 ps and completes at various times. The simulations indicate longer than picosecond timescales and therefore low electronic coupling between bridge and acceptor.

Following excitation at $t = 0$, most trajectories do not immediately begin the CT process. Some excited state dynamics almost always ensues as the system responds to the impulsive vertical excitation before the initiating charge transfer. Once CT has completed, the charge separation is stable and long lived. We have not evolved the trajectories passed 10 ps; formation of the second charge transfer state (C^+-P-C_{60}^-) is expected to be on the order of tens to hundreds of picoseconds with a lifetime into the nanosecond scale.

All four molecules successfully undergo some amount of charge transfer within 10 ps. This implies that charge transfer itself is determined by the offset between HOMO and LUMO energy levels of the donor and acceptor moieties. Additionally, this agrees with experimentally observed quantum yield of unity. However, the amount and rate of CT depend on the finer details of the molecular structure. We see the range of total charge transferred, between 0.05-0.35 q is consistent through all molecules, while the

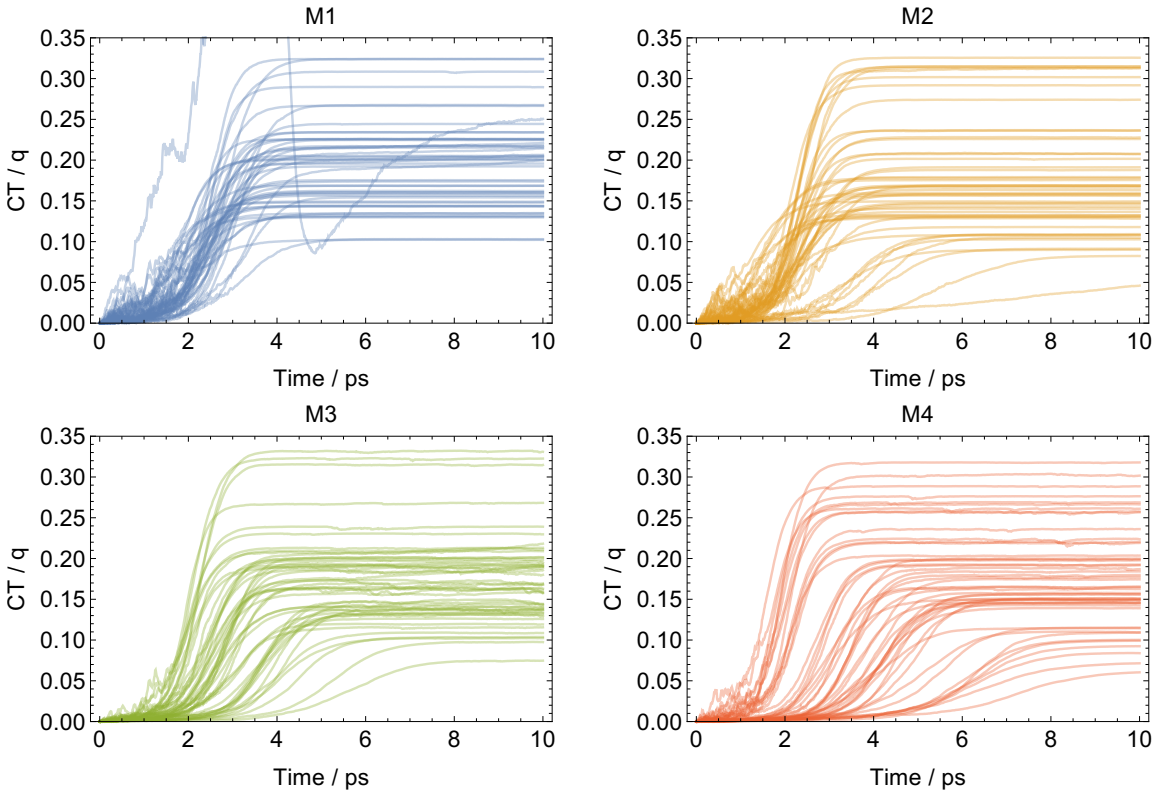

Figure 3.3: Charge accumulation plots for each trajectory in **M1-4**. Plots generally follow a logistic growth regression excluding one anomalous trajectory in **M1**.

distributions of the CT curves differ among them.

Charge accumulation follows a logistic growth function quite well ($R^2 > 0.99$), excluding one errant trajectory in **M1**. Further investigation of this simulation shows the lowest energy porphyrin excitation ~ 3 eV producing an extremely "hot" exciton. This excitation is anomalously high compared to the 1.9 eV for all other trajectories. Therefore, we deem this a significant outlier and exclude it from further analysis. We approximate the CT rate constant as the inverse of the 95% completion time (using the logistic fits) which are plotted. Figure 3.4 adds crucial information about the CT rates, which are difficult to assess from the multitude of trajectories in Figure 3.3. From Figure 3.4, we can clearly see that difference in chemical structure, even subtle ones, alter CT rates and their distribution. While our approach overestimates rates for **M1** and **M2** as compared to experiment, the rates for **M3** and **M4** are in good agreement with experimental results[42, 43] (dashed lines). However, the overestimation is systematic between the pair of molecules and we are able to reproduce their relative similarity of rates seen in experiment. Furthermore, our results differ from the LSC approach by less than a factor of 10[6, 50].

Interestingly, trajectories that take the longest to initiate the charge transfer process tend to have a proportionately lower total amount of charge transferred. **M2-4** have prominent examples of such where CT does not begin until about 4 ps after excitation and ultimately settles at 0.1 q of charge transferred. In these instances, the system undergoes vibrational relaxation during the delay time. In the frame work of the potential energy surfaces, the system relaxes into an energy well and cannot make the transition into a charge transfer state. Such trajectories may be examples of cold excitons. Here, excess excitation energy is lost and dissociation occurs along low energy paths resulting in slower CT[81].

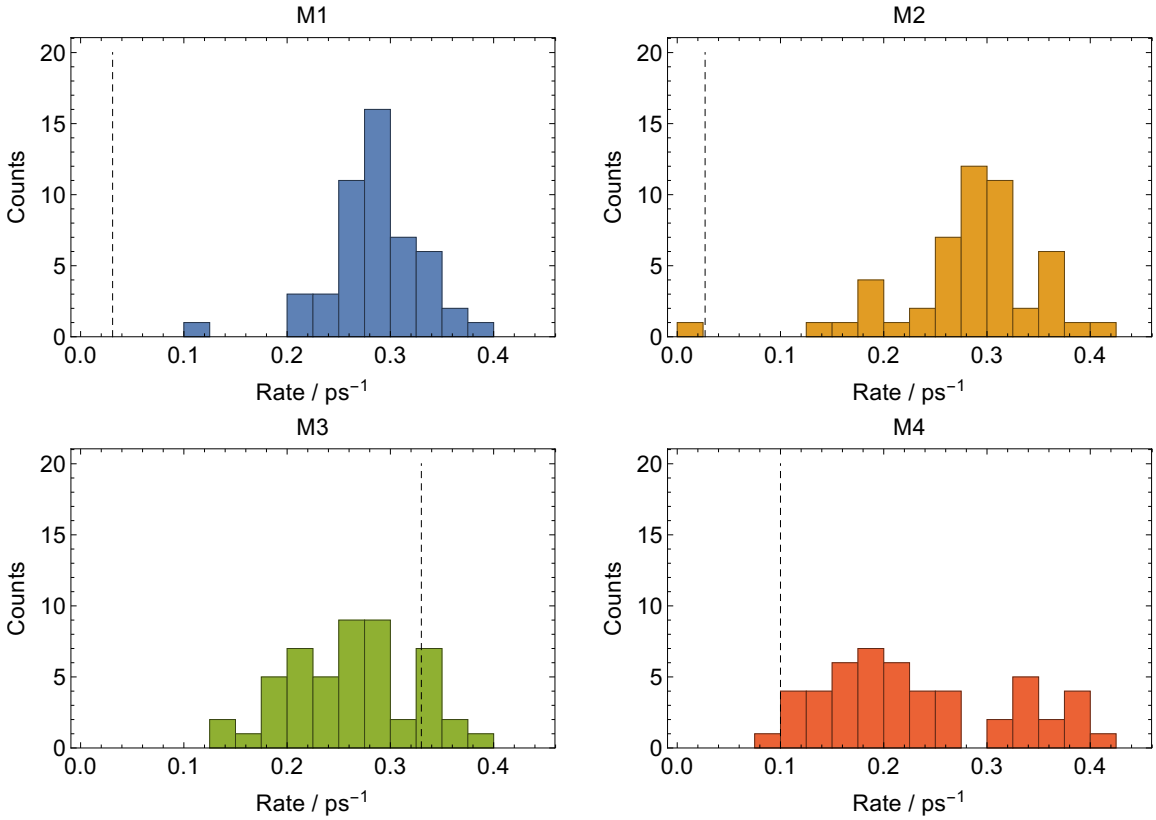

Figure 3.4: Histogram of the charge transfer rates of each ensemble. Averaged rates for **M1-4** are 0.28 ± 0.04 ps^{-1}, 0.28 ± 0.07 ps^{-1}, 0.26 ± 0.06 ps^{-1}, and, 0.23 ± 0.06 ps^{-1} respectively. Dashed lines indicate experimentally reported values with rate for **M4** reported as a maximum[43].

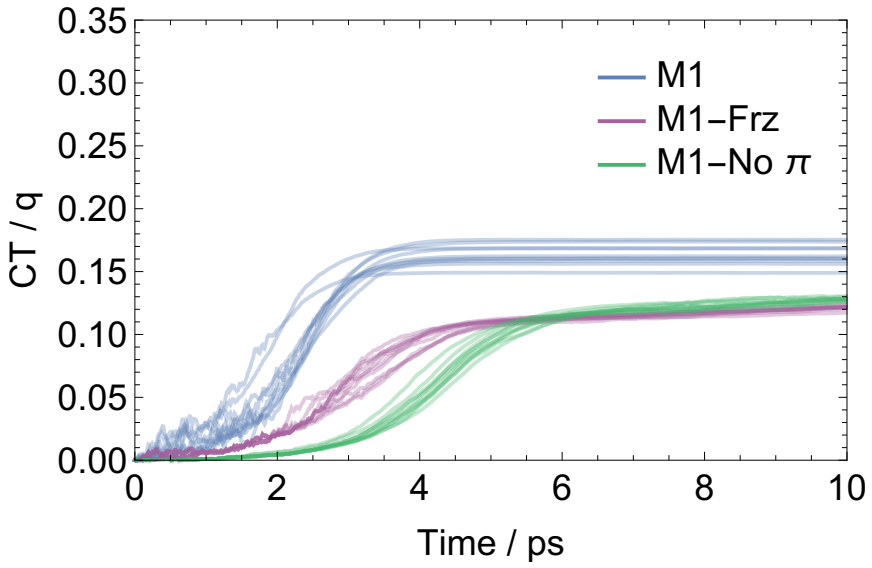

Figure 3.5: Charge accumulation plots for select trajectories in **M1** (M1, blue), trajectories with frozen side groups (M1-Frz, purple), and trajectories with no π contribution from side groups (M1-No π, pale green).

We further explore the electronic effect of the aryl side groups in **M1**. We take the subset of 10 initial configurations that were chosen for vibrational analysis (discussed in the next chapter) and resimulate the excitation and relaxation dynamics with two variations. In the first, we freeze the aryl group atoms while maintaining their electronic contribution to the π-system ("quantum active"). In the second, side group atoms are free to move but provide no quantum contribution ("quantum inactive"). Figure 3.5 plots the charge accumulation in the acceptor for these trajectories, including the original ten.

Within each sub-ensemble, CT behavior remained similar to one another, however, imposing these exclusions mitigates CT. Computed as previously, CT rate constants are 0.29 ps^{-1}, 0.20 ps^{-1}, and 0.16 ps^{-1} for M1, M1-Frz, and M1-No π sub-ensembles, respectively. Freezing side aryl group nuclei causes nearly a 30% reduction in CT rate and slightly diminishes the quantity of charge transferred, qualitatively agreeing with results from simulations of similar motifs in Ref [46]. Disabling the π contribution

from the side aryl groups further reduces the CT rate and quantity. Notably, the CT curves this sub-ensemble are smooth relative to the other two. Very little excited state electronic dynamics occurs until the actual CT process is initiated. This is also seen in the CT plots for **M3** and **M4**. The aryl groups modulate the electronic structure of the bridge at a small level. Furthermore, including the side groups into the conjugated system provides a relaxation pathway where the system can funnel excess electronic energy until energetic overlap becomes suitable for CT. This is consistent with **M3** and **M4** as their bridge side groups are not conjugated and overall these systems have slower CT according to our simulations. However, this does not agree with experimental results which indicate that formation of the initial CT state in **M1** and **M2** be much slower than that of **M3** and **M4**. As a final remark, these results indicate bridge side groups provide electronic and vibrational relaxation pathways that improve the charge transfer rate in such DBA triads.

Chapter 4

Transient Vibrational Dynamics Related to Charge Transfer in Model Triads

The compact form of the Marcus expression suggests that only few degrees of freedom are involved in the CT process. However, it is not altogether obvious how to determine these motions. Here, we propose the Short Time Fourier Transform (STFT) of the velocity autocorrelation function to resolve vibrational modes in both frequency and time, visualizing how the nuclear modes evolve over the course of the charge transfer process. By examining at each trajectory individually, some noise will be inherent. Examining the entire ensemble collectively will reduce the noise and amplify common modes in the wavenumber dimension. However, trajectories within an ensemble are not all temporally similar. Even if two trajectories exhibit similar vibrational information independently, the dissimilarities in charge transfer time-frame may dilute spectral intensity in the time dimension.

To avoid this, we choose a sub-ensemble of trajectories for each molecule that exhibit similar temporal charge transfer character. To choose these, we compute a pairwise difference squared integral to calculate the "distance" between two trajectories. Let $q_i(t)$ and $q_j(t)$ be two logistic regressions (per Equation 2.49) for trajectories indexed i and j of the same molecule. We define a quantitative measure as the "distance" between trajectories i and j,

$$\epsilon_{ij} = \int_{t_1}^{t_2} \left(q_1(t) - q_2(t)\right)^2 dt \tag{4.1}$$

for t_1 and t_2 defining a given time window considered. Identical functions result in a vanishing integral and deviations results in increasingly larger ϵ. We choose the set of 10 trajectories which mutually give the smallest collective difference integral. Since the Fourier Transform and STFT are linear operations, the average spectrogram can be achieved by averaging the autocorrelations of the trajectories chosen and then performing the transform.

Each molecule simulated contains between 238-272 atoms which leads to over 700 $(3N - 6)$ vibrational modes. The entire triad is a highly conjugated system whose moieties themselves are conjugated subsystems. Naturally, these subsystems have overlapping vibrational spectra wherein spectral intensity from some modes of one moiety could be washed out by that of another moiety. While this indicates such modes are weaker relative to the total system, they still may be relevant to the CT process. Analysis of the entire molecule as a whole may obscure crucial results. By considering where the differences in construction between molecules lie or crucial points in the molecule (linkers), we can isolate the necessary atomic subsets, perform an autocorrelation of their velocities, and then perform the STFT to reveal the underlying dynamics. We then screen the resulting spectrograms for modes with temporal behavior aligned to the charge transfer process. The temporal limits for

screening, t_1 and t_2, are defined by the 5% and 95% CT completion times, respectively, to sufficiently encapsulate the region of interest.

4.1 M1 and M2 Bridge Aryl Substituent Groups

We first consider molecules **M1** and **M2**. Here, the major difference between the two is the number of methyl groups on the *meso* aryl side groups attached to the porphyrin. In Figure 4.1 we present the STFT spectrograms for these two. Immediately noticeable, modes of energies higher than 700 cm^{-1} are of low intensity or absent altogether. Typical vibrational spectra for mesitylene and toluene consist of out-of-plane ring distortions at 700-800 cm^{-1} and C=C stretches at 1500 cm^{-1}. The low energy vibrations in Figure 4.1 are torsional modes with the side aryl rings rotating across the bond linking them to the porphyrin bridge. We confirm this by computing the time series of the dihedral angle of these groups to the porphyrin. Fourier Transform of these time-series reveals spectral intensities centered at the same frequency in the STFT spectrograms (Figure 4.2, top).

Regarding **M1**, two modes at 190 and 425 cm^{-1} appear in all three visualizations, confirming both the torsional dynamics and relevance to the CT process. The dihedral angle here modulates the amount of p_z orbital overlap and therefore has an effect on the local electronic structure of the bridge. A previous calculation gives the first excitation energy dependence on the dihedral angle of an idealized configuration of an isolated diaryl porphyrin, plotted in Figure 4.3. This calculation was performed at the PPP theory level rotating the side benzene rings maintaining D_2 symmetry. The computation reveals only a 0.1 eV difference in excitation energy for dihedral angles between 0° (full p_z overlap) and 90° (no p_z overlap). However, given the amount

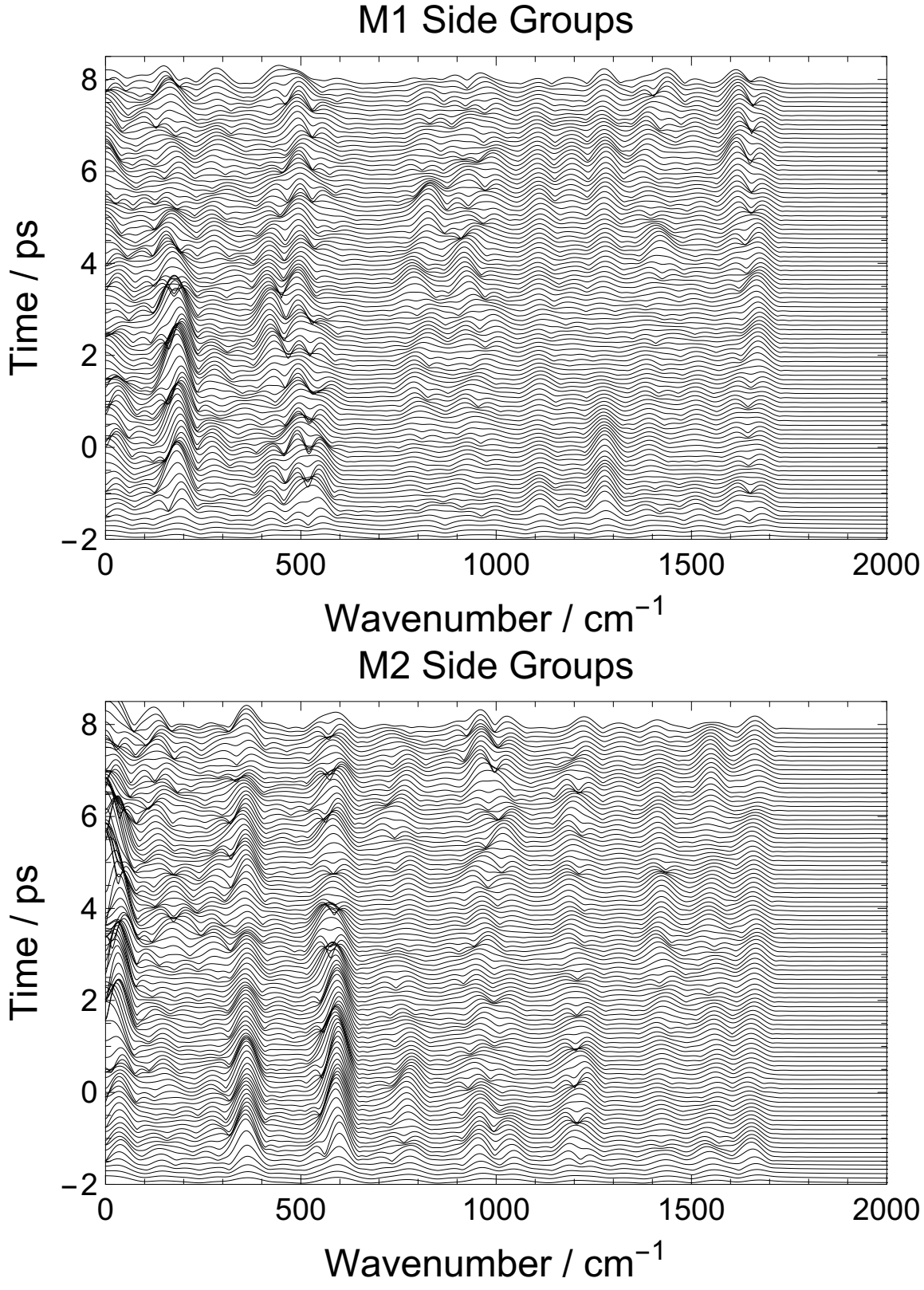

Figure 4.1: STFT spectrograms for **M1** (top) and **M2** (bottom) aryl side groups on the bridge.

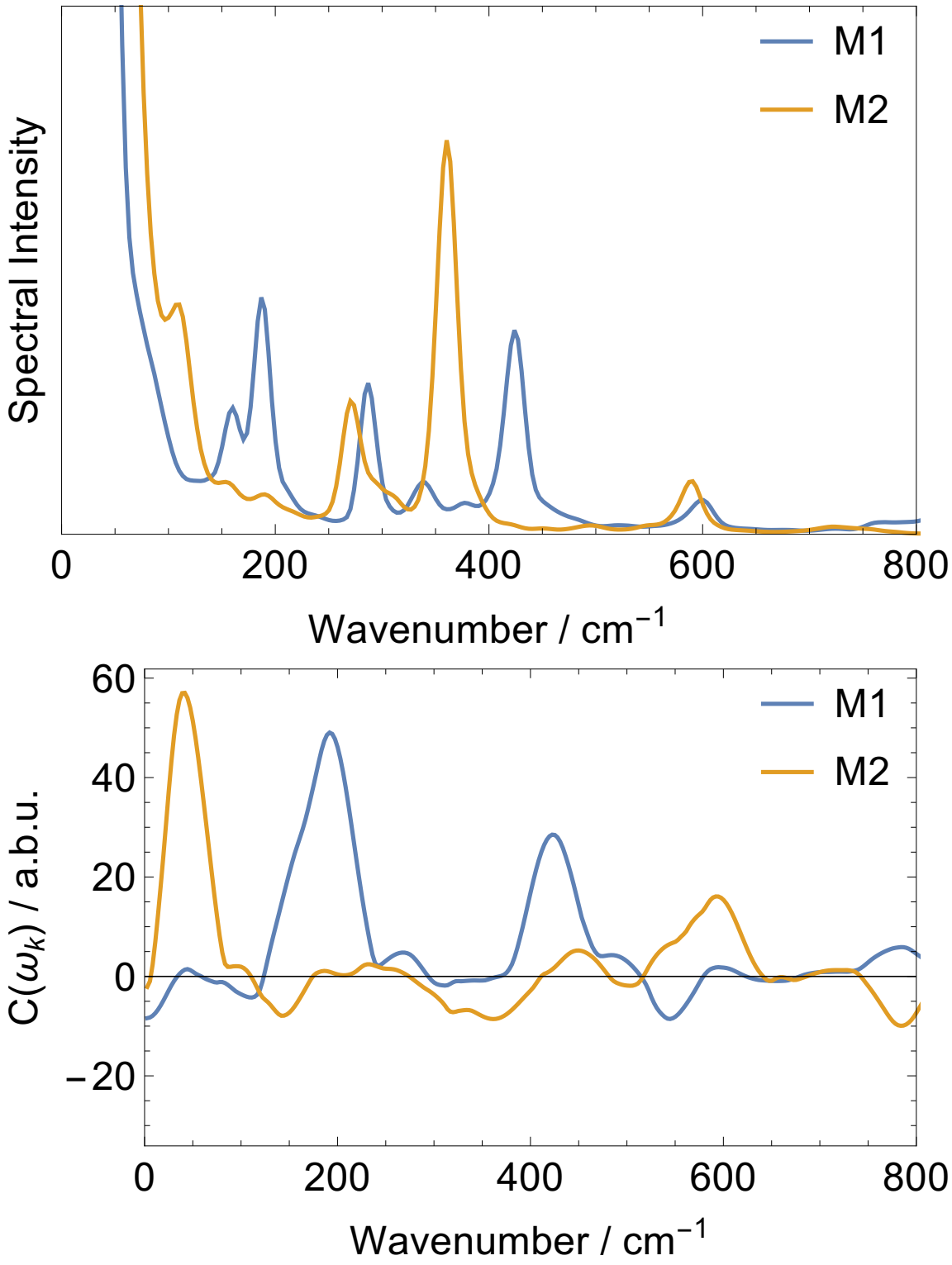

Figure 4.2: (Top) Fourier Transform of the dihedral angle time-series between the bridge and its aryl side groups for **M1** and **M2**. (Bottom) Screened STFT modes for the same subunits in **M1** and **M2**. Both figures here are featureless over 800 cm^{-1}.

of steric hindrance experienced, the accessible angle space is constrained between 72° (± 4°) and the excitation energy accordingly differs by about 6 meV. Thus, for practical purposes, the dihedral angle of these conjugated side groups has little effect on the bridge electronic structure and therefore CT. Despite this, we can conclude there is still some interplay between the CT process and this particular mode: in the context of vibronic transitions, excitation of this mode certainly accompanies the π-π^* electronic excitation.

With less steric hindrance, **M2** has a much stronger response from the 360 cm^{-1} torsional mode; however, this mode is generally active throughout the simulation time and therefore does not pass screening. Conversely, the strong 600 cm^{-1} mode correlates to CT but is not prominent in the dihedral spectrum and therefore this is likely to be some low energy ring distortion. Charge transfer in **M2**, then, seems unaffected by torsional motions of the aryl side groups. We make the same arguments as with **M1**; these modes are activated during the initial excitation but do not play an essential role in electronic relaxation of the system into a charge transfer state.

4.2 M3 and M4 Bridge/Acceptor Aryl Linker

We next consider **M3** and **M4**. The bridge side groups here are not π active and, similar to the previous molecules, do not have a significant impact on the electronic structure of the bridge directly. From an experimental standpoint, aliphatic substituents like these are included to control the solubility of the molecule. However, as pointed out by Bahr, the longer chains in the β positions sterically interfere with and force the linker aryl groups into perpendicular orientations[39]. Recent DFT computations on these triads showed that optimized structures have dihedral angles

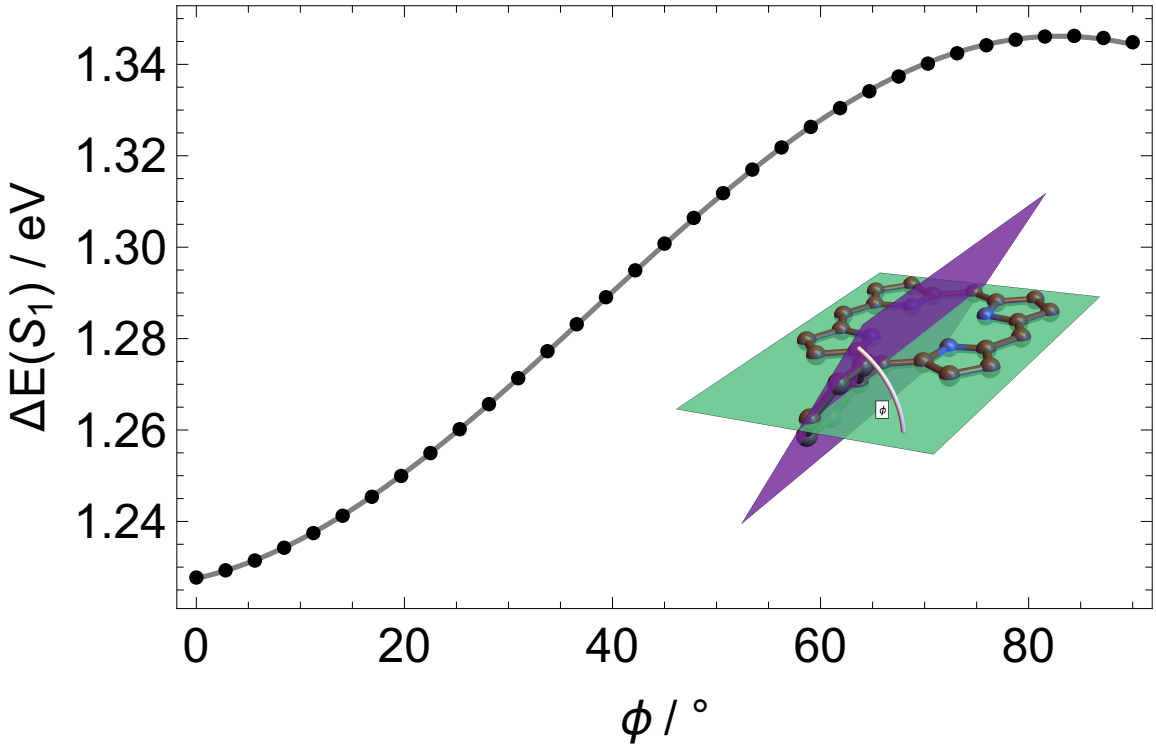

Figure 4.3: First excitation energy of isolated diaryl *meso* substituted porphyrin dependence on dihedral angle, ϕ. Inset is a rendering of a planar porphyrin with one *meso* aryl substituent. Hydrogens have been removed for clarity. ϕ indicates dihedral (small) angle between porphyrin and benzene planes. Points correspond to calculated energies. Gray curve corresponds to a curve fit. Fit: $\Delta E(\phi) = E_{\max} - \delta E * \cos^2(\phi + \gamma)$, $E_{max} = 1.35$ eV, $\delta E = 0.12$ eV, $\gamma = 0.12$ (radians).

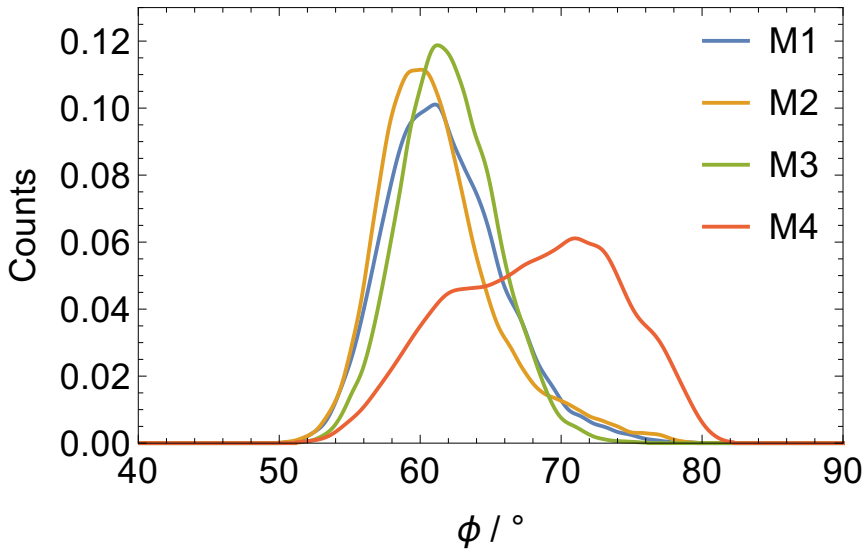

Figure 4.4: Distribution of dihedral angles sampled by the aryl linker between the bridge and acceptor moieties during dynamics for **M1-4**.

that agree with this notion[82]. The dihedral angle of these linkers to the porphyrin should have a much larger impact on CT the side aryl groups. Increased orbital overlap between porphyrin and these linkers extends the molecular π orbital towards the fullerene acceptor and therefore increases the electronic interaction between the two subsystems. Figure 4.4 plots a smooth histogram distribution of the dihedral angles sampled by these linkers (for molecules **M1-4**). Curiously, our dynamics results show a contrary trend. The linker in **M3** is constrained to a smaller angle space, but does not sample angles perpendicular to the porphyrin and instead stays in the region of just over 60°. The linker in **M4** accesses a wider angle space and, although sterically allowed to have a more planar angle, tends to remain closer to perpendicular to the porphyrin plane. However, our dynamics indicate aryl linkers in **M1** and **M2** oscillate around 61°, the ground state orientation observed in DFT calculations[82].

STFT spectrograms for these subunits are found in Figures 4.5. The low frequency modes (390 and 600 cm^{-1}) are muted in **M3**. These frequencies appear in the dihedral angle times-series analysis (Figure 4.6, top) and we define both torsional

modes. When screened, the former mode is of no interest, but the latter mode shows as strongly anticorrelated. Low spectral intensity in STFT is consistent with the narrow distribution of angles sampled. We argue here that as electron density travels through the linker, the angle with the bridge plane is relatively locked in place. The mode at 950 cm^{-1} has strong response at early time then slowly decays. This mode accompanies the initial photoexcitation but diminishes once CT begins. Furthermore, it does not appear in the dihedral analysis and therefore we define this as a ring distortion mode. Of lower overall intensity, the 1010 cm^{-1} mode shows some response following excitation, then increased activity during CT, and finally decaying after that. Weaker still, the 1570 cm^{-1} mode shows as highly correlated. We determine these last two modes as ring vibrations as they do not appear in the dihedral angle analysis.

Spectra for the aryl linker in **M4** reveal the presence of similar modes to that of **M3**. The 390 cm^{-1} mode appears anticorrelated but has a far too weak intensity to have significance. Contrary to **M3**, the 600 cm^{-1} mode is not only highly active, strengthens the anticorrelation. Intense activity of this mode before CT corroborates with large angle space the wide range of angles sampled; however, torsion is restricted once current flows through the linker. Temporal shape of the 950 cm^{-1} mode in largely mimics that of the 600 cm^{-1} mode in **M3** but with larger intensity: this mode is active at photoexcitation, but absent after that. Two final vibrational modes at 1120 and 1600 cm^{-1} are relatively weak in the STFT spectrogram, however, still correlated to CT.

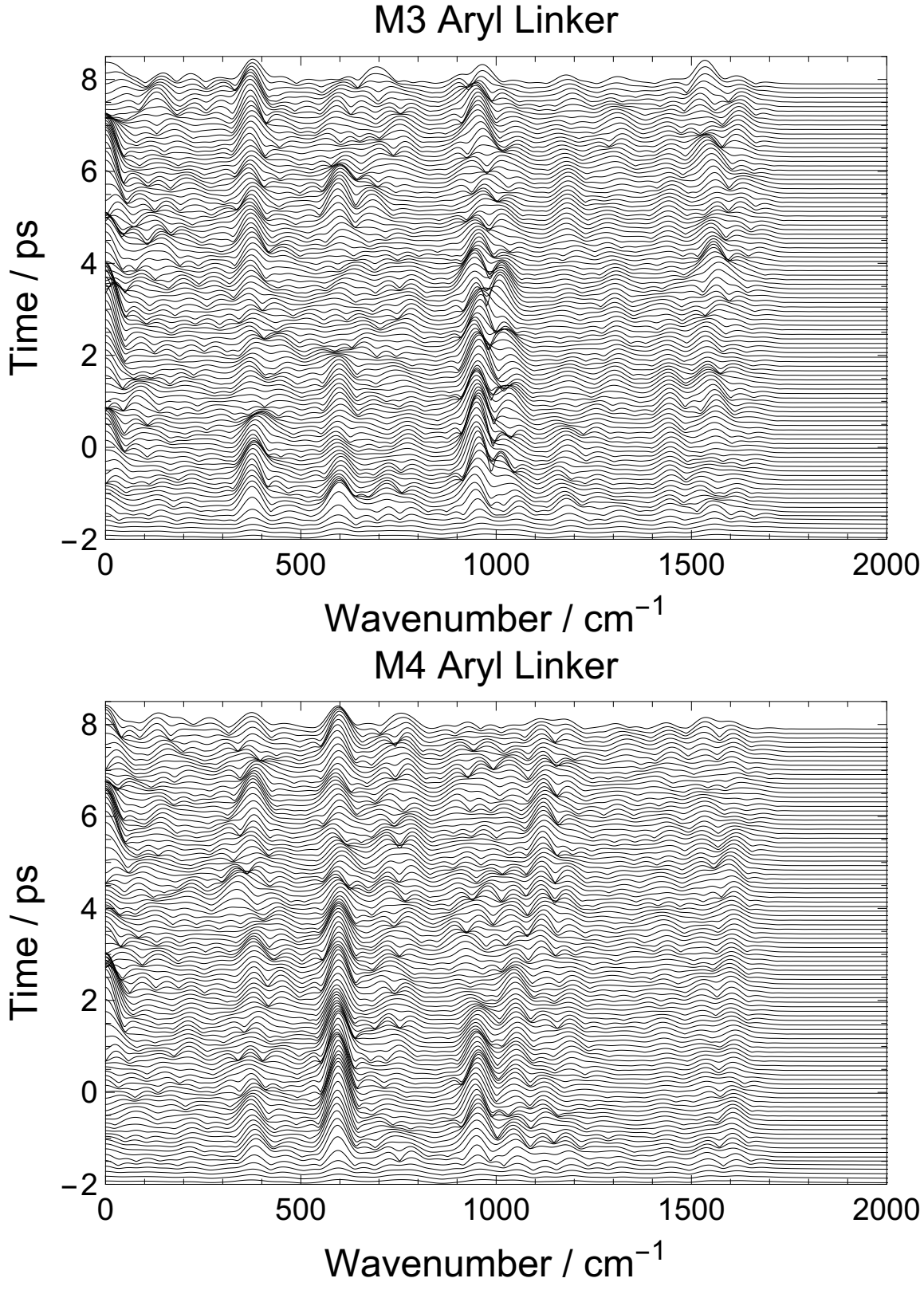

Figure 4.5: STFT spectrograms for aryl linker between bridge and acceptor in **M3** (top) and **M4** (bottom).

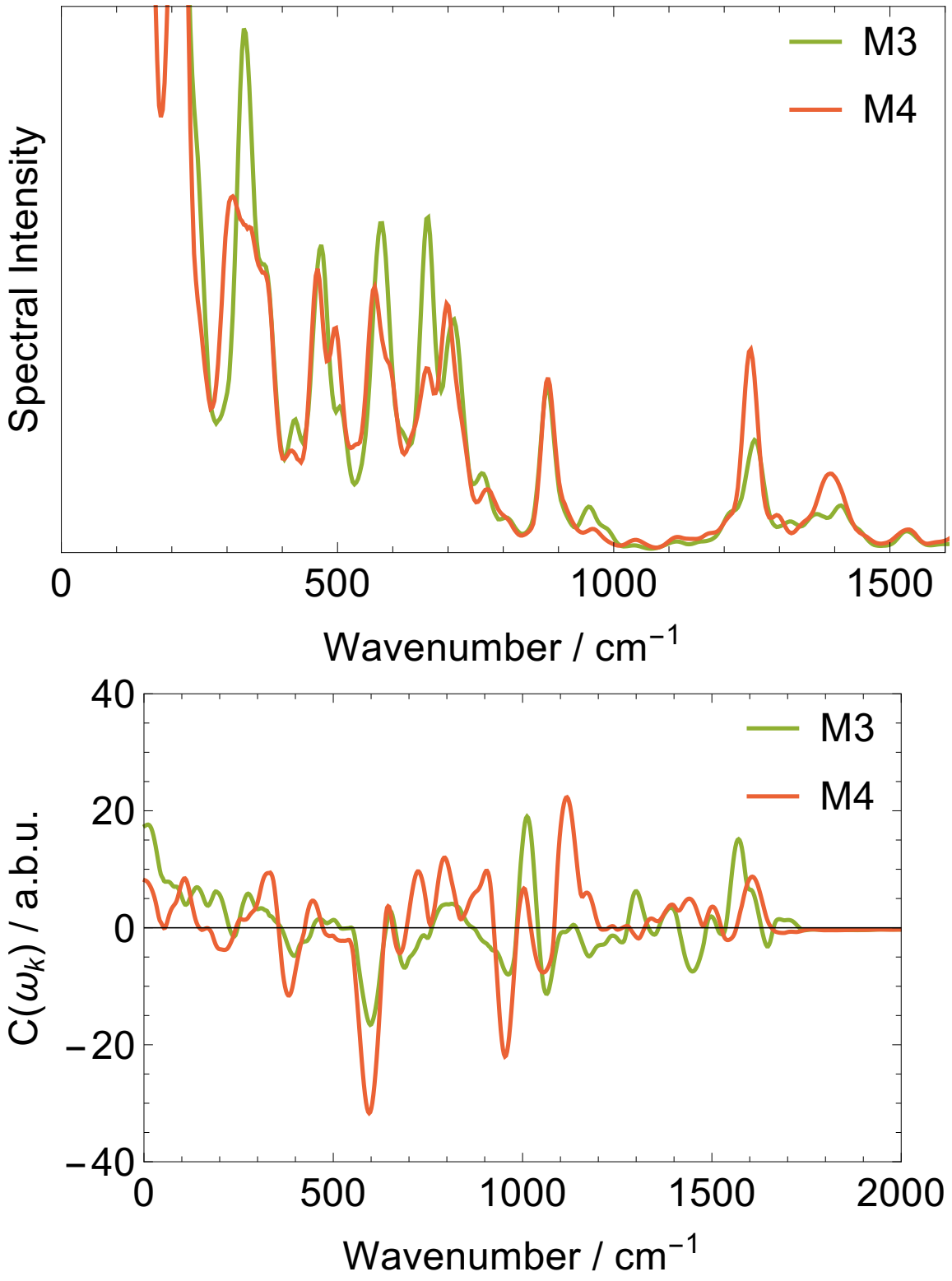

Figure 4.6: (Top) Fourier Transform of the dihedral angle time-series between the bridge and the aryl ring which links bridge and acceptor moieties for **M3** and **M4**. Figure is featureless above 1500 cm^{-1}. (Bottom) Screened STFT modes for the same subunits in **M3** and **M4**.

4.3 Donor/Bridge Amide Linker

Amide linker STFT spectrograms are plotted in Figures 4.7 and 4.8. Prevalent for all four molecules are intense features at 700, 1000, 1200, and 1300-1400 cm^{-1}. Upon screening these modes in time, only the 700 and 1200 cm^{-1} appear to play significant roles. Vibrational computation on formamide suggests the 700 cm^{-1} mode corresponds to a twisting motion along the C-N bond. In **M1** and **M2**, this mode is sporadically activated and deactivated. However, there is a stark difference with **M3** and **M4**, recalling that direction of this linker is reversed between the two. According to rectification studies, the amide favors current travelling from the nitrogen side to the carbon side[48]. Thus, the linker in **M3** has favorable orientation to allow electron density to pass from the donor to the bridge. In our studies, this mode correlates extremely well with CT to the acceptor. Naturally, the net positive charge in the bridge will begin to attract electron density from the donor. In **M4**, the amide is in the unfavorable orientation and we show here that the 700 cm^{-1} mode is deactivated during CT from bridge to acceptor. It is reasonable to believe that this particular mode is highly sensitive to the direction of current passing through the amide. The mode in the region of 1200 cm^{-1} (Amide III: C-N stretch, C-N-H bend) is universally deactivated during CT in all four molecules indicating this mode is not conducive to CT. In contrast to the findings published in Ref. [50], we see no activity in amide modes above 1500 cm^{-1}.

4.4 Bridge and Acceptor

We turn our attention now to the porphyrin bridge moiety. STFT spectrograms are found in Figures 4.10 and 4.11. Mode screening for the four bridges are found

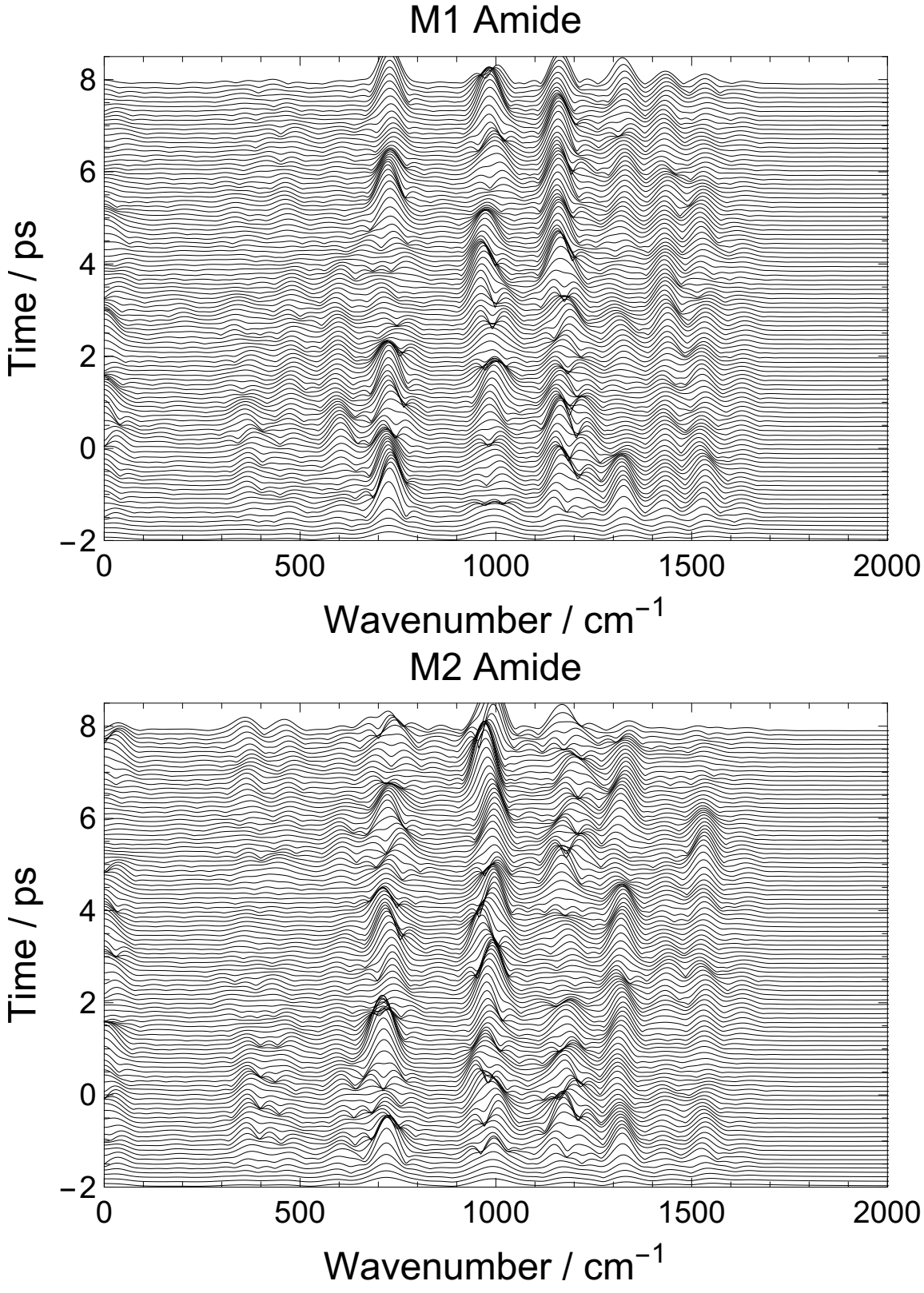

Figure 4.7: STFT spectrograms for the amide linker in **M1** (top) and **M2** (bottom).

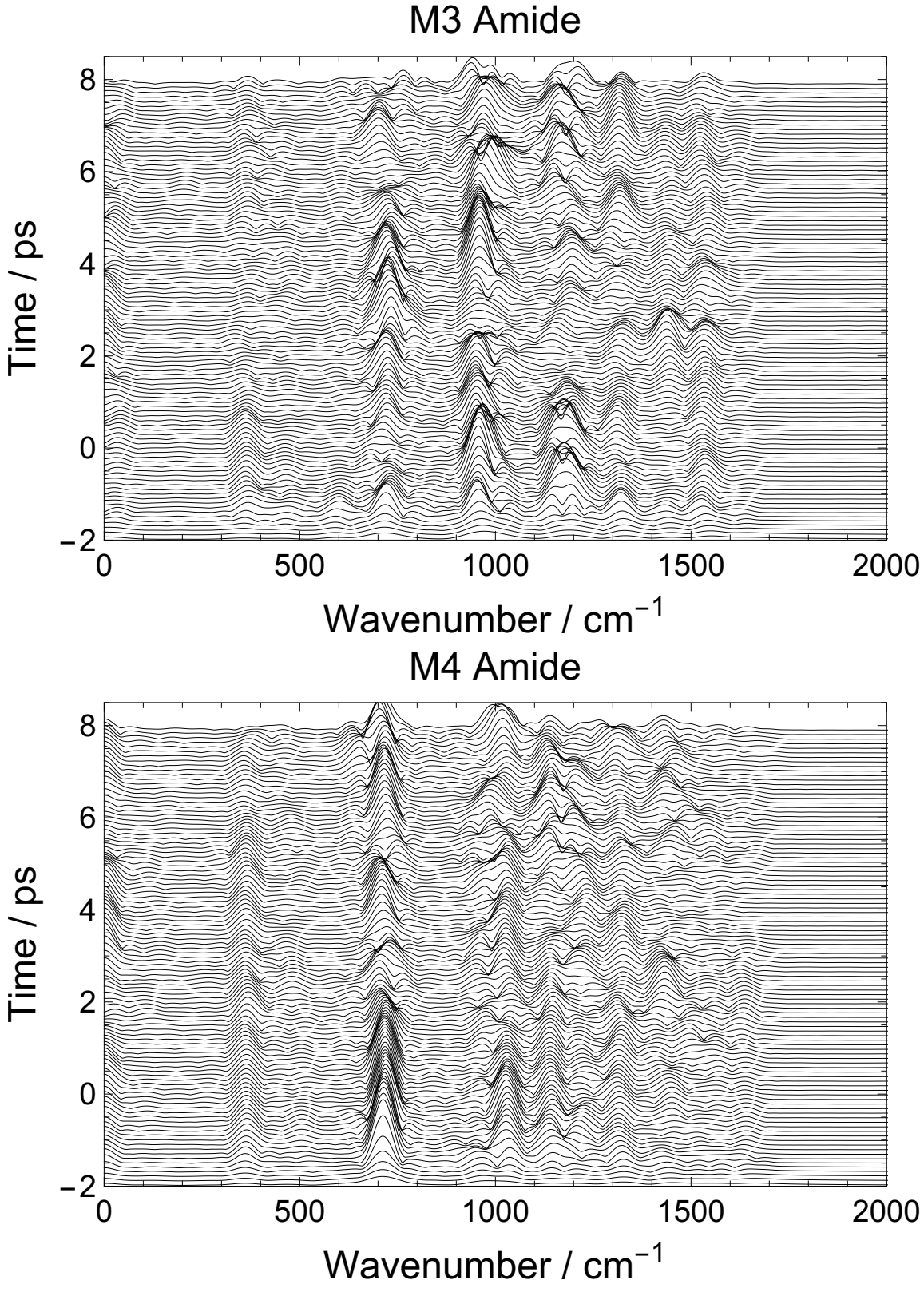

Figure 4.8: STFT spectrograms for the amide linker in **M3** (top) and **M4** (bottom).

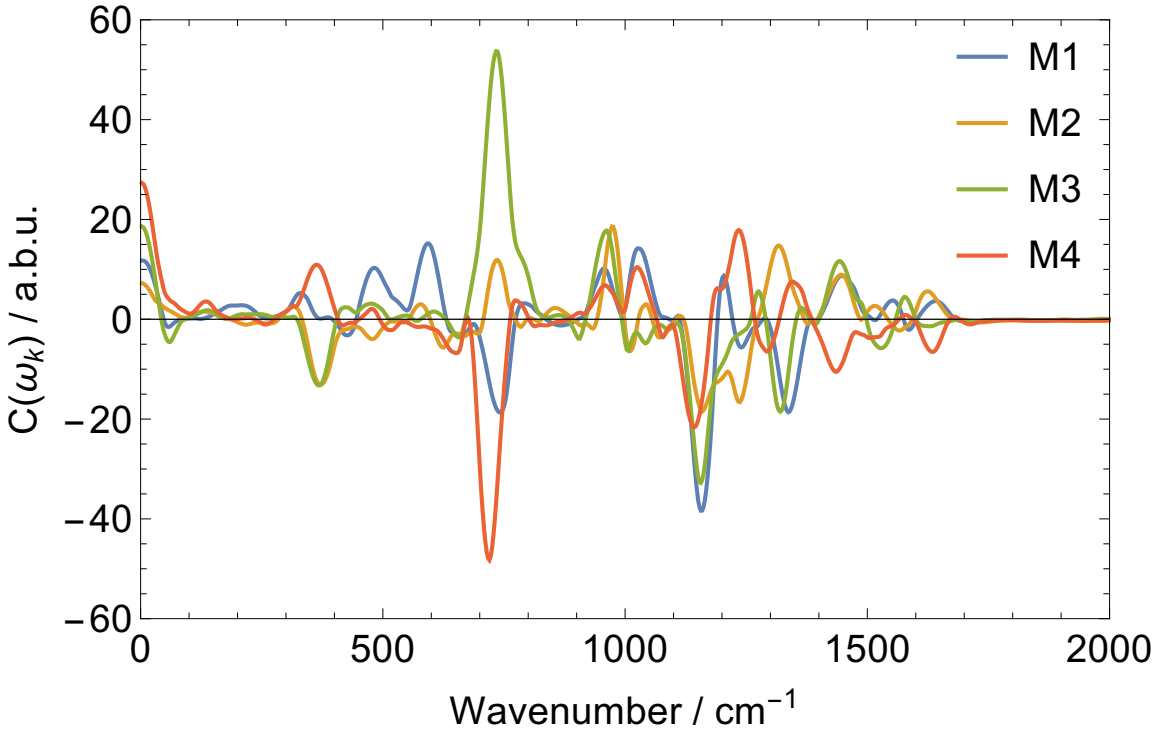

Figure 4.9: Screened vibrational modes for the amide linker in **M1-4**.

in Figure 4.12. What is immediately obvious between the four spectrograms is the general absence of features under 500 cm^{-1} and above 1500 cm^{-1}. This featureless region is line with ground state experimental and computational vibrational studies of water solvated free base (and select derivatives of) porphryrin[83, 74]. **M1** and **M2** exhibit some low energy modes at 500 and 600 cm^{-1}, respectively, that strongly correlate with CT. These macromolecular modes are note present in **M3** or in **M4**, thus we can speculate these modes resemble macro-ring twisting with nodes at the *meso* position. Such modes would alter the aromaticity of ring (breaking planarity, and therefore having a marked impact on electronic structure) but would be far more restricted by the heavier alkyl β substitutions in **M3** and **M4**. The 720 cm^{-1} mode, likely some alternative ring deformation, is active in all four molecules and is anticorrelated with CT, excluding **M3** where it is uncorrelated. The intensity of this mode is oscillatory throughout the dynamics, activated universally at the time

of excitation then progressing with seemingly inconsistent behavior.

In the 950-1050 cm^{-1} vibrational region, **M1** and **M4** show CT correlation while **M2** and **M3** show anticorrelation with spectral intensities differing between all four molecules. **M1** solely sees a strongly anticorrelated mode at 1110 cm^{-1}. This mode in both **M1** and **M2** is activated at excitation then decays in time; however, the mode reemerges after CT in **M1**. The same frequency in **M3** and **M4** is almost entirely nonexistent lending to the idea this mode is ring torsion with nodes at *meso* carbonds. Most nuclear displacement would occur at the pyrolle rings which would be inhibited by heavy β substituents. Finally, a mode in the mid 1400 cm^{-1} range shows as correlated in **M2**, anticorrelated in **M3**, and decorrelated in **M1** and **M4**. These modes are of low intensity overall and temporal behavior is universally noisy, perhaps only coincidentally aligned (misaligned) with CT in **M2** (**M3**).

We finish here with examination of the acceptor moiety. STFT spectrograms for the fullerene dynamics can be found in Figures 4.13 and 4.14. The mode screening for the acceptors can be found in Figure 4.15. Contrary to the bridge counterparts, the acceptors have much more activity in the low-frequency range and relatively subdued activity at higher frequencies. In general, the spectrograms reveal that most of the dynamical activity is in breathing or other macromolecular modes as opposed to local carbon-carbon stretches. Density Functional Theory (B3LYP level) studies of C_{60} and C_{60}^{-} indicate ionic forms have larger overall radii and distortion of the spherical geometry versus the neutral molecule[84]. We expect to see breathing modes accompany electronic charging of this moiety. Indeed, all four molecules show some level of correlation to CT with a mode between 200-250 cm^{-1} with **M4** having a strong correlation. A multitude of other modes show strong correlation in either direction (e.g., 500, 700, 1000-1200, 1500 cm^{-1}) but are not consistent between systems.

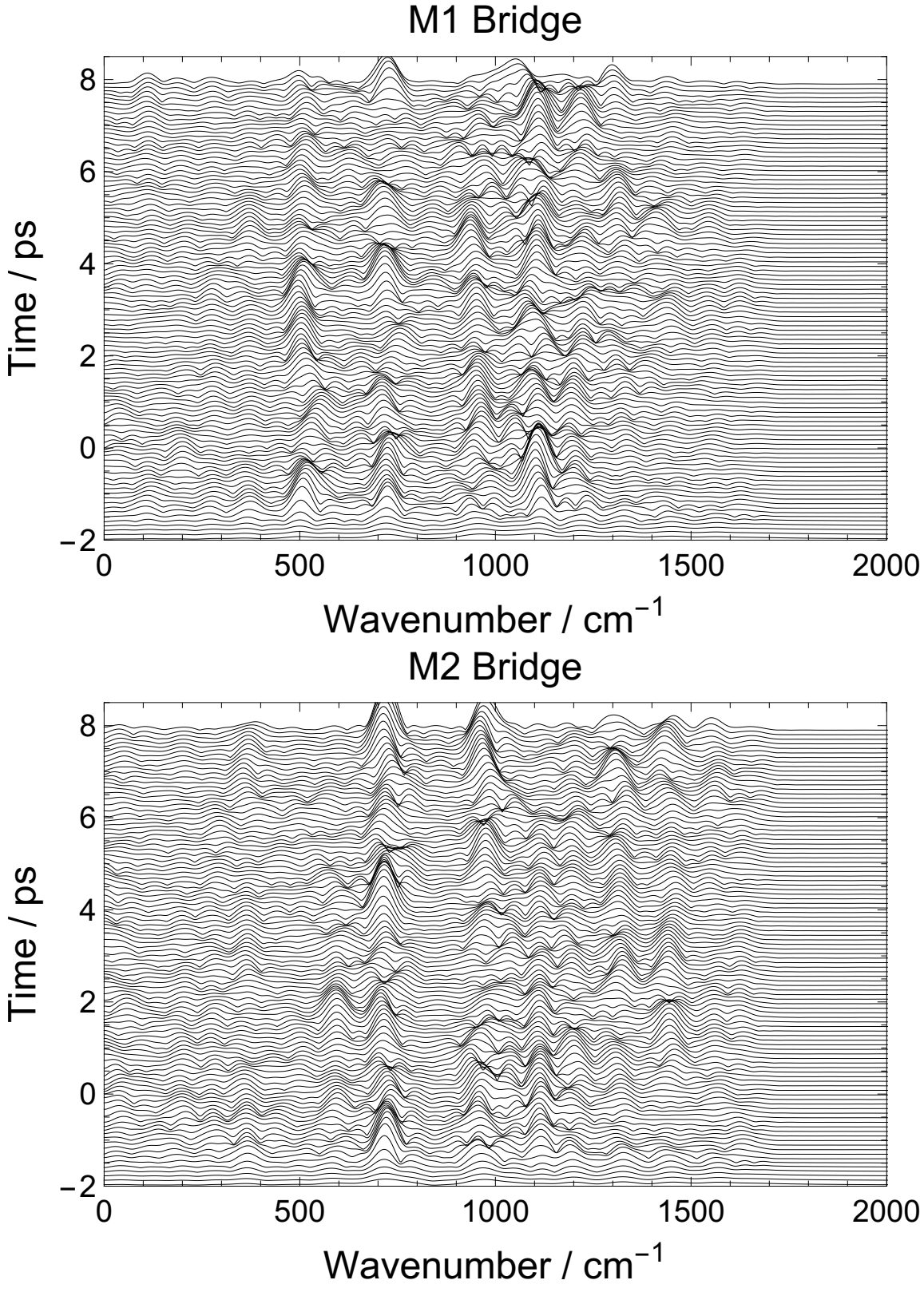

Figure 4.10: STFT spectrograms for the porphryin bridge in **M1** (top) and **M2** (bottom).

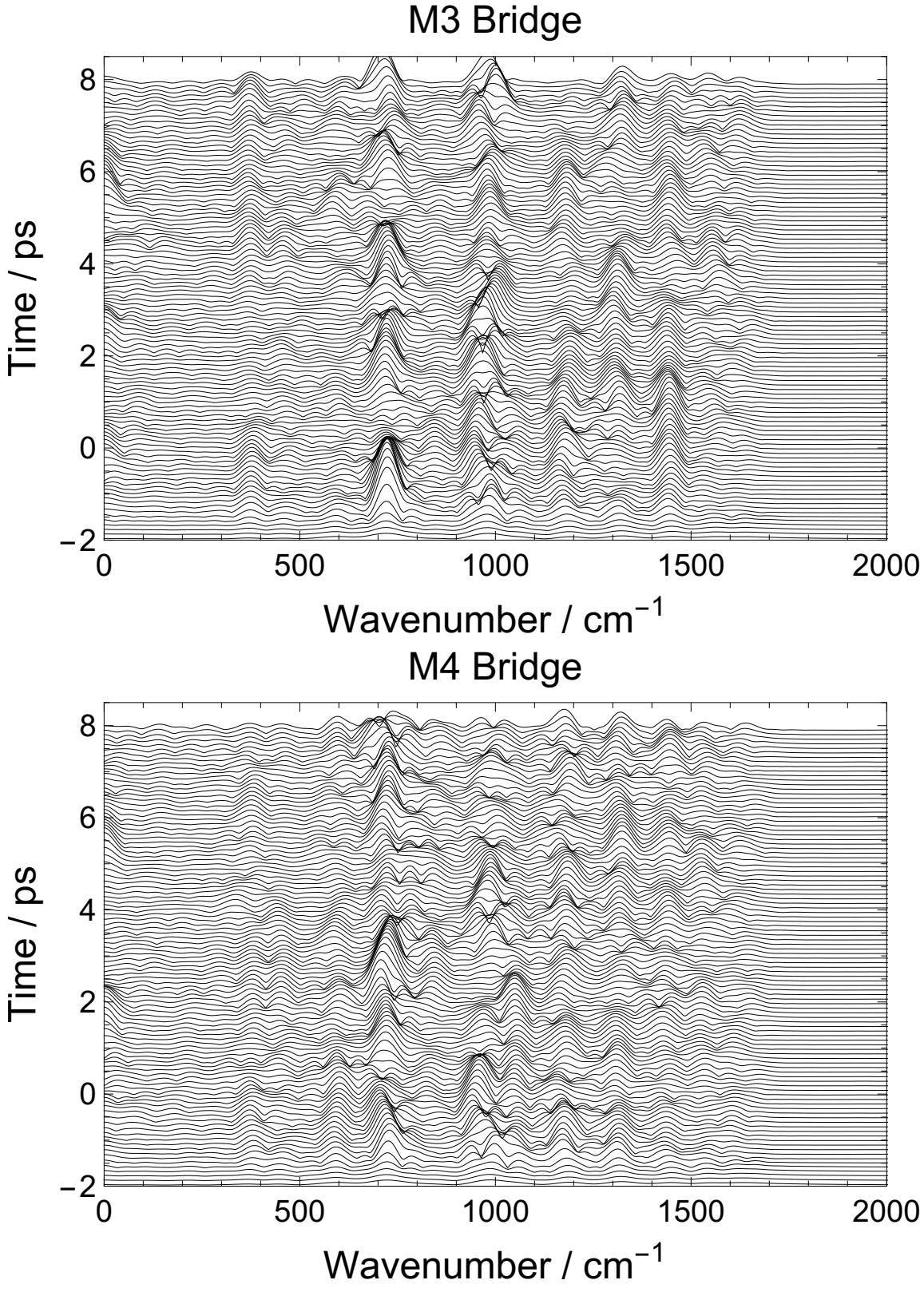

Figure 4.11: STFT spectrograms for the porphryin bridge in **M3** (top) and **M4** (bottom).

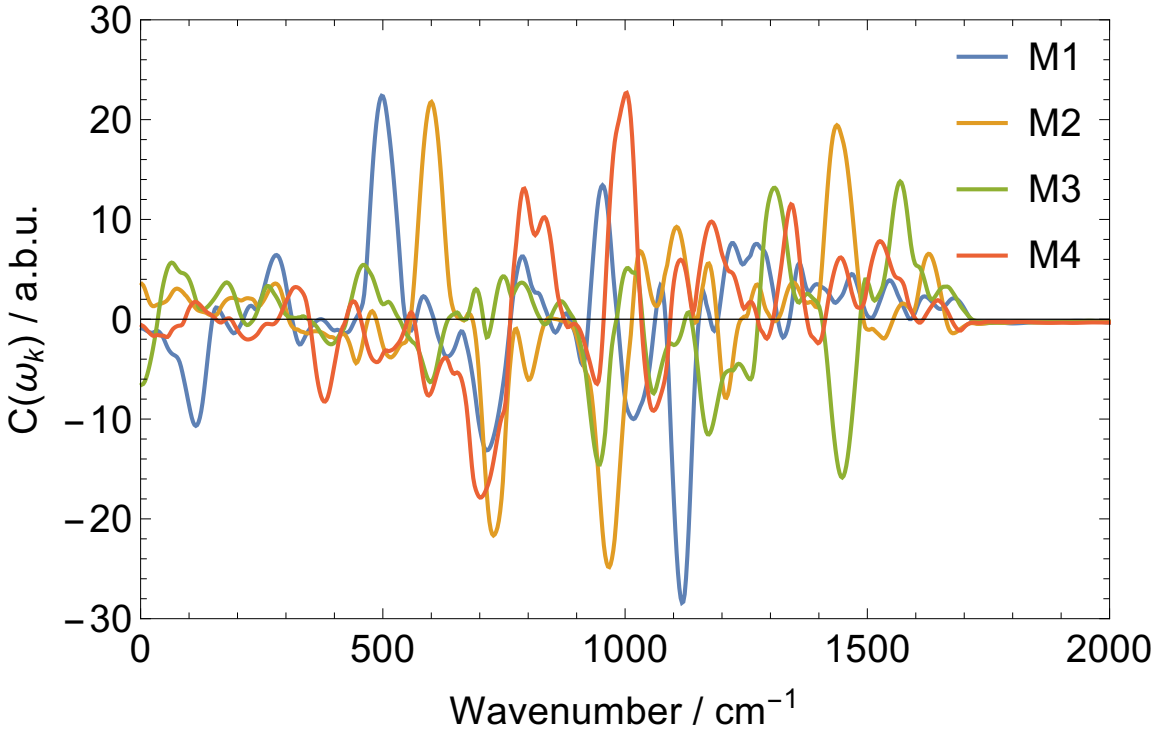

Figure 4.12: Screened vibrational modes for the porphyrin moiety in **M1-4**.

Vibrational energy within the porphyrin is primarily distributed amongst higher frequency modes, which is likely associated with the large electronic excitation energy pumped into a localized region. The fullerene, conversely, has most of its vibrational energy distributed among low energy modes. Outside of the modes mentioned above, no other modes seem to behave consistently across these triads, at least temporally. The inconsistency in vibrational behavior could be due to a few reasons. Our subset for STFT analysis chooses only 10 trajectories, which may be an insufficient number for analyzing larger structures. Analysis of linker and bridge side groups involved at most 12 atoms, limiting the total number of normal coordinates, whereas the same analysis of full moieties requires the consideration of 30 atoms and upwards. A large number of degrees of freedom gives rise to more vibrational modes that are both energetically similar (possibly degenerate) and energetically accessible. Alternatively, the dynamics of the triads depends heavily on the initial conditions, and this is easily

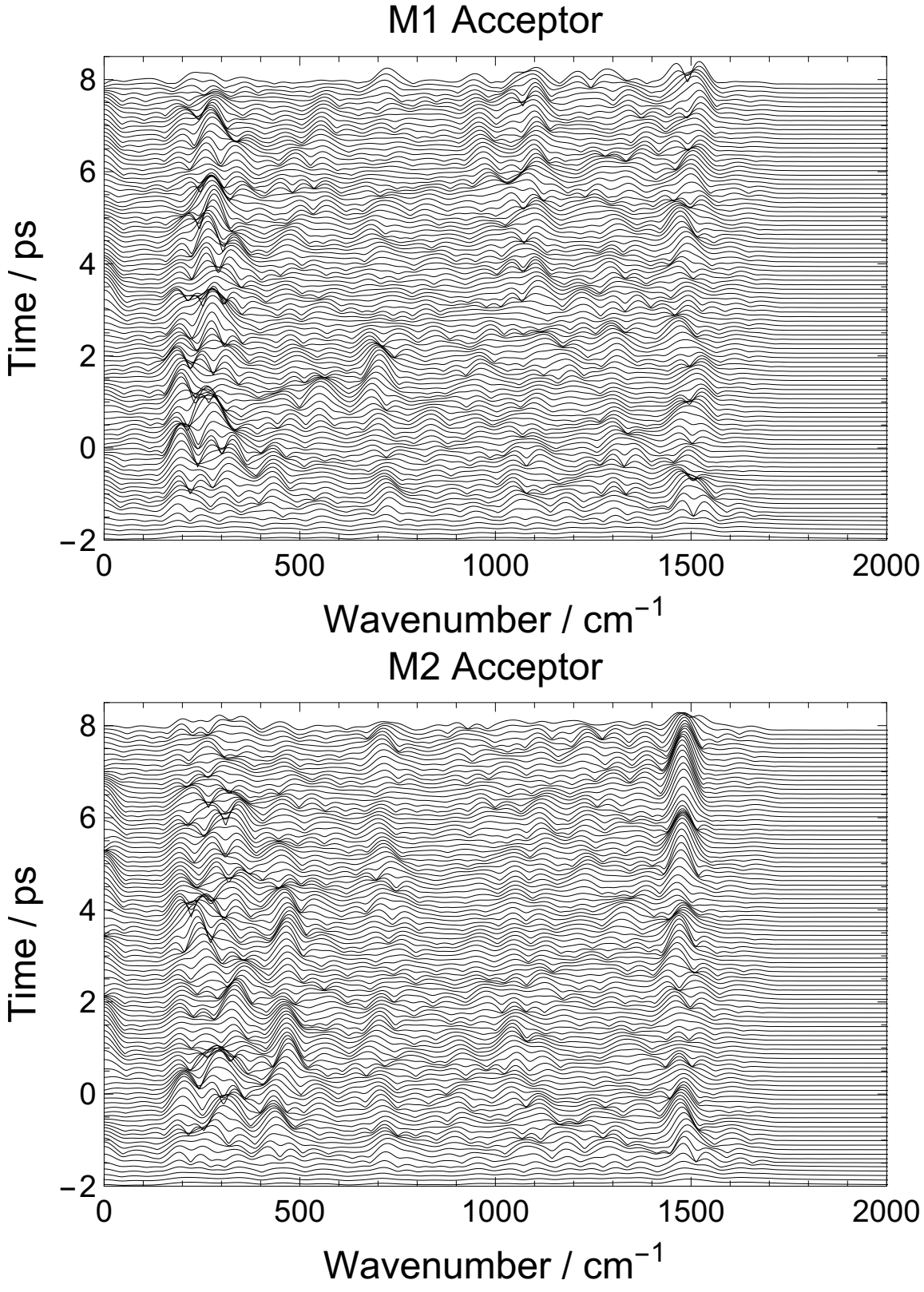

Figure 4.13: STFT spectrograms for the fullerene acceptor in **M1** (top) and **M2** (bottom).

seen in Figures 3.3 and 3.4. Variations in the geometries necessarily result in varying excitation energies and excited electronic structures which in turn give diverse charge transfer characteristics. Therefore, averaging more trajectories would either reduce the vibrational noise resulting from the increase in accessible modes or, if noise is not reduced, give further credence to the sensitivity to initial conditions.

An additional effect of the dynamical evolution of force constants with the electronic structure is the blue- or red-shifting of the vibrational modes, most prominently in the amide STFT spectrograms. Within the harmonic oscillator approximation, the frequency of a mode is proportional to the square root of the force constant which is modulated by the π bond order (off-diagonal elements of the site basis density matrix). The most extreme case of this shift is around 50 cm^{-1} seen in the 1000 cm^{-1} mode in Figure 4.1 (bottom) whereas this effect is less pronounced in other spectrograms. While this shift should be an indicator of fluctuating electronic state, our choice of time-frequency analysis and window function width in this work gives a frequency uncertainty of about 28 cm^{-1} and, consequently, we reserve detailed discussion of the effect in the present results. The trade-off between uncertainty in temporal or frequency resolution is unavoidable. However, the wavelet transform technique of time-frequency analysis does provide better resolution than STFT and future analyses will benefit from this approach[71, 68, 85].

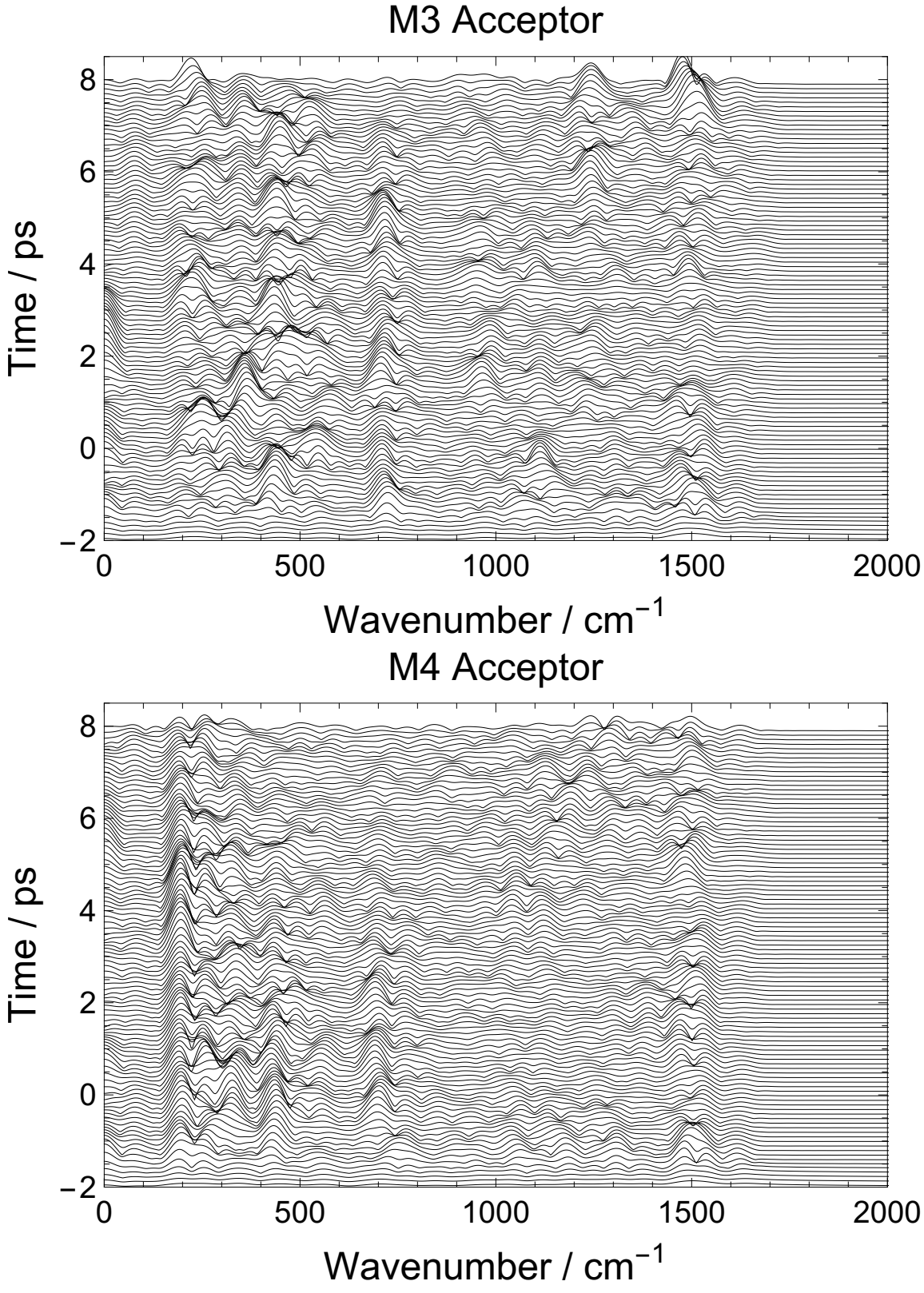

Figure 4.14: STFT spectrograms for the fullerene acceptor in **M3** (top) and **M4** (bottom).

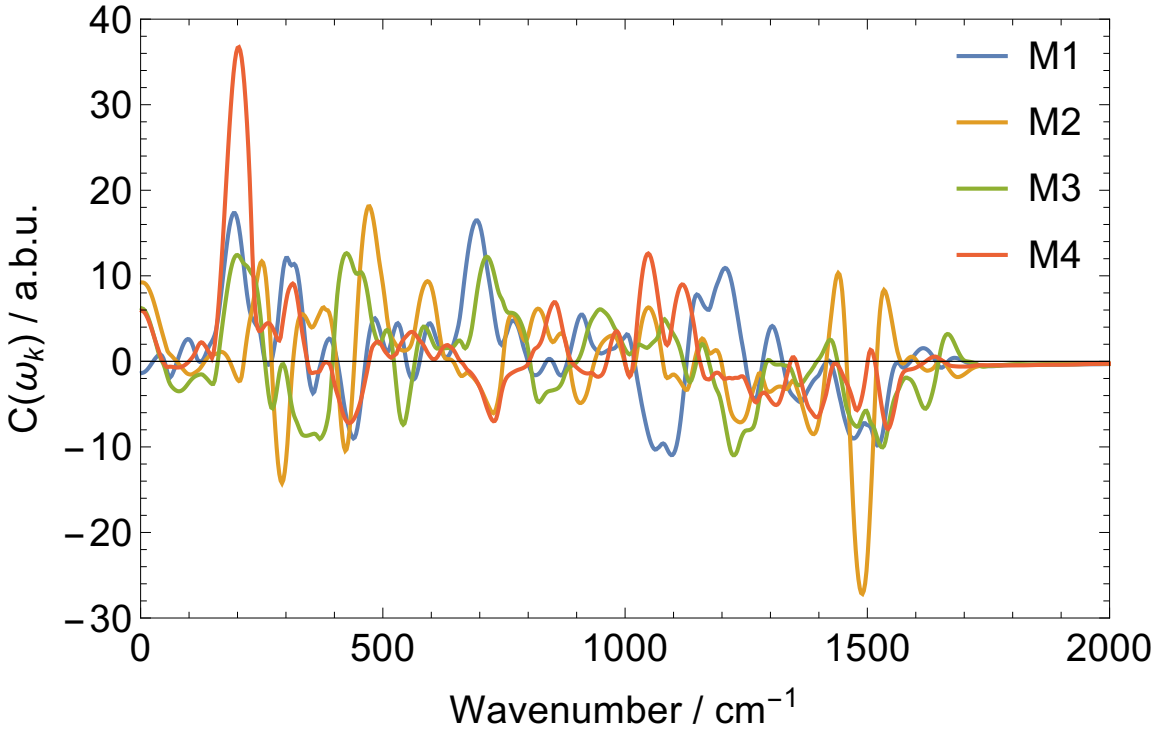

Figure 4.15: Screened vibrational modes for the acceptor moiety in **M1-4**.

Chapter 5

Conclusion

We have presented here the results of mixed quantum mechanical/molecular mechanics simulations modeling electron and nuclear dynamics in four Donor-Bridge-Acceptor triads, each with slightly differing bridge constructions. By taking a statistical approach, we ensure our simulations are representative of the average charge transfer behavior of these systems. Our model can reproduce experimentally accurate excitation energies for a localized excitation on the porphyrin bridge. Electron population analysis indicates initial charge transfer from porphyrin bridge to fullerene acceptor occurs at the picosecond scale. While we systematically overestimate charge transfer rates in **M1** and **M2**, we reasonably reproduces rates in **M3** and **M4** when compared to experimental findings[43].

We further provide a time-resolved vibrational power spectrum which characterizes the nuclear vibrations occurring in our simulations. Our mode filtering technique provides a qualitative correlation of vibrational modes to the initial charge transfer process. By considering a subset of atoms from each triad, we can pinpoint which

nuclear motions have the strongest coupling to the electronic states. We find that conjugated aryl side groups on the porphyrin undergo dihedral rotations (lower energy) over ring breathing and ring distortions (higher energy). Despite a strong response to CT, we show these motions have very little effect on the electronic structure of the porphyrin at large. Conversely, aryl rings which link the bridge and acceptor units are shown to have little torsional behavior, and instead, nuclear motions take the form of ring distortions. Although this work focuses on charge migration from the bridge to acceptor, we still analyze the vibrations of the amide linking bridge and donor units. Here we find with **M3** and **M4** a torsional mode that is sensitive to the direction of electron flow. Finally, our analysis of bridge and acceptor moieties sees very little commonalities in nuclear dynamics between triads specific to the CT, despite the level of similarity in chemical structure. Nuclear energy is broadly distributed among high energy ring distortion modes in the bridge and low energy breathing modes in the acceptor.

Within our considerations in this work, we limit discussion to a qualitative relationship between vibrational modes and the electron transfer process without necessarily indicating dependency. Exploration via full quantum mechanical models would give detailed insight into the interplay between electronic and nuclear dynamics. Modes which show strong correlation may drive the CT process. We interpret anticorrelated modes here as having no activity during CT (perhaps even deactivated by CT). Recent experiments show the reverse occurring in systems with acetylide bridges. Selective vibrational excitation effectively disables CT in these DBA systems entirely[86].

Several quantum models implemented by Bittner *et. al.* study OPV's and charge transfer properties[87, 81]. Such models consider electronic states accompanied by

a small set of vibrational modes for acceptor, bridge (if included), and donor units and can reasonably recreate certain photophysical phenomena seen in experiment. With the vibrational analysis presented here, we begin the work to parameterize these models in a more robust sense, expanding the number of modes considered and differentiating the modes for each molecular unit. Furthermore, we can separate modes which have a marked effect on the electronic structure of the system versus modes which can be relegated to a phonon bath. What remains missing is the electron-vibration coupling parameter which is certainly a non-trivial computation.

Future Work

Marcus postulated that for larger ions, the change in electric field associated with charge transfer is smaller as the new charges are distributed over a larger volume of space[3]. As a result, there is less energy associated with the reorganization of solvent molecules, but the local system must undergo a more extensive rearrangement in response to new charges. For this work, we assumed solvent response would be minimal with respect to this charge transfer process and therefore excluded them. This not only eases computation but allows us to observe the vibrational response of the triads without interference.

We do, however, recognize the importance of inclusion of solvents. The addition of nuclear degrees of freedom provides an energy sink that can pull vibrational energy out of the system. This will either reduce some vibrational noise (modes associating with CT will remain active) or restrict the system from specific modes (torsional and breathing, for example) due to crowding. Polar molecules would stabilize any charge that emerges and the associated drop in energy may provoke more charge transfer

than the 0.35 q that we see. Inclusion of this interaction proves to be a non-trivial task in the TINKER software, however, as there is no explicit interaction term in the PPP Hamiltonian with solvent dipoles or external electric fields. Such interactions are currently only indirect: MM force fields of the solvent interact with those of the system. Direct interaction of the electronic structure with solvent molecules can be done if the choice of solvent has π conjugation. This would entail quantum mechanical treatment of π electrons in explicit solvent molecules and while this will increase computational cost, it is still a feasible computation in the TINKER software.

In the previous chapter, we briefly discuss an increase in ensemble size for the purpose of reducing vibrational noise and producing more consistent mode correlation from STFT. In our limited sample size here, choosing temporally similar trajectories required that we poll around the average case (the most populous region in the rate distribution). Vibrational analysis of this set allows us to qualitatively discuss modes which are relevant to CT. However, an increase in the ensemble size would give rise to even more trajectories which exhibit slow or fast CT. We can then analyze the extreme case sub-ensembles and investigate vibrational mode involvement in the retardation or hastening of CT.

The overall CT behavior for each trajectory is highly sensitive to initial configuration, and an exhaustive examination over every degree of freedom would require enormous computation effort. A large enough ensemble size would also present a sufficient training data set for the development of a machine learning model. Such a model could determine whether the charge transfer behavior in these triads is predictable and if so, identify key features (initial geometries, dihedral angles, excitation energies, and vibrational modes) that lead to optimal charge transfer properties.

Appendix A

Codes

A.1 Added Keywords

`EXCITES [int]`	-	Perform CI-S singlet excitations. Integer indicates the excitation number.
`EXCITET [int]`	-	Perform CI-S triplet excitations. Integer indicates the excitation number. (Experimental)
`TDHF`	-	Use Time-Dependent Hartree-Fock and density matrix evolution.
`TDHFCR [real]`	-	Sets the screening radius for Yukawa-like interaction in Hartree-Core calculation.
`TDHFDEBUG`	-	Increase verbosity of TDHF output.
`USEURHF`	-	Use unrestricted Hartree-Fock method. By default, THDF will use restricted HF. (Experimental)
`PRINTEVERY [int]`	-	Output frequency of TDHF. Integer defines number of iterations between TDHF data outputs.

PROBCURR	-	Compute the probability current at each quantum-active site. (Experimental)
RESUME	-	Import data from previous calculation and resume from there. (Experimental)

A.2 cicalc.f

```
!
!     ##########################################################
!     ##                                                      ##
!     ##  subroutine cicalc  --  Calculates CI                ##
!     ##                         energies and states          ##
!     ##                                                      ##
!     ##     Originally written by Bittner for an older       ##
!     ##     version of Tinker, Rewriten by Kush Patel        ##
!     ##     for Tinker 7.1.2.                                ##
!     ##                                                      ##
!     ##########################################################

      subroutine cicalc

      use sizes
      use atoms
      use bndstr
      use civars
      use iounit
      use piorbs
      use units
!--------------------------------
```

```
!     to see if TDHF is requested
      use tdhfvars
!-----------------------------------

      implicit none

!     cicalc variables
!     index variables
      integer i,j,k,l,m,ma,iip,iorb,jorb,iir
      integer ih,ie

      integer nv,nc,nci,nci_max,nl,nu
!        nv  - num valence orbitals
!        nc  - num condunting orbitals
!        nci - total CI interactions
!        nci_max - maximum calculated CI interactions
!        nl  - lower limit (for systems w/ <10 oribtals)
!        nu  - upper limit (for systems w/ <10 orbitals)
      parameter(nv = 20, nc = 20, nci_max = nv*nc)
      integer iconfig(nci_max,2)
!        keeps track of configurations
!        (n-9,n+1)
!        (n-9,n+2)
!           ...
!        (n-9,n+10)
!        (n-8,n+1)
!           ...
!        (n,n+10)
!      real*8 hfoc(norbit)
!      real*8 exciton(norbit),tranvect(norbit)
!      real*8 hodens(norbit),eldens(norbit)
```

```
!       real*8 nco1(norbit,norbit),nco2(norbit,norbit)
!       real*8 nco3(norbit,norbit),nco4(norbit,norbit)
      real*8 nhf(norbit),nh(norbit),ne(norbit)
      real*8 dens(norbit,norbit)
      real*8 cibcm(norbit,norbit)
      real*8 hcst(norbit,norbit),gst(norbit,norbit)


!     Summation and Integral Variables
      real*8 qx(nci_max),qy(nci_max),qz(nci_max)
      real*8 qxsum,qysum,qzsum,Jint,Kint
      real*8 s1,s2,qtot,fosci,sume,sumh
      real*8 sum1,sum2,sum3,sum4
      real*8 sum,p

!     CI Hamiltonian matrices
      real*8, allocatable :: sing(:,:)
      real*8, allocatable :: trip(:,:)
      real*8, allocatable :: A1(:,:),A2(:,:)


!     LAPACK/BLAS work variables
      real*8, allocatable :: d(:)
      real*8, allocatable :: work(:)
      real*8, allocatable :: evec(:,:)
      integer lwork,info
!        d    - eigenvalues of CI hamiltonian
!        work - work variable

!     Density matrix (State Rep)
      real *8 stds(norbit,norbit)
```

```
!     New Coherences variables
      integer r,s,o,kd

      nl = max(ci_nfill - nv + 1, 1)
      nu = min(ci_nfill + nc, norbit)
      nci = (ci_nfill - nl + 1)*(nu - ci_nfill)
      lwork = 32*nci

307   format(X,A9,I3)
      write(ciout,307) 'ci_nfill: ', ci_nfill
      write(ciout,307) 'nl:       ', nl
      write(ciout,307) 'nu:       ', nu
      write(ciout,307) 'norbit:   ', norbit
      write(ciout,307) 'nci:      ', nci
      write(ciout,307) 'iex:      ', iex
      write(ciout,*)

      if(iex.lt.1 .or. iex.gt.nci) then
         write(ciout,*) 'Invalid excitation number'
         call fatal
      end if

!     Allocate orbitals according to the CI interactions
      allocate (sing(nci,nci))
      allocate (trip(nci,nci))
      allocate (d(nci))
      allocate (work(lwork))
      allocate (evec(nci,nci))

!-----Test Print 10/8/16----------
      write(ciout,*) 'Input Energies (ci_en)'
```

```
116         do i=1,norbit
               write(ciout,*) i, ci_en(i)
            end do
    !-----End Test Print--------------

121 !       sum over molecular orbitals
            m = 0
            do i = nl, ci_nfill
               do j=ci_nfill+1, nu
                  m = m+1
126               iconfig(m,1) = i
                  iconfig(m,2) = j

                  qxsum = 0.0d0
                  qysum = 0.0d0
131               qzsum = 0.0d0
    !       transitions before CI
                  qx(m) = 0.0d0
                  qy(m) = 0.0d0
                  qz(m) = 0.0d0
136
                  do iip = 1,norbit
                     iorb = iorbit(iip)
                     qxsum = qxsum + ci_v(iip,i)*ci_v(iip,j)*x(iorb)
                     qysum = qysum + ci_v(iip,i)*ci_v(iip,j)*y(iorb)
141                  qzsum = qzsum + ci_v(iip,i)*ci_v(iip,j)*z(iorb)
                  end do
                  qx(m) = qxsum
                  qy(m) = qysum
                  qz(m) = qzsum
146
```

```
                ma = 0

!       construct CI Hamiltonian for singlet and triplet
                do k = nl,ci_nfill
151                 do l = ci_nfill+1,nu
                        ma = ma + 1
                        Jint = 0.0d0
                        Kint = 0.0d0

156                     do iip=1,norbit
                            s1 = 0.0d0
                            s2 = 0.0d0
                            do iir = 1,norbit
                               s1 = s1 + ci_gamma(iip,iir)*ci_v(iir,j)*
161         >                        ci_v(iir,l)
                               s2 = s2 + ci_gamma(iip,iir)*ci_v(iir,k)*
            >                        ci_v(iir,l)
                            end do
                            Jint = Jint + ci_v(iip,i)*ci_v(iip,k)*s1
166                         Kint = Kint + ci_v(iip,i)*ci_v(iip,j)*s2
                        end do
                        sing(m,ma) = -Jint + 1.0d0*Kint
!       Original        sing(m,ma) = -Jint + 2.0d0*Kint
                        sing(ma,m) = sing(m,ma)
171                     trip(m,ma) = -Jint
                        trip(ma,m) = -Jint
                    end do
                end do
                sing(m,m) = sing(m,m) + (ci_en(j)-ci_en(i))
176             trip(m,m) = trip(m,m) + (ci_en(j)-ci_en(i))
            end do
```

```
      end do
!-----Test Print--------
!     Need to see what all the values in iconfig mean
      write(ciout,*) "  m      i    j"
 999  format(i5,i5,'  ->  ',i5)
      do i=1,m
         write(ciout,999) i, iconfig(i,1), iconfig(i,2)
      end do
      write(ciout,*) ''
!-----End Test Print----



!
!     Print the CI Hamiltonians
!
 302  format(E13.5)
      if(usesinglet) then
         write(ciout,*) "CI Singlet Hamiltonian"
         do i=1,nci
            do j=1,nci
               write(ciout,302,advance='no') sing(i,j)
            end do
            write(ciout,*) ''
         end do
      end if
      if(usetriplet) then
         write(ciout,*) "CI Triplet Hamiltonian"
         do i=1,nci
            do j=1,nci
               write(ciout,302,advance='no') trip(i,j)
            end do
```

```
            write(ciout,*) ''
         end do
      end if
      write(ciout,*)
!
!     Get the eigensystem, print
!
      if(usesinglet) then
         call dsyev('V','U',nci,sing,nci,d,work,lwork,info)
!     At this point 'sing' holds the eigenvectors
!     Copy them to 'evec'
         do i=1,nci
            do j=1,nci
               evec(i,j) = sing(i,j)
            end do
         end do

         write(ciout,*) 'CI Singlet Energies (ev)'
      end if
      if(usetriplet) then
         call dsyev('V','U',nci,trip,nci,d,work,lwork,info)
!     At this point 'trip' holds the eigenvectors
!     Copy them to 'evec'
         do i=1,nci
            do j=1,nci
               evec(i,j) = trip(i,j)
            end do
         end do

         write(ciout,*) 'CI Triplet Energies (ev)'
      end if
```

```
!     Print out the energies
303   format(i5,2f12.6)
      do i=1,min(20,nci)
         write(ciout,303) i,d(i)*evolt
      end do
      write(ciout,*)
!
!     Analyse CI states
!
      if(usesinglet) then
         write(ciout,*) 'Analysis of Singlet States'
      else if(usetriplet) then
         write(ciout,*) 'Analysis of Triplet States'
      end if

!      do i=1,min(20,nci)
      do i=1,nci
         qxsum = 0.0d0
         qysum = 0.0d0
         qzsum = 0.0d0

         do j=1,nci
            qxsum = qxsum + evec(j,i)*qx(j)
            qysum = qysum + evec(j,i)*qy(j)
            qzsum = qzsum + evec(j,i)*qz(j)
         end do

         qtot = qxsum*qxsum + qysum*qysum + qzsum*qzsum
         fosci = 0.0875161*qtot*d(i)
         if(fosci.lt.10.0d0**(-90)) fosci = 0.0d0
```

```
271
            write(ciout,304) i,d(i)*evolt,fosci
      304   format('E(',i3,')␣=␣',f12.6,2x,'fosci␣=␣',e12.6)
            write(ciout,305) qxsum,qysum,qzsum,sqrt(qtot)
      305   format('qx␣=␣',e12.6, 2x, 'qy␣=␣',e12.6, 2x, 'qz␣=␣',e12.6,
276   >         2x,'qtot␣=␣',e12.6)

            write(ciout,*) 'v␣->␣!␣␣␣␣␣␣␣␣␣Ec-Ev␣␣␣␣␣␣␣␣␣␣coef'
            do j=1,nci
               if(evec(j,i)**2.gt.0.1) then
281               write(ciout,'(i3,"␣->",i3,3x,f12.6,3x,f12.6)')
          >              iconfig(j,1),iconfig(j,2),
          >              (ci_en(iconfig(j,2))-ci_en(iconfig(j,1)))*evolt,
          >              evec(j,i)
               end if
286         end do
            write(ciout,*)
         end do

      !
291   !     Modify electron densitites for excitations.
      !     He were use the assumption that the CI(S)
      !     state add and subtract single electron
      !     density from the ci_ed(i,j) matrix
      !
296   !     May need iex here.
      !
      !     First construct the density matrix using
      !     the CI coefficients for iex in MO basis
      !
301         sum1 = 0.0d0
```

```
      do i=1,norbit
         do j=1,norbit
            dens(i,j) = 0.0d0
         end do
      end do

      do m=1,nci
         i = iconfig(m,1)
         j = iconfig(m,2)
         dens(i,j) = evec(m,iex)
      end do

      write(ciout,*) 'CI Eigenvectors'
      do i=1,nci
         write(ciout,306) (evec(i,j),j=1,nci)
      end do
      write(ciout,*)

  306 format(8f12.6)
      write(ciout,*) 'CI expansion coefficients'
      do i=1,norbit
         write(ciout,306) (dens(i,j),j=1,norbit)
      end do
      write(ciout,*)
!     At this point, we have the CI density matrix in the local orbital basis.

!     Now we need to write the populations in the orbital basis. For this, we
!     first determine the orbital population changes
!
      do i=1,norbit
         nhf(i) = 0.0d0
```

```
            nh(i)  = 0.0d0
            ne(i)  = 0.0d0
            if(i.le.ci_nfill) then
               if(usetdhf) then
                  nhf(i) = 1.0d0
               else
                  nhf(i) = 2.0d0
               end if
            end if
         end do

         do ih = nl,ci_nfill
            sum = 0.0d0
            do ie = ci_nfill+1,nu
               sum = sum + dens(ih,ie)**2
            end do
            nh(ih) = sum
         end do
         do ie=ci_nfill+1,nu
            sum = 0.0d0
            do ih=nl,ci_nfill
               sum = sum + dens(ih,ie)**2
            end do
            ne(ie) = sum
         end do

   !     Check: write the occupations over the range nl->nu
    308  format(I3,4F13.5)
         write(ciout,*) "State Populations Post-CI over CI range"
         write(ciout,*) " k       nhf            nh            ne            adj"
         do k=nl,nu
```

```
            write(ciout,308) k,nhf(k),nh(k),ne(k),nhf(k)+ne(k)-nh(k)
         end do

366

   !-----Test Code 20 Nov 2017---------------------------------
   !     The state populations might be inherently included
   !     in the CI eigenvectors. So lets take the eigenvector
371 !     and use that to populate coherences and diagonals.

   !     start with the state representation
         do o=1, norbit                    ! state
            do m=1, norbit                 ! state
376            sum1 = 0.0d0
               stds(o,m) = 0.0d0
               do i=nl,ci_nfill            ! hole 1
                  do j=ci_nfill+1,nu       ! elec 1
                     do k=nl,ci_nfill      ! hole 2
381                     do l=ci_nfill+1,nu ! elec 2
                           kd = 0
                           if(l.eq.o .and. i.eq.k .and. m.eq.j) then
                              kd = kd+1
                           end if
                           if(l.eq.j .and. k.eq.m .and. o.eq.i) then
386                           kd = kd-1
                           end if
                           if(l.eq.j .and. i.eq.k .and. o.eq.m) then
                              if(o.le.ci_nfill) kd = kd+1
                           end if
391                        sum1= sum1 + float(kd)*dens(i,j)*dens(k,l)
                        end do
                     end do
```

```
                end do
            end do
            stds(o,m) = sum1
        end do
    end do

!   unitary transform into site represenation
    do r=1,norbit
        do s=1,norbit
            sum = 0.0d0
            do o=1,norbit
                do m=1,norbit
                    sum = sum + stds(o,m)*ci_v(r,o)*ci_v(s,m)
                end do
            end do
            cibcm(r,s) = sum
        end do
    end do

    write(ciout,*) 'Density Matrix (State Rep)'
    do i=1,norbit
        do j=1,norbit
            write(ciout,309,advance='no') stds(i,j)
        end do
        write(ciout,*) ''
    end do
    write(ciout,*) ''
```

```
!     Print out the CI density/bond-charge matrix
 309  format(F13.5)
      write(ciout,*) 'Density/Bond-Charge Matrix'
      do i=1,norbit
         do j=1,norbit
            write(ciout,309,advance='no') cibcm(i,j)
         end do
         write(ciout,*) ''
      end do
      write(ciout,*) ''

!     As a check, take a look at the changes in the bond-orders.
!     These will go in to the forcefield calculation next.
      write(ciout,*) 'bond   i   j   old new'
      do k=1, nbpi
         p = pnpl(k)
         i = ibpi(2,k)
         j = ibpi(3,k)
         if(usetdhf) then
            if(useurhf) then
               pnpl(k) = realpart(cibcm(i,j) + tdhfedb(i,j))
            else
               pnpl(k) = realpart(cibcm(i,j) + 0.5d0*tdhfed(i,j))
            end if
!-----Test 03.20.17----
            pbpl(k) = pnpl(k)*ci_hc(i,j)/(-0.0757d0)
!-----------------------
         else
            pnpl(k) = cibcm(i,j)
!-----Test 03.20.17----
            pbpl(k) = cibcm(i,j)*ci_hc(i,j)/(-0.0757d0)
```

```
!----------------------
          end if
          i = ibnd(1,ibpi(1,k))
          j = ibnd(2,ibpi(1,k))
          write(ciout,'(5i5,2f12.2)') k,ibpi(2,k),ibpi(3,k),i,j,p,pnpl(k)
       end do
       write(ciout,*)


!-----------------------------------------------------------------------
!      If needed, update the TDHF electron density matrix
       if(usetdhf) then
          write(ciout,*)
          write(ciout,*) 'Updating TDHF electron density matrix'
          write(ciout,*)
          do i=1, norbit
             do j=1, norbit
                if(useurhf) then
                   tdhfeda(i,j) = cibcm(i,j)
                else
                   tdhfed(i,j) = cibcm(i,j) + 0.5d0*tdhfed(i,j)
                end if
             end do
          end do
       end if
!-----------------------------------------------------------------------

       call flush(ciout)

       end subroutine
```

A.3 ciinnit.f

```
!
!     ################################################
!     ##         Written by Kush Patel - 2/29/16       ##
!     ################################################
!
!     ##################################################################
!     ##                                                              ##
!     ##  subroutine ciinit  -- checks for keyword "EXCITES" or       ##
!     ##                        "EXCITET" for single or triplet       ##
!     ##                        calculations respectively, then       ##
!     ##                        initializes necessary values          ##
!     ##                                                              ##
!     ##################################################################
!
!     "ciinit" checks for the keyword "EXCITES" or "EXCITET" which
!     decides whether to do excited state calculations
!        'EXCITES' - singlet
!        'EXCITET' - triplet


      subroutine ciinit
      use sizes
      use keys
      use civars
      logical exist
      integer i, next, freeunit
      character*20 keyword
      character*120 record, string
      character*120 cifile
```

```
      usesinglet = .false.
      usetriplet = .false.
      do i=1,nkey
         next = 1
         record = keyline(i)
         call gettext (record,keyword,next)
         call upcase (keyword)
         if(keyword(1:8) .eq. 'EXCITES ') then
            usesinglet = .true.
            usetriplet = .false.
            ciout = freeunit()
            cifile = 'ci.sing.debug'
            string = record(next:120)
            iex = 0
            read(string,*,err=10,end=10) iex
   10       continue
         else if(keyword(1:8) .eq. 'EXCITET ') then
            usesinglet = .false.
            usetriplet = .true.
            ciout = freeunit()
            cifile = 'ci.trip.debug'
            string = record(next:120)
            iex = 0
            read(string,*,err=11,end=11) iex
   11       continue
         end if
      end do

!     open the CI debug file
      if(usesinglet.or.usetriplet) then
         inquire(file=cifile,exist=exist)
```

```
      if(exist) then
         open(unit=ciout,file=cifile,status='old')
         rewind(ciout)
      else
         open(unit=ciout,file=cifile,status='new')
      end if
   end if


   write(ciout,*) 'ci sing: ', usesinglet
   write(ciout,*) 'ci trip: ', usetriplet

   return
   end
```

A.4 civars.f

```
!
!     ################################################################
!     ##                                                            ##
!     ##  module civars  --  contents of the CI singlet and         ##
!     ##                     triplet calculations                   ##
!     ##                                                            ##
!     ################################################################
!
!
!     maxkey    maximum number of lines in the keyword file
!
!     nkey      number of nonblank lines in the keyword file
!     keyline   contents of each individual keyword file line
!
```

```
!
      module civars
      implicit none
      logical usetriplet,usesinglet
      integer ciout

!     copy variables from other calculations
      integer ci_nfill, iex
      real*8, allocatable :: ci_gamma(:,:)
      real*8, allocatable :: ci_v(:,:)
      real*8, allocatable :: ci_en(:)
      real*8, allocatable :: ci_ed(:,:)
      real*8, allocatable :: ci_hc(:,:)
      save
      end
```

A.5 fullpi.f

```
!
!
!     ######################################################################
!     ##                                                                  ##
!     ##  subroutine fullpi  --  replaces subroutine pical!               ##
!     ##                         because the given routine does not       ##
!     ##                         include the whole pi system              ##
!     ##                         together in one calculation              ##
!     ##                                                                  ##
!     ######################################################################
!
!
!     "picalc" performs a modified Pariser-Parr-Pople molecular
```

```
!     orbital calculation for each conjugated pi-system. The
!     implication here is that the pi-systems, then, do not
!     interact with each other except by force fields (MM). As a
!     result, there are no population dynamics between pi-systems.
!
!     "fullpi" does the same calculation, but treats all pi orbitals
!     as a single large pi-system
!

      subroutine fullpi
      use sizes
      use bndstr
      use couple
      use inform
      use iounit
      use piorbs
      use tors
!-------------------------
!     for tdhf calculations
      use tdhfvars
!     for CI calculations
      use civars
!-------------------------
      use atoms
!
      implicit none
      integer i,j,k,m,ib,i!
      integer ii,jj,kk
      integer iorb,jorb
      integer ncalls
      data ncalls  / 0 /
```

```
      save ncalls
      real*8 xij,yij,zij,rij

!      write(iout,*) "Full Pi called"
!      write(iout,*) "Status: ", tdhffirst


!
!
!     only needs to be done if pisystem is present
!
      if (norbit .eq. 0)  return
!
!     compute MOs for full pisystem
!
      norbit = 0
      do i = 1, nconj
         do j = iconj(1,i), iconj(2,i)
            norbit = norbit + 1
            iorbit(norbit) = kconj(j)
         end do
      end do
!      write(iout,*) 'norbit:', norbit
!
!     find and store the pisystem bonds for this pisystem
!
!---------old code-----------------------------------------------
      nbpi = 0
         kk = iconj(2,i) - iconj(1,i) + 1
      do ii = 1, norbit-1
         iorb = iorbit(ii)
         do jj = ii+1, norbit
```

```
76              jorb = iorbit(jj)
               do k = 1, n12(iorb)
                   if (i12(k,iorb) .eq. jorb) then
                      nbpi = nbpi + 1
                      do m = 1, nbond
81                       if (iorb.eq.ibnd(1,m) .and.
     &                        jorb.eq.ibnd(2,m)) then
                            ibpi(1,nbpi) = m
                            ibpi(2,nbpi) = ii
                            ibpi(3,nbpi) = jj
86                          goto 10
                         end if
                      end do
    10                continue
                   end if
91              end do
                end do
             end do
!-----------------------------------------------------------------
!-------------new code------------------------------------------
96 !     Point of this code is to find pi-atoms within a certain
!     distance of each other and consider them pi-bonded.
!     This will allow for inter-pisystem coupling and therefore
!     population dynamics therewithin. This should hold for
!     molecular systems with unconnected pisystems and
101 !     pairs of molecules that are within proximity.
!
!      nbpi = 0
!      do i = 1, norbit-1
!         iorb = iorbit(i)
106 !         do j = i+1, norbit
```

```
!             jorb = iorbit(j)
!             nbpi = nbpi + 1
!             do m = 1, nbond
!                ibpi(1,nbpi) = m
!                ibpi(2,nbpi) = i
!                ibpi(3,nbpi) = j
!             end do
!          end do
!       end do
!      Notes:
!      bohr - ratio of Bohr/Angstrom

!-----------------------------------------------------------------


c
!     find and store the pisystem torsions for this pisystem
c
      ntpi = 0
      do ii = 1, ntors
         ib = itors(2,ii)
         ic = itors(3,ii)
         if (listpi(ib) .and. listpi(ic)) then
            do jj = 1, nbpi
               k = ibpi(1,jj)
               if (ib.eq.ibnd(1,k).and.ic.eq.ibnd(2,k) .or.
     &             ib.eq.ibnd(2,k).and.ic.eq.ibnd(1,k)) then
                  ntpi = ntpi + 1
                  itpi(1,ntpi) = ii
                  itpi(2,ntpi) = jj
                  goto 20
               end if
```

```
            end do
      20    continue
         end if
141      end do
   !
   !     print a header for the molecular orbital calculation
   !
         if (verbose) then
146         if (nconj .eq. 1) then
               write (iout,30)
      30       format (/,' Modified Pariser-Parr-Pople Molecular',
     &            ' Orbitals :')
            else
151            write (iout,40)  i
      40       format (/,' Modified Pariser-Parr-Pople MOs for',
     &            ' Pi-System',i4,' :')
            end if
         end if
156   !
   !     get SCF-MOs, then scale bond and torsional parameters
   !

   !     need to see if TDHF is requested
161      if(usetdhf .and. .not. tdhffirst) then
            if(useurhf) then
               call pitduhf
            else
               call pitdrhf
166         end if
         else
            call fullpiscf
```

```
!     if CI calculations are requested
         if(usesinglet.or.usetriplet) call cical!

         tdhffirst = .false.
      end if

      call pialter
!     end do

      return
      end


!
!     ################################################################
!     ##                                                            ##
!     ##  subroutine fullpiscf  --  SCF molecular orbital           ##
!     ##                            calculation                     ##
!     ##                                                            ##
!     ################################################################
!
!
!     "piscf" performs an SCF molecular orbital calculation for a
!     pisystem to determine bond orders used in parameter scaling
!
!
!     "fullpiscf" is a modificaion of "piscf" that fills in
!     off diagonal terms that allow for intermolecular/interpisystem
!     interactions
!
      subroutine fullpiscf
```

```
      use sizes
      use atomid
      use atoms
      use bndstr
      use couple
      use inform
      use iounit
      use orbits
      use piorbs
      use pistuf
      use units

!-----------------------------------------------
!     for the initialization of tdhf variables
      use tdhfvars
      use civars
!-----------------------------------------------

      implicit none
      integer i,j,k,m
      integer iter,maxiter
      integer iatn,jatn
      integer iorb,jorb,nfill
      real*8 delta,converge
      real*8 xij,yij,zij,p
      real*8 hcii,gii,gij
      real*8 g11,g11sq,g12,g14
      real*8 rij,erij,brij
      real*8 ovlap,covlap
      real*8 cionize
      real*8 iionize,jionize
```

```
231         real*8 rijsq,hcij,qi
            real*8 total,totold
            real*8 ebeta,aeth,abnz
            real*8 ebe,ebb,ble,blb
            real*8 eebond,bebond
236         real*8 s1,s2,gjk
            real*8 vij,vik,vmj,vmk
            real*8 xi,xj,xk,xg
            real*8, allocatable :: povlap(:,:)
            real*8, allocatable :: en(:)
241         real*8, allocatable :: ip(:)
            real*8, allocatable :: fock(:,:)
            real*8, allocatable :: hc(:,:)
            real*8, allocatable :: v(:,:)
            real*8, allocatable :: gamma(:,:)
246         real*8, allocatable :: ed(:,:)
            character*6 mode
      !-----Testing Variables---------------------
            real*8, allocatable :: FR(:,:)
            real*8, allocatable :: RF(:,:)
251
            integer testout

            allocate (FR(norbit,norbit))
            allocate (RF(norbit,norbit))
256   !-------------------------------------------
      !
      !
      !     initialize some constants and parameters
      !
261   !     mode       planar or nonplanar pi-calculation
```

```
!     maxiter    maximum number of SCF iterations
!     converge   criterion for SCF convergence
!     ebeta      value of resonance integral for ethylene
!     cionize    ionization potential of carbon (Hartree)
!
      mode = 'PLANAR'
      maxiter = 500 ! default is 50
      converge = 0.00000001d0
      ebeta = -0.0757d0
      cionize = -11.16d0 / evolt
!
!     set the bond energies, alpha values and ideal bond length
!     parameter for carbon-carbon pibond type parameters
!
!     ebe    equilibrium bond energy in ethylene
!     ebb    equilibrium bond energy in benzene
!     aeth   the P-P-P constant "a" in ethylene
!     abnz   the P-P-P constant "a" in benzene
!     ble    equilibrium bond length in ethylene
!     blb    equilibrium bond length in benzene
!
      ebe = 129.37d0
      ebb = 117.58d0
      aeth = 2.309d0
      abnz = 2.142d0
      ble = 1.338d0
      blb = 1.397d0
!
!     perform dynamic allocation of some local arrays
!
      allocate (povlap(norbit,norbit))
```

```
      allocate (en(norbit))
      allocate (ip(norbit))
      allocate (fock(norbit,norbit))
      allocate (hc(norbit,norbit))
      allocate (v(norbit,norbit))
      allocate (gamma(norbit,norbit))
      allocate (ed(norbit,norbit))
!
!     assign empirical one-center Coulomb integrals, and
!     first or second ionization potential depending on
!     whether the orbital contributes one or two electrons
!
      nfill = 0
      do i = 1, norbit
         iorb = iorbit(i)
         gamma(i,i) = emorb(iorb)
         ip(i) = worb(iorb) + (1.0d0-qorb(iorb))*emorb(iorb)
         nfill = nfill + nint(qorb(iorb))
      end do
      nfill = nfill / 2
!
!     calculate two-center repulsion integrals
!     according to Ohno's semi-empirical formula
!
      do i = 1, norbit-1
         iorb = iorbit(i)
         gii = gamma(i,i)
         do j = i+1, norbit
            jorb = iorbit(j)
            g11 = 0.5d0 * (gii+gamma(j,j))
            g11sq = 1.0d0 / g11**2
```

```
            xij = x(iorb) - x(jorb)
            yij = y(iorb) - y(jorb)
            zij = z(iorb) - z(jorb)
            rijsq = (xij**2 + yij**2 + zij**2) / bohr**2
            g12 = 1.0d0 / sqrt(rijsq+g11sq)
            gamma(i,j) = g12
            gamma(j,i) = g12
         end do
      end do
!
!     zero out the resonance integral values
!
      do i = 1, norbit
         do j = 1, norbit
            hc(j,i) = 0.0d0
         end do
      end do
!
!     the first term in the sum to find alpha is the first
!     or second ionization potential, then the two-center
!     repulsion integrals are added
!
      do i = 1, norbit
         hcii = ip(i)
         do j = 1, norbit
            if (i .ne. j) then
               jorb = iorbit(j)
               hcii = hcii - qorb(jorb)*gamma(i,j)
            end if
         end do
         hc(i,i) = hcii
```

```
      end do
!
!               ----Original Code-----------
!
!     get two-center repulsion integrals via Ohno's formula
!
!      do k = 1, nbpi
!         i = ibpi(2,k)
!         j = ibpi(3,k)
!c       do i = 1, norbit-1
!c       do j = i+1, norbit
!         iorb = iorbit(i)
!         jorb = iorbit(j)
!         iatn = atomic(iorb)
!         jatn = atomic(jorb)
!         xij = x(iorb) - x(jorb)
!         yij = y(iorb) - y(jorb)
!         zij = z(iorb) - z(jorb)
!         rij = sqrt(xij**2 + yij**2 + zij**2)
!         rijsq = rij**2 / bohr**2
!         g11 = 0.5d0 * (gamma(i,i)+gamma(j,j))
!         g11sq = 1.0d0 / g11**2
!         g12 = gamma(i,j)
!
!     compute the bond energy using a Morse potential
!
!         erij = aeth * (ble-rij)
!         brij = abnz * (blb-rij)
!         eebond = (2.0d0*exp(erij)-exp(2.0d0*erij)) * ebe / hartree
!         bebond = (2.0d0*exp(brij)-exp(2.0d0*brij)) * ebb / hartree
!
```

```
!     compute the carbon-carbon resonance integral using
!     the Whitehead and Lo formula
!
!         g14 = 1.0d0 / sqrt(4.0d0*rijsq+g11sq)
!         hcij = 1.5d0*(bebond-eebond) - 0.375d0*g11
!     &             + (5.0d0/12.0d0)*g12 - g14/24.0d0
!
!     if either atom is non-carbon, then factor the resonance
!     integral by overlap ratio and ionization potential ratio
!
!         if (iatn.ne.6 .or. jatn.ne.6) then
!            call overlap (iatn,jatn,rij,ovlap)
!            call overlap (6,6,rij,covlap)
!            hcij = hcij * (ovlap/covlap)
!            iionize = ip(i)
!            if (qorb(iorb) .ne. 1.0d0) then
!               if (iatn .eq. 7)  iionize = 0.595d0 * iionize
!               if (iatn .eq. 8)  iionize = 0.525d0 * iionize
!               if (iatn .eq. 16)  iionize = 0.89d0 * iionize
!            end if
!            jionize = ip(j)
!            if (qorb(jorb) .ne. 1.0d0) then
!               if (jatn .eq. 7)  jionize = 0.595d0 * jionize
!               if (jatn .eq. 8)  jionize = 0.525d0 * jionize
!               if (jatn .eq. 16)  jionize = 0.89d0 * jionize
!            end if
!            hcij = hcij * (iionize+jionize)/(2.0d0*cionize)
!         end if
!
!     set symmetric elements to the same value
!
```

```
!          hc(i,j) = hcij
!          hc(j,i) = hcij
!       end do
!       end do

!-----Test Code 11.04.16-----------------------------------
!     Check if a particular pair is bonded. If so, do hc calcs
!     as normal. If not, apply the calculation and scale it
!     by Yukawa.
!---------------------------------------------------------------
!       do k = 1, nbpi
!          i = ibpi(2,k)
!          j = ibpi(3,k)
        do i = 1, norbit-1
        do j = i+1, norbit
           iorb = iorbit(i)
           jorb = iorbit(j)
           iatn = atomic(iorb)
           jatn = atomic(jorb)
           xij = x(iorb) - x(jorb)
           yij = y(iorb) - y(jorb)
           zij = z(iorb) - z(jorb)
           rij = sqrt(xij**2 + yij**2 + zij**2)
           rijsq = rij**2 / bohr**2
           g11 = 0.5d0 * (gamma(i,i)+gamma(j,j))
           g11sq = 1.0d0 / g11**2
           g12 = gamma(i,j)
!
!     compute the bond energy using a Morse potential
!
```

```
            erij = aeth * (ble-rij)
            brij = abnz * (blb-rij)
            eebond = (2.0d0*exp(erij)-exp(2.0d0*erij)) * ebe / hartree
            bebond = (2.0d0*exp(brij)-exp(2.0d0*brij)) * ebb / hartree
      !
      !     compute the carbon-carbon resonance integral using
      !     the Whitehead and Lo formula
      !
            g14 = 1.0d0 / sqrt(4.0d0*rijsq+g11sq)
            hcij = 1.5d0*(bebond-eebond) - 0.375d0*g11
           &             + (5.0d0/12.0d0)*g12 - g14/24.0d0
      !
      !     if either atom is non-carbon, then factor the resonance
      !     integral by overlap ratio and ionization potential ratio
      !
            if (iatn.ne.6 .or. jatn.ne.6) then
               call overlap (iatn,jatn,rij,ovlap)
               call overlap (6,6,rij,covlap)
               hcij = hcij * (ovlap/covlap)
               iionize = ip(i)
               if (qorb(iorb) .ne. 1.0d0) then
                  if (iatn .eq. 7)  iionize = 0.595d0 * iionize
                  if (iatn .eq. 8)  iionize = 0.525d0 * iionize
                  if (iatn .eq. 16)  iionize = 0.89d0 * iionize
               end if
               jionize = ip(j)
               if (qorb(jorb) .ne. 1.0d0) then
                  if (jatn .eq. 7)  jionize = 0.595d0 * jionize
                  if (jatn .eq. 8)  jionize = 0.525d0 * jionize
                  if (jatn .eq. 16)  jionize = 0.89d0 * jionize
               end if
```

```
                hcij = hcij * (iionize+jionize)/(2.0d0*cionize)
            end if
!
!     See if this pair of orbitals is bonded.
!
            bonded = .false.
            do k=1, nbpi
               iorb = ibpi(2,k)
               jorb = ibpi(3,k)
               bonded = (iorb.eq.i .and. jorb.eq.j) .or. bonded
            end do
            if(.not.bonded) then
               hcij = hcij*(exp(-1.0d0*rij/tdhfcr))/rij
 900           format(2i3,L5,E12.5)
            end if
!-----------------------------------------------
!
!     set symmetric elements to the same value
!
            hc(i,j) = hcij
            hc(j,i) = hcij
         end do
         end do

!---------------------------------------------------------------


!
!     construct an initial guess to the Fock matrix
!
         do i = 1, norbit
            do j = 1, norbit
```

```
                fock(j,i) = hc(j,i)
511          end do
          end do
          do i = 1, norbit
             fock(i,i) = 0.5d0 * ip(i)
          end do
516  !
     !     make the SCF-MO computation; do it twice, for a planar analog
     !     of the actual system and for the actual (nonplanar) system
     !
     !-----Test Code 31-Mar-18-----------------------------------
521  !     We're looking at convergence issues with deviant parameter
     !     sets. Print out iteration number, energy, and delta
          testout = 99
          open(unit=testout,file='SCF.out')

526       do while (mode.eq.'PLANAR' .or. mode.eq.'NONPLN')
             if (mode .eq. 'NONPLN') then
                call fulltilt (povlap)
     !           do k = 1, nbpi
     !              i = ibpi(2,k)
531  !              j = ibpi(3,k)
                do i = 1, norbit-1
                do j = i+1, norbit
                   hc(i,j) = hc(i,j) * povlap(i,j)
                   hc(j,i) = hc(i,j)
536             end do
                end do
             end if
     !
     !     perform SCF iterations until convergence is reached; diagonalize
```

```
!     the Fock matrix "f" to get the MOs, then use MOs to form the
!     next "f" matrix assuming zero differential overlap except for
!     one-center exchange repulsions
!

!-----Test Code 31-Mar-18----------------------------------------
          write(testout,*) mode
          write(testout,*) 'Iter␣␣␣Energy␣␣␣␣␣Delta'
 910      format(I4,E12.5,F17.5)
!---------------------------------------------------------------

          iter = 0
          delta = 2.0d0 * converge
          do while (delta.gt.converge .and. iter.lt.maxiter)
             iter = iter + 1
             call jacobi (norbit,fock,en,v)
             do i = 1, norbit
                do j = i, norbit
                   s1 = 0.0d0
                   s2 = 0.0d0
                   gij = gamma(i,j)
                   do k = 1, nfill
                      s2 = s2 - v(i,k)*v(j,k)*gij
                      if (i .eq. j) then
                         do m = 1, norbit
                            s1 = s1 + 2.0d0*gamma(i,m)*v(m,k)**2
                         end do
                      end if
                   end do
                   fock(i,j) = s1 + s2 + hc(i,j)
                   fock(j,i) = fock(i,j)
```

```
            end do
         end do
!
!     calculate the ground state energy, where "xi" sums the
!     molecular core integrals, "xj" sums the molecular coulomb
!     repulsion integrals, "xk" sums the molecular exchange
!     repulsion integrals, and "xg" sums the nuclear repulsion
!
         xi = 0.0d0
         xj = 0.0d0
         xk = 0.0d0
         xg = 0.0d0
         do i = 1, nfill
            do j = 1, norbit
               vij = v(j,i)
               do k = 1, norbit
                  vik = v(k,i)
                  gjk = gamma(j,k)
                  xi = xi + 2.0d0*vij*vik*hc(j,k)
                  do m = 1, nfill
                     vmj = v(j,m)
                     vmk = v(k,m)
                     xj = xj + 2.0d0*vij*vij*vmk*vmk*gjk
                     xk = xk - vij*vmj*vik*vmk*gjk
                  end do
               end do
            end do
         end do
         do i = 1, norbit-1
            iorb = iorbit(i)
            qi = qorb(iorb)
```

```
                  do j = i+1, norbit
                     jorb = iorbit(j)
                     xg = xg + qi*qorb(jorb)*gamma(i,j)
606                  end do
               end do
               total = xi + xj + xk + xg
               if (iter .ne. 1)  delta = abs(total-totold)
               totold = total
611 !-----Test Code 31-Mar-18-----------------------------------
               write(testout,910) iter,total,delta
    !-------------------------------------------------------------
            end do

616 !
    !     print warning if SCF-MO iteration did not converge
    !
            if (delta .gt. converge) then
               write (iout,10)
621    10        format (' PISCF  --  The SCF Molecular Orbitals have',
      &                  ' Failed to Converge')
    !          call fatal
            end if
    !
626 !     calculate electron densities from filled MO's
    !
            do i = 1, norbit
               do j = 1, norbit
                  ed(i,j) = 0.0d0
631               do k = 1, nfill
                     ed(i,j) = ed(i,j) + 2.0d0*v(i,k)*v(j,k)
                  end do
```

```
               end do
            end do
636
      !
      !     print out results for the SCF computation
      !
            if (verbose) then
641            if (mode .eq. 'PLANAR') then
                  write (iout,20)
         20       format (/,' SCF-MO Calculation for Planar System :')
               else
                  write (iout,30)
646      30       format (/,' SCF-MO Calculation for Non-Planar',
        &                    ' System :')
               end if
               write (iout,40)  total,norbit,delta,iter
         40    format (/,' Total Energy',11x,f12.4,
651     &              /,' Number of Orbitals',5x,i12,
        &              /,' Convergence',12x,d12.4,
        &              /,' Iterations',13x,i12)
               write (iout,50)  xi,xj,xk,xg
         50    format (/,' Core Integrals',9x,f12.4,
656     &              /,' Coulomb Repulsion',6x,f12.4,
        &              /,' Exchange Repulsion',5x,f12.4,
        &              /,' Nuclear Repulsion',6x,f12.4)
               write (iout,60)
         60    format (/,' Orbital Energies')
661            write (iout,70)  (en(i),i=1,norbit)
         70    format (8f9.4)
               write (iout,80)
         80    format (/,' MolecularOrbitals') ! Intentionally removed
```

```
                                                    ! space for easier grep'ing
666               do i = 1, norbit
                     write (iout,90)  (v(i,j),j=1,norbit)
       90            format (8f9.4)
                  end do
                  write (iout,100)
671    100        format (/,' Fock Matrix')
                  do i = 1, norbit
                     write (iout,110)  (fock(i,j),j=1,norbit)
       110           format (8f9.4)
                  end do
676               write (iout,120)
       120        format (/,' Density Matrix')
                  do i = 1, norbit
                     write (iout,130)  (ed(i,j),j=1,norbit)
       130           format (8f9.4)
681               end do
                  write (iout,140)
       140        format (/,' H-Core Matrix')
                  do i = 1, norbit
                     write (iout,150)  (hc(i,j),j=1,norbit)
686    150           format (8f9.4)
                  end do
                  write (iout,160)
       160        format (/,' Gamma Matrix')
                  do i = 1, norbit
691                  write (iout,170)  (gamma(i,j),j=1,norbit)
       170           format (8f9.4)
                  end do
               end if
```

```
696 !
    !     now, get the bond orders (compute p and p*b)
    !
             if (verbose) then
                write (iout,180)
701    180     format (/,' Pisystem Bond Orders')
             end if
             do k = 1, nbpi
                i = ibpi(2,k)
                j = ibpi(3,k)
706             p = 0.0d0
                do m = 1, nfill
                   p = p + 2.0d0*v(i,m)*v(j,m)
                end do
                if (mode .eq. 'PLANAR') then
711                pbpl(k) = p * hc(i,j)/ebeta
                else if (mode .eq. 'NONPLN') then
                   pnpl(k) = p
                end if
                if (verbose) then
716                i = ibnd(1,ibpi(1,k))
                   j = ibnd(2,ibpi(1,k))
                   write (iout,190)  i,j,p
      190          format (3x,2i6,2x,f10.4)
                end if
721          end do
    !
    !     if we have done planar calculation, do the nonplanar;
    !     when both are complete, alter the pisystem constants
    !
726          if (mode .eq. 'PLANAR') then
```

```
               mode = 'NONPLN'
            else if (mode .eq. 'NONPLN') then
               mode = '      '
            end if
731      end do
   !-----------------------------------------
         close(unit=testout)
   !-----------------------------------------


736
   !-------------------------------------------------
   !     If using tdhf, we need to allocate and copy
   !     over the matrices for propagation
   !      write(iout,*) "usetdhf: ", usetdhf
741      if(usetdhf) then
            tdhf_nfill = nfill
            if(.not.allocated(tdhfhc)) then
               allocate (tdhfhc (norbit,norbit))
            end if
746         if(.not.allocated(tdhfgamma)) then
               allocate (tdhfgamma(norbit,norbit))
            end if

            if(useurhf) then
751            if(.not.allocated(tdhfeda)) then
                  allocate (tdhfeda(norbit,norbit))
               end if
               if(.not.allocated(tdhfedb)) then
                  allocate (tdhfedb(norbit,norbit))
756            end if
               if(.not.allocated(tdhffocka)) then
```

```
            allocate (tdhffocka(norbit,norbit))
         end if
         if(.not.allocated(tdhffockb)) then
            allocate (tdhffockb(norbit,norbit))
         end if
      else
         if(.not.allocated(tdhffock)) then
            allocate(tdhffock(norbit,norbit))
         end if
         if(.not.allocated(tdhfed)) then
            allocate(tdhfed(norbit,norbit))
         end if
      end if

      do i=1,norbit
         do j=1,norbit
            tdhfhc(i,j)    = hc(i,j)
            tdhfgamma(i,j) = gamma(i,j)

            if(useurhf) then
               tdhfeda(i,j)   = ed(i,j)*0.5d0
               tdhfedb(i,j)   = ed(i,j)*0.5d0
               tdhffocka(i,j) = fock(i,j)
               tdhffockb(i,j) = fock(i,j)
            else
               tdhfed(i,j)   = ed(i,j)
               tdhffock(i,j) = fock(i,j)
            end if
         end do
      end do
   end if
```

```
!-------------------------------------------------
!     If using CI theory, we need to allocate and
!     copy over matrices: {gamma,v,en} to use
      if(usesinglet.or.usetriplet) then
!-------------------------------------------------------------
         if(usetriplet) then
            write(iout,*) 'Sorry, CI-triplets not available right now.'
            call fatal
         end if
!-------------------------------------------------------------
         write(ciout,*) "Allocating CI matrices"
         if(.not.allocated(ci_gamma)) allocate (ci_gamma(norbit,norbit))
         if(.not.allocated(ci_v))     allocate (ci_v(norbit,norbit))
         if(.not.allocated(ci_en))    allocate (ci_en(norbit))
         if(.not.allocated(ci_ed))    allocate (ci_ed(norbit,norbit))
         if(.not.allocated(ci_hc))    allocate (ci_hc(norbit,norbit))

         ci_nfill = nfill
         write(ciout,*) "Populating CI matrices"
         do i=1,norbit
            ci_en(i) = en(i)
            do j=1,norbit
               ci_v(i,j) = v(i,j)
               ci_gamma(i,j) = gamma(i,j)
               ci_ed(i,j) = ed(i,j)
               ci_hc(i,j) = hc(i,j)
            end do
         end do
         write(ciout,*) ''
      end if
!-------------------------------------------------
```

```
!
!     perform deallocation of some local arrays
!
      deallocate (povlap)
      deallocate (en)
      deallocate (ip)
      deallocate (fock)
      deallocate (hc)
      deallocate (v)
      deallocate (gamma)
      deallocate (ed)
      return
      end



!
!     ################################################################
!     ##                                                            ##
!     ##  subroutine fulltilt  --  direction cosines for pisystem   ##
!     ##                                                            ##
!     ################################################################
!
!
!     "fulltilt" calculates for each pibond the ratio of the
!     actual p-orbital overlap integral to the ideal overlap
!     if the same orbitals were perfectly parallel
!
!     performs the exact same calculation as PiTilt, but applies it
!     for all pi-atom pairs.
```

```
!
!
      subroutine fulltilt (povlap)
      use sizes
      use atomid
      use atoms
      use couple
      use piorbs
      implicit none
      integer i,j,k,m
      integer iorb,jorb
      integer list(8)
      real*8 ideal,cosine,rnorm
      real*8 xij,yij,zij,rij
      real*8 a1,b1,c1,a2,b2,c2
      real*8 x2,y2,z2,x3,y3,z3
      real*8 xr(8),yr(8),zr(8)
      real*8 povlap(norbit,norbit)
!
!
!     planes defining each p-orbital are in "piperp"; transform
!     coordinates of "iorb", "jorb" and their associated planes
!     to put "iorb" at origin and "jorb" along the x-axis
!
!      do k = 1, nbpi
!         i = ibpi(2,k)
!         j = ibpi(3,k)
       do i=1, norbit-1
       do j=i+1, norbit
         iorb = iorbit(i)
         jorb = iorbit(j)
```

```
            list(1) = iorb
            list(2) = jorb
            do m = 1, 3
               list(m+2) = piperp(m,iorb)
               list(m+5) = piperp(m,jorb)
            end do
            call pimove (list,xr,yr,zr)
!
!     check for sp-hybridized carbon in current bond;
!     assume perfect overlap for any such pibond
!
            if ((atomic(iorb).eq.6 .and. n12(iorb).eq.2) .or.
     &          (atomic(jorb).eq.6 .and. n12(jorb).eq.2)) then
               povlap(i,j) = 1.0d0
!
!     find and normalize a vector parallel to first p-orbital
!
            else
               x2 = xr(4) - xr(3)
               y2 = yr(4) - yr(3)
               z2 = zr(4) - zr(3)
               x3 = xr(5) - xr(3)
               y3 = yr(5) - yr(3)
               z3 = zr(5) - zr(3)
               a1 = y2*z3 - y3*z2
               b1 = x3*z2 - x2*z3
               c1 = x2*y3 - x3*y2
               rnorm = sqrt(a1*a1+b1*b1+c1*c1)
               a1 = a1 / rnorm
               b1 = b1 / rnorm
               c1 = c1 / rnorm
```

```
!
!     now find vector parallel to the second p-orbital,
!     "a2" changes sign to correspond to internuclear axis
!
            x2 = xr(7) - xr(6)
            y2 = yr(7) - yr(6)
            z2 = zr(7) - zr(6)
            x3 = xr(8) - xr(6)
            y3 = yr(8) - yr(6)
            z3 = zr(8) - zr(6)
            a2 = y2*z3 - y3*z2
            b2 = x3*z2 - x2*z3
            c2 = x2*y3 - x3*y2
            rnorm = sqrt(a2*a2+b2*b2+c2*c2)
            a2 = -a2 / rnorm
            b2 = b2 / rnorm
            c2 = c2 / rnorm
!
!     compute the cosine of the angle between p-orbitals;
!     if more than 90 degrees, reverse one of the vectors
!
            cosine = a1*a2 + b1*b2 + c1*c2
            if (cosine .lt. 0.0d0) then
               a2 = -a2
               b2 = -b2
               c2 = -c2
            end if
!
!     find overlap if the orbitals were perfectly parallel
!
            xij = x(iorb) - x(jorb)
```

```
                yij = y(iorb) - y(jorb)
                zij = z(iorb) - z(jorb)
946             rij = sqrt(xij**2 + yij**2 + zij**2)
                call overlap (atomic(iorb),atomic(jorb),rij,ideal)
!
!     set ratio of actual to ideal overlap for current pibond
!
951             povlap(i,j) = ideal*a1*a2 + b1*b2 + c1*c2
             end if
          end do
          end do
          return
956       end
```

A.6 pitdrhf.f

```
!     ************************************************************
!     ##  This code is written by Kush Patel.                   ##
3     !     ##                                                        ##
!     ##  Prints various matrices and calls the TDHF            ##
!     ##  iterator to update rho according to the               ##
!     ##  equation:                                             ##
!     ##                                                        ##
8     !     ##      (ih)(d rho/dt) = <[F(rho),rho]> = i L rho      ##
!     ##                                                        ##
!     ************************************************************

          subroutine pitdrhf
13        use sizes
          use atoms
          use atomid
```

```
      use files
      use iounit
      use orbits
      use piorbs
      use bndstr
      use units
      use tdhfvars

      implicit none

!     pitdhf variables
      integer i,j,k,m
      integer info
      complex*16 tau(norbit),work(8*norbit),fockcopy(norbit,norbit)
      real*8 d(norbit),e(norbit-1), p
!     variables for reconstructing fock
      integer iorb,jorb
      integer iatn,jatn
      complex*16 s1,s2,s3,sf4
      real*8 hcii, hcij
      real*8 brij,erij
      real*8 bebond,eebond
      real*8 ebb,ebe,abnz,aeth,ble,blb
      real*8 xij,yij,zij,rij,rijsq
      real*8 gii,gij,g11,g11sq,g12,g14
      real*8 ovlap,covlap
      real*8 cionize,iionize,jionize
      real*8 ip(norbit)
!     variables for population analysis
      real*8 Smatrix(norbit,norbit), pop(norbit)
!     variables for iterative output
```

```
      integer lext
      character*7 ext
      integer freeunit
      logical exist
!     variables for total energy calculation
      complex*16 energy
!     Dipole calculation
      real*8 hcom(3)
      real*8 ecom(3)
      real*8 qii
      real*8 dipl(3)
!-------Testing-----------
!     Seeing if original method of calculating energy
!     results in regular fluctuations
      complex*16 xi,xj,xk,xg,xcor
      complex*16 pii,pjj,pij
!-------------------------
!     Electron current
      real*8 currax, curray, curraz
      real*8 currbx, currby, currbz
      real*8 cc1,cc2,cc3

 300  format(2E17.6, 3X)
 301  format(8E16.5)

!     open an output file
      outiter = outiter + 1
      printq = mod(outiter,outfreq).eq.0
      printq = printq.or.(outiter.eq.1)
      lext = 6
      if(printq) then
```

```
78          call numeral(outiter,ext,lext)
            tdhfout = freeunit()
            inquire(file=filename//'.tdhf.'//ext(1:lext),exist=exist)
            if(exist) then
               open(unit=tdhfout,
83       &        file=filename(1:leng)//'.tdhf.'//ext(1:lext),status='old')
               rewind(unit=tdhfout)
            else
               open(unit=tdhfout,
       &        file=filename(1:leng)//'.tdhf.'//ext(1:lext),status='new')
88          end if

    309     format(A18,E16.5)
            write(tdhfout,*) '##␣Build-Date␣20-Mar-18␣##'
            write(tdhfout,*) '##␣Begin␣piTDHF␣iteration␣##'
93          write(tdhfout,*) ''
            write(tdhfout,309) '␣Coupling␣radius:␣', tdhfcr
            write(tdhfout,309) '␣Time␣Step:␣␣␣␣␣␣␣', tdhfdt
            write(tdhfout,*) ''
         end if
98
   !-----------------------------------------------------------
   !     Following code develops the gamma matrix as well
   !     as the hc matrix. These are the various integrals
   !     and the Hatree Core matrix
103 !     Taken straight from the piscf subroutine
   !     of picalc.f
   !-----------------------------------------------------------
   !     initialize some constants and parameters
         cionize = -11.16d0 / evolt
108
```

```
!     set the bond energies, alpha values and ideal bond length
!     parameter for carbon-carbon pibond type parameters
      ebe   = 129.37d0
      ebb   = 117.58d0
      aeth =   2.309d0
      abnz =   2.142d0
      ble   =   1.338d0
      blb   =   1.397d0

!     assign empirical one-center Coulomb integrals, and
!     first or second ionization potential depending on
!     whether the orbital contribures one or two electrons
      do i = 1, norbit
         iorb = iorbit(i)
         tdhfgamma(i,i) = emorb(iorb)
         ip(i) = worb(iorb) + (1.0d0-qorb(iorb))*emorb(iorb)
      end do

!     calculate two-center repulsion integrals
!     according to Ohno's semi-empirical formula
      do i = 1, norbit
         iorb = iorbit(i)
         gii = tdhfgamma(i,i)
         do j = i+1, norbit
            jorb = iorbit(j)
            g11 = 0.5d0 * (gii+tdhfgamma(j,j))
            g11sq = 1.0d0 / g11**2
            xij = x(iorb) - x(jorb)
            yij = y(iorb) - y(jorb)
            zij = z(iorb) - z(jorb)
            rijsq = (xij**2 + yij**2 + zij**2) / bohr**2
```

```
            g12 = 1.0d0 / sqrt(rijsq + g11sq)
            tdhfgamma(i,j) = g12
            tdhfgamma(j,i) = g12
         end do
      end do

!     the first term in the sum to find alpha is the first
!     or second ionization potential, then the two-center
!     repulsion integrals are added
      do i = 1, norbit
         hcii = ip(i)
         do j = 1, norbit
            if(i.ne.j) then
               jorb = iorbit(j)
               hcii = hcii - qorb(jorb)*tdhfgamma(i,j)
            end if
         end do
         tdhfhc(i,i) = hcii
      end do

!     get two-center repulsion integrals via Ohno's formula
!      do k = 1, nbpi
!         i = ibpi(2,k)
!         j = ibpi(3,k)
      do i = 1, norbit-1
      do j = i+1, norbit
         iorb = iorbit(i)
         jorb = iorbit(j)
         iatn = atomic(iorb)
         jatn = atomic(jorb)
         xij = x(iorb) - x(jorb)
```

```
            yij = y(iorb) - y(jorb)
            zij = z(iorb) - z(jorb)
173         rij = sqrt(xij**2 + yij**2 + zij**2)
            rijsq = rij**2 / bohr**2
            g11 = 0.5d0 * (tdhfgamma(i,i) + tdhfgamma(j,j))
            g11sq = 1.0d0 / g11**2
            g12 = tdhfgamma(i,j)
178
!     compute the bond energy using a Morse potential
            erij = aeth * (ble-rij)
            brij = abnz * (blb-rij)
            eebond = (2.0d0*exp(erij)-exp(2.0d0*erij)) * ebe / hartree
183         bebond = (2.0d0*exp(brij)-exp(2.0d0*brij)) * ebb / hartree

!     compute the carbon-carbon resonance integral using
!     the Whitehead and Lo formula
            g14 = 1.0d0 / sqrt(4.0d0*rijsq+g11sq)
188         hcij = 1.5d0*(bebond-eebond) - 0.375d0*g11
     &         + (5.0d0/12.0d0)*g12 - g14/24.0d0

!     if either atom is non-carbon, then factor the resonance
!     integral by overlap ratio and ionization potential ratio
193         if(iatn.ne.6 .or. jatn.ne.6) then
               call overlap(iatn,jatn,rij,ovlap)
               call overlap(6,6,rij,covlap)
               hcij = hcij * (ovlap/covlap)
               iionize = ip(i)
198            if(qorb(iorb) .ne. 1.0d0) then
                  if(iatn .eq.  7) iionize = 0.595d0 * iionize
                  if(iatn .eq.  8) iionize = 0.525d0 * iionize
                  if(iatn .eq. 16) iionize = 0.890d0 * iionize
```

```
               end if
               jionize = ip(j)
               if(qorb(jorb) .ne. 1.0d0) then
                  if(jatn .eq.  7) jionize = 0.595d0 * jionize
                  if(jatn .eq.  8) jionize = 0.525d0 * jionize
                  if(jatn .eq. 16) jionize = 0.890d0 * jionize
               end if
               hcij = hcij * (iionize+jionize)/(2.0d0*cionize)
            end if
!-----Test Code 11.04.16---------------------------------------
!     This code is the pitdhf copy of the similar thing in fullpi.f.
!     Applies Yukowa scaling to hcij terms of nonbonded atoms
!------------------------------------------------------------
            bonded = .false.
            do k=1, nbpi
               iorb = ibpi(2,k)
               jorb = ibpi(3,k)
               bonded = (iorb.eq.i .and. jorb.eq.j) .or. bonded
            end do
            if(.not.bonded) then
               hcij = hcij*(exp(-1.0d0*rij/tdhfcr))/rij
            end if
!------------------------------------------------------------


!     set symmetric elements to the same value
            tdhfhc(i,j) = hcij
            tdhfhc(j,i) = hcij
         end do
         end do

c!     reconstruct the Fock matrix
```

```
233 !       do i=1, norbit
    !          do j=1, norbit
    !             gij = tdhfgamma(i,j)
    !             s1 = (0.0d0,0.0d0)
    !             s2 = -1.0d0*tdhfed(i,j)*gij
238 c
    !             if(i.eq.j) then
    !                do k=1, norbit
    c!                  s1 = s1 + 2.0d0*tdhfgamma(i,k)*tdhfed(k,k) original
    !                   s1 = s1 + 1.0d0*tdhfgamma(i,k)*tdhfed(k,k)
243 !                end do
    !             end if
    !             tdhffock(i,j) = tdhfhc(i,j) + s1 + s2
    !          end do
    !       end do
248
    !      reconstruct the Fock Matrix
           do i=1, norbit
              do j=1, norbit
                 s1 = 0.0d0
253              if(i.eq.j) then
                    do k=1,norbit
                       s1 = s1 + tdhfed(k,k)*tdhfgamma(i,k)
                    end do
                 end if
258              s2 = 0.5d0*tdhfed(i,j)*tdhfgamma(i,j)

                 tdhffock(i,j) = tdhfhc(i,j) + s1 - s2
              end do
           end do
263
```

```
!     diagonalize the fock matrices to obtain eigenvalues
!     for the superoperator
c


c
!     print density matrices
c
      if(printq) then
         write(tdhfout,*) 'TDHF Electron Density Matrix'
         do i=1, norbit
            do j=1, norbit
               write(tdhfout,300,advance='no') tdhfed(i,j)
            end do
            write(tdhfout,*) ''
         end do
         write(tdhfout,*) ''

!     Print the diagonal elements
 307     format(2E20.9,3X)
         write(tdhfout,*) 'TDHF Density Matrix Diagonal'
         s1 = (0.0d0,0.0d0)
         do j=1, norbit
            s1 = s1 + tdhfed(j,j)
            if(printq) write(tdhfout,307) tdhfed(j,j)
         end do

 302     format(A12,2F17.6)
         write(tdhfout,*) ''
         write(tdhfout,302) 'Trace: ', s1
         write(tdhfout,*) ''
      end if
```

```
c
!     copy fock matrix to tdhfcopy
c
      if(printq) write(tdhfout,*) 'Fock matrix'
      do i=1, norbit
         do j=1, norbit
            fockcopy(i,j) = tdhffock(i,j)
            if(printq) then
               write(tdhfout,300,advance='no') tdhffock(i,j)
            end if
         end do
         if(printq) write(tdhfout,*) ''
      end do
      if(printq) write(tdhfout,*) ''

c
!     occasionally recalculate the eigenvalues
c
      if(mod(outiter,100).eq.1 .or. res1) then
         res1 = .false.
         if(tdhfdebug) then
            write(debugout,*) 'Recalculating Super Eigenvalues'
         end if
c
!     tridiagonalize
!
         call zhetrd('U',norbit,fockcopy,norbit,d,e,tau,
     &                 work,8*norbit,info)
         if(tdhfdebug) then
            write(debugout,*) 'Info: ', info
```

```
            write(debugout,*) ''
            write(debugout,*) 'Post Tridiagonal Fock'
            do i=1,norbit
               do j=1, norbit
                  write(debugout,300,advance='no') tdhffock(i,j)
               end do
               write(debugout,*)''
            end do
            write(debugout,*)''

            write(debugout,*) 'tri-diagonal matrix'
            write(debugout,*) 'diagonal     superdiagonal'
            do i=1, norbit
               write(debugout,301,advance='no') d(i)
               if(i.ne.norbit) then
                  write(debugout,301,advance='no') e(i)
               end if
               write(debugout,*) ''
            end do
            write(debugout,*) ''
         end if

!     get the eigenvalues
         superevmax = 0.0d0
         superevmin = 0.0d0

         call zsteqr('N',norbit,d,e,'null',norbit,'null',info)

         if(printq) then
            write(tdhfout,*) 'Fock Eigenvalues: '
            write(tdhfout,301) (d(i),i=1,norbit)
```

```
            write(tdhfout,*)''
358         end if

            superevmax = max(superevmax,d(norbit)-d(1))
            superevmin = min(superevmin,d(1)-d(norbit))
            if(printq) then
363            write(tdhfout,*) 'Supereigenvalues: {', superevmin,', ',
     &           superevmax, '}'
               write(tdhfout,*) ''
            end if
            superevmax = 1.5d0*superevmax
368         superevmin = 1.5d0*superevmin

         end if

c
373 !     calculate total electronic energy
c
         if(printq) then
            energy = (0.0d0,0.0d0)
            do i=1, norbit
378            do j=1, norbit
                  pij = tdhfed(i,j)
                  energy = energy + 0.5d0*pij*(tdhfhc(j,i)+tdhffock(j,i))
               end do
            end do
383  305    format(A25,2E16.5)
            write(tdhfout,305) 'Total Electronic Energy: ', energy
            write(tdhfout,*) ''
         end if
```

```
388         if(probcurr .and. printq) then
               write(tdhfout,*) 'Probability Current'
               write(tdhfout,*) ' qx    qy    qz    ax    ay    az bx by bz'
               do i=1  , norbit
                  iorb = iorbit(i)
393               currax = 0.0d0
                  curray = 0.0d0
                  curraz = 0.0d0
                  do j=1, norbit
                     jorb = iorbit(j)
398                  cc1 = 2.0d0*tdhfhc(j,i)*aimag(tdhfed(j,i))
                     currax = currax + cc1* (x(jorb)-x(iorb))
                     curray = curray + cc1* (y(jorb)-y(iorb))
                     curraz = curraz + cc1* (z(jorb)-z(iorb))
                  end do
403   306         format(9E16.5)
                  write(tdhfout,306) x(iorb),y(iorb),z(iorb),currax,curray,
         &            curraz,currbx,currby,currbz
               end do
            end if
408
    !      call tdhfiterate and update rho
           call tdhfiterate(tdhfed,tdhffock,superevmax,
         &      superevmin,tdhfdt,norbit,9)

413 !-----Test Code 13 Jan 18-----------------------------------
    !      Test fock and density matrices for Hermiteness
    !        s1 = 0.0d0 ! Density
    !        s2 = 0.0d0 ! Fock
    !        do i=1,norbit
418 !           do j=1,norbit
```

```
!              s3 = abs(realpart( tdhfed(i,j) - tdhfed(j,i) ))
!              s3 = abs(aimag(    tdhfed(i,j) + tdhfed(j,i) ))
!              s1 = s1 + s3 + s4
!              s3 = abs(realpart( tdhffock(i,j) - tdhffock(j,i) ))
!              s4 = abs(aimag(    tdhffock(i,j) + tdhffock(j,i) ))
!              s2 = s2 + s3 + s4
!           end do
!        end do
!  308   format(A4,2X,2E16.5)
!        write(tdhfout,*) 'Hermiteness: '
!        write(tdhfout,308) 'Dens',s1
!        write(tdhfout,308) 'Fock',s2
!        write(tdhfout,*) ''
!-------------------------------------------------------------

!     Impose zero for imaginary parts on diagonal
      do i=1,norbit
         tdhfed(i,i) = realpart(tdhfed(i,i))
      end do


!     update the nonplanar pi bond orders (pnpl)
!     this is the same code that's at the end of picalc

 401  format(5i5,2f12.2)
 402  format(3A5,2A15)
      if(printq) write(tdhfout,402) 'Bond','i','j','old','new'
      do k=1, nbpi
         i = ibpi(2,k)
         j = ibpi(3,k)
         p = pnpl(k)
```

```
!         take just the real part of electron density
!          pnpl(k) = realpart( tdhfed(i,j) )
!         take the real part of the electron density
!          pnpl(k) = zabs( tdhfeda(i,j) + tdhfedb(i,j))
          pnpl(k) = realpart( tdhfed(i,j) )
          pbpl(k) = pnpl(k) * tdhfhc(i,j)/(-0.0757d0)

          i = ibnd(1,ibpi(1,k))
          j = ibnd(2,ibpi(1,k))
          if(printq)
     &         write(tdhfout,401) k,ibpi(2,k),ibpi(3,k),i,j,p,pnpl(k)
       end do
       if(printq) write(tdhfout,*)''

!-------------------------
       if(printq) then
          xi = 0.0d0
          xj = 0.0d0
          xk = 0.0d0
          xg = 0.0d0
          xcor = 0.0d0

          write(tdhfout,*) 'Original Type Energy Calculation'
          do i=1, norbit
             do j=1, norbit
                pii = tdhfed(i,i)
                pjj = tdhfed(j,j)
                pij = tdhfed(i,j)
                xi = xi + pij*tdhfhc(i,j)
                xj = xj + 0.50d0*pij*tdhfgamma(i,j)
```

```
                    xk = xk - 0.25d0*pij*tdhfgamma(i,j)
                    xcor = xcor - 0.5d0*(pij*pij - pii*pjj)*tdhfgamma(i,j)
483                 end do
               end do
               do i=1, norbit-1
                  do j=i+1, norbit
                     xg = xg + tdhfgamma(i,j)
488                  end do
               end do
               write(tdhfout,*) 'OEnergy:  ', xi+xj+xk+xg+xcor
               write(tdhfout,*) 'Core:      ', xi
               write(tdhfout,*) 'Coulomb:  ', xj
493            write(tdhfout,*) 'Exchange: ', xk
               write(tdhfout,*) 'Nuclear:  ', xg
               write(tdhfout,*) 'Xcorrect: ', xcor
    !--------------------------


498            write(tdhfout,*)
               write(tdhfout,*)'## End of piTDHF iteration ##'
               write(tdhfout,*)''
               write(tdhfout,*)''

503 !      Close the output file
               flush(tdhfout)
               close(unit=tdhfout)

    !      Write restart information
508        call writerestart()


           end if
```

```
513         end subroutine
518
    !       ################################################################
    !       ##   This code is written by Kush Patel based off of the      ##
    !       ##   methods described in the following reference             ##
523 !       ##                                                            ##
    !       ##    An accurate and efficient scheme for propagating the    ##
    !       ##    time dependent Schrodinger equation                     ##
    !       ##    H. Tal-Ezer and R. Kosloff                              ##
    !       ##    The Journal of Chemical Physics                         ##
528 !       ##    81, 3967 (1984); doi: 10.1063/1.448136                  ##
    !       ################################################################
          subroutine tdhfiterate(rho0,oper,emax,emin,dt,m,Np)

    !       rho0 - complex*16, input/output, matrix size m*m
533 !              Density operator
    !              input is the state at time t
    !              output is the state at time t+dt
    !
    !       oper - complex*16, input, matrix size m*m
538 !              Fock Matrix
    !              operator Exp(-i 0 dt)
    !
    !       emax - real*8, input
    !              largest eigenvalue of oper
```

```
543 !
    !     emin - real*8, input
    !            smallest eigenvalue of oper
    !
    !     dt -   real*8, input
548 !            time step
    !
    !     m -    integer, input
    !            dimension of rho0 and oper
    !
553 !     Np -   integer, input
    !            Chebyshev polynomial expansion limit
    !            code will expand to polynomials J_0 to J_Np

          use sizes
558       use tdhfvars

          implicit none

          integer     i,j,k,m,Np,c(Np+1)
563       real*8      dt,emin,emax,R,G,hToEvPS,dta
          complex*16  ii,sum,a(Np+1)
          complex*16  oper(m,m),X(m,m)
          complex*16  phi(Np+1,m,m),rho0(m,m)
          complex*16  FR(m,m),RF(m,m),temp(m,m)
568       real*8      v1,v2

          dta = dt*41341.48d0 ! dt in atomic units

    !      tdhfdebug = .false.
573 !     Debugging prints
```

```
 111  format(2E17.6,3X)
      if(printq.and.tdhfdebug) then
         write(tdhfout,*) '#####␣TDHFITERATE␣START␣#####'
         write(tdhfout,*) 'emax␣=␣', emax
578      write(tdhfout,*) 'emin␣=␣', emin
         write(tdhfout,*) 'dt␣␣␣=␣', dt
         write(tdhfout,*) 'dta␣␣=␣', dta
         write(tdhfout,*) 'm␣␣␣␣=␣', m
         write(tdhfout,*) 'Np␣␣␣=␣', Np
583      write(tdhfout,*) "operator"
         do i=1, m
            do j=1, m
               write(tdhfout,111,advance='no') oper(i,j)
            end do
588         write(tdhfout,*) ''
         end do
         write(tdhfout,*) 'rho0␣in'
         do i=1, m
            do j=1, m
593            write(tdhfout,111,advance='no') rho0(i,j)
            end do
            write(tdhfout,*) ''
         end do
      end if
598
!     initialize some values
      ii = (0.d0,1.d0)
      R = 0.5*dta*(emax-emin)
      G = dta*emin
603
!     define X matrix
```

```
!       write(tdhfout,*) 'X(i,j)'
        do i=1,m
           do j=1,m
608           X(i,j)= -1*ii*dta*oper(i,j)/R
!             write(tdhfout,111,advance='no') X(i,j)
           end do
!          write(tdhfout,*) ''
        end do
613 !     write(tdhfout,*) ''

!     phi is a 3 index array that acts as a temporary
!     container for values. Better visualized as a list
!     of matrices.
618 !     these values will be summed up
!     phi is indexed as:
!       phi(expansion_number,density_matrix_row,
!                                 density_matrix_column)
!
623 !     the next 2 do loops populate phi(1) and phi(2)
!     phi(1)=rho0, phi(2)=[oper,rho]=oper.rho-rho.oper
!     note: phi(k) refers to the (k-1)th expansion
!      write(tdhfout,*) 'phi(1,i,j)'
c
628 !     Note: Fortran starts counting indices at 1
!     Therefore, index (k) refers to the (k-1)th
!     expansion. This is applicable to phi,a, and c

        do i=1,m
633        do j=1,m
              phi(1,i,j) = rho0(i,j)
!             write(tdhfout,*,advance='no') phi(1,i,j)
```

```
            end do
!           write(tdhfout,*) ''
        end do
!       write(tdhfout,*) ''

!       matrix dot product of X and rho0
        FR = matmul(X,rho0)
        RF = matmul(rho0,X)
!d      do i=1,m
!d         do j=1,m
!d            print *, FR(i,j)
!d            print *, RF(i,j)
!d         end do
!d         print *,""
!d      end do

        do i=1,m
           do j=1,m
              phi(2,i,j) = FR(i,j)-RF(i,j)

!       assign values to temp to be used as phi(k-1)
              temp(i,j)  = phi(2,i,j)
!d            print *, phi(2,i,j)
           end do
!d         print *,""
        end do

!       C(k) coefficients
!       c(0)   = 1,
!       c(k>1) = 2
        c(1)=1
```

```
        do k=2,Np+1
668        c(k)=2
   !d        print *, c(k)
        end do
        if(printq.and.tdhfdebug) then
           write(tdhfout,*) '!-coeffs'
673        do k=1,Np+1
              write(tdhfout,*) c(k)
           end do
        end if

678  !     use the recursion relation to populate phi
        do k=3,Np+1
           FR = matmul(X,temp)
           RF = matmul(temp,X)
           do i=1,m
683           do j=1,m
                 phi(k,i,j) = 2*FR(i,j)-2*RF(i,j)+phi(k-2,i,j)
                 temp(i,j) = phi(k,i,j)
              end do
           end do
688     end do
        if(printq.and.tdhfdebug) then
           do k=1,Np+1
              write(tdhfout,'("phi(",i3,")") ') k
              do i=1,m
693              do j=1,m
                    write(tdhfout,111,advance='no') phi(k,i,j)
                 end do
              write(tdhfout,*) ''
              end do
```

```
698            end do
           end if

   !       populate a(k) integrals
           do k=1,Np+1
              a(k) = CDEXP(ii*(R+G))*c(k)*besjN(k-1,R)
   !d           print *, a(k)
           end do
           if(printq.and.tdhfdebug) then
              write(tdhfout,*) 'alpha coeffs'
              do k=1,Np+1
                 write(tdhfout,111) a(k)
              end do
           end if

   !       perform final multiplication, sum up the expansions
   !       and store values in rho0
           do i=1,m
              do j=1,m
                 sum = (0.d0,0.d0)
                 do k=1,Np+1
                    sum = sum + a(k)*phi(k,i,j)
                 end do
                 rho0(i,j) = sum
              end do
   !d           print *,rho0(i,j)
           end do

   !-----Test Code 20 Mar 18---------------
   !       Long-time iterations accrue lots of error and we start
   !       getting non-Hermitian matrices.
```

```
!     Here we'll just average transposed elements (and make
!     sure imaginary sign is preserved).
      do i=1,m
         do j=1,m
            v1 = 0.5d0*(realpart( rho0(i,j)+rho0(j,i) ))
            v2 = 0.5d0*(imagpart( rho0(i,j)-rho0(j,i) ))
            rho0(i,j) = v1 + ii*v2
            rho0(j,i) = v1 - ii*v2
         end do
      end do


      if(printq.and.tdhfdebug) then
         write(tdhfout,*) "rho0 out"
         do i=1, m
            do j=1, m
               write(tdhfout,111,advance='no') rho0(i,j)
            end do
            write(tdhfout,*) ''
         end do
      end if

      if(printq.and.tdhfdebug) then
         write(tdhfout,*) "##### TDHFITERATE END #####"
         write(tdhfout,*) ''
         write(tdhfout,*) ''
         write(tdhfout,*) ''
      end if
```

```
      return
      end

763

!     ################################################################
!     ##                                                            ##
!     ##   This code is written by Kush Patel based off of the      ##
768 !     ##   methods described in the following reference         ##
!     ##                                                            ##
!     ##    Population Analysis (Mulliken, Lowdin)                  ##
!     ##    Modern Quantum Chemistry                                ##
!     ##    Attila Szabo, Neil S. Ostlund                           ##
773 !     ##                                                            ##
!     ################################################################

      subroutine popanal(ml,N,rho,S,pop)

778 !     ml -  character, input, length 1
!           'M' for Mulliken Poplation Analysis
!           'L' for Lowdin Population Analysis
!
!     N -   integer, input
783 !           dimension of square matrices rho and S
!
!     rho - complex*16, input, matrix size N*N
!           Electron Density Matrix
!
788 !     S -   real*16, input, matrix size N*N
!           Overlap Matrix
!
```

```
!     pop - real*16, output, vector size N
!           will contain the electron populations on exit

      use sizes
      use tdhfvars
      implicit none

      character*1  ml
      integer      N,i,j,lwrk,info
      real*8       S(N,N), pop(N),wrk(N*34),evals(N)
      real*8       SevecsT(N,N), Ssqrt(N,N)
      complex*16   rho(N,N), rho2(N,N)

!     Mulliken Population Analysis is most simple.
!     P' = P.S
!     Check for this first
      if(ml .eq. 'M') then
         rho2 = matmul(rho,S)
         do i=1,N
            pop(i) = zabs(rho2(i,i))
         end do
         return
      end if

!-----------------------------------------------------------------
!     At this point, Mulliken was not chosen, default to Lowdin

!     Diagonalize the overlap matrix to get its Eigensystem
      lwrk = 34*N
      call dsyev('V','U',N,S,N,evals,wrk,lwrk,info)
      write(tdhfout,*) 'dsyev called'
```

```
        write(tdhfout,*) 'info:␣␣␣', info
        write(tdhfout,*) 'wrk(1):␣', wrk(1)
        write(tdhfout,*) ''

401     format(E12.5,'␣␣')
        write(tdhfout,*) 'Overlap␣Eigenvalues'
        do i=1, N
           write(tdhfout,401) evals(i)
        end do
        write(tdhfout,*) ''

        write(tdhfout,*) 'Overlap␣Eigenvectors'
        do i=1,N
           do j=1,N
              write(tdhfout,401,advance='no') S(i,j)
           end do
           write(tdhfout,*) ''
        end do
        write(tdhfout,*) ''

!       S is now the eigenvector matrix.
!       Populate the diagonal elements of Ssqrt with the square roots
!       of the eigenvalues. Simultaneously populate SevecsT as the
!       transpose of the eigenvector Matrix.
        do i=1,N
           do j=1,N
              SevecsT(i,j) = S(j,i)
              Ssqrt(i,j) = 0.0d0
           end do
           Ssqrt(i,i) = sqrt(evals(i))
        end do
```

```
853
          write(tdhfout,*) 'Overlap Root Eigenvalues'
          do i=1,N
             do j=1,N
                write(tdhfout,401,advance='no') Ssqrt(i,j)
858          end do
          end do
          write(tdhfout,*) ''

    !     Recover the overlap matrix root
863 !     S^(1/2) = V.s^(1/2).V^T
          Ssqrt = matmul(Ssqrt,SevecsT)
          Ssqrt = matmul(S,Ssqrt)

          write(tdhfout,*) 'Overlap Root Matrix'
868       do i=1,N
             do j=1,N
                write(tdhfout,401,advance='no') Ssqrt(i,j)
             end do
             write(tdhfout,*) ''
873       end do
          write(tdhfout,*) ''

    !     Perform final matrix multiplication according to
    !     Lowdin Population Analysis
878 !     P' = S^(1/2) . P . S^(1/2)
          rho2 = matmul(rho,Ssqrt)
          rho2 = matmul(Ssqrt,rho2)

          do i=1, N
883          pop(i) = zabs(rho2(i,i))
```

```
      end do

      return

888   end subroutine


893
!     ################################################################
!     ##                                                            ##
!     ##   writerestart - subroutine to save information for        ##
!     ##          resuming an old dynamics simulation               ##
898 !     ##                                                            ##
!     ################################################################

      subroutine writerestart

903   use iounit
      use orbits
      use piorbs
      use tdhfvars
      implicit none
908
      integer rout, freeunit
      integer i,j
      logical exist
      character*20 fname
913
      fname = 'restart/restart.data'
```

```
      rout = freeunit()

      inquire(file=fname,exist=exist)
918
      if(exist) then
         open(unit=rout,file=fname,status='old')
         rewind(rout)
      else
923      call system('mkdir restart')
         open(unit=rout,file=fname,status='new')
      end if

  501 format(A10,2X,I8)
928   502 format(A10,2X,L1)
      write(rout,501) "Iter: ",outiter
      write(rout,502) "Use URHF: ", useurhf
      write(rout,*)   ""

933   503 format(8E17.6)
      if(useurhf) then
         write(rout,*) 'Density Matrix (alpha, RE)'
         do i=1,norbit
            write(rout,503) (realpart(tdhfeda(i,j)),j=1,norbit)
938      end do
         write(rout,*) 'Density Matrix (alpha, IM)'
         do i=1,norbit
            write(rout,503) (aimag(tdhfeda(i,j)),j=1,norbit)
         end do
943
         write(rout,*) 'Density Matrix (beta, RE)'
         do i=1,norbit
```

```
            write(rout,503) (realpart(tdhfedb(i,j)),j=1,norbit)
         end do
         write(rout,*) 'Density Matrix (beta, IM)'
         do i=1,norbit
            write(rout,503) (aimag(tdhfedb(i,j)),j=1,norbit)
         end do

      else
         write(rout,*) 'Density Matrix (RE)'
         do i=1,norbit
            write(rout,503) (realpart(tdhfed(i,j)),j=1,norbit)
         end do
         write(rout,*) 'Density Matrix (IM)'
         do i=1,norbit
            write(rout,503) (aimag(tdhfed(i,j)),j=1,norbit)
         end do
      end if


      flush(rout)
      close(unit=rout)

      end subroutine writerestart


!     ##################################################################
!     ##                                                              ##
!     ##   tdhfloadold - subroutine to load old information           ##
```

```
!     ##          from previous dynamics simulation                 ##
!     ##                                                            ##
!     ################################################################

      subroutine tdhfload

      use iounit
      use files
      use orbits
      use piorbs
      use tdhfvars
      implicit none


      character*20  fname
      character*120 line
      integer io,freeunit,i,j
      real*16, allocatable :: temp1(:,:),temp2(:,:)
      complex*16 II
      logical exist

!     check if *.dyn exists. If not do not resume
      fname = filename(1:leng) // '.dyn'
      inquire(file=fname,exist=exist)
      if(.not.exist) then
         write(iout,*) ' Could not find old dynamics file'
         return
      end if

      II = (0.0d0,1.0d0)
      fname = "restart/restart.data"
```

```
1008        io=freeunit()
            open(unit=io,file=fname,action='read')

        501 format(A10,2X,I8)
        502 format(A10,2X,L1)
1013        read(io,501) line,outiter
            read(io,502) line,useurhf

    !        if(useurhf) print   *, 'Use unrestricted'
    !        if(.not.useurhf) print *,'Use restricted'
1018        read(io,*) line

            allocate(temp1(norbit,norbit))
            allocate(temp2(norbit,norbit))

1023        if(.not.allocated(tdhfhc)) then
               allocate (tdhfhc(norbit,norbit))
            end if
            if(.not.allocated(tdhfgamma)) then
               allocate (tdhfgamma(norbit,norbit))
1028        end if

        503 format(8E17.6)
            if(useurhf) then
               if(.not.allocated(tdhffocka)) then
1033              allocate(tdhffocka(norbit,norbit))
               end if
               if(.not.allocated(tdhffockb)) then
                  allocate(tdhffockb(norbit,norbit))
               end if
1038           if(.not.allocated(tdhfeda)) allocate(tdhfeda(norbit,norbit))
```

```
            if(.not.allocated(tdhfedb)) allocate(tdhfedb(norbit,norbit))

            do i=1,norbit
               read(io,503) (temp1(i,j),j=1,norbit)
            end do
            read(io,*) line
            do i=1,norbit
               read(io,503) (temp2(i,j),j=1,norbit)
            end do
            do i=1,norbit
               do j=1,norbit
                  tdhfeda(i,j) = temp1(i,j)+II*temp2(i,j)
               end do
            end do

            read(io,*) line
            do i=1,norbit
               read(io,503) (temp1(i,j),j=1,norbit)
            end do
            read(io,*) line
            do i=1,norbit
               read(io,503) (temp2(i,j),j=1,norbit)
            end do
            do i=1,norbit
               do j=1,norbit
                  tdhfedb(i,j) = temp1(i,j)+II*temp2(i,j)
               end do
            end do
         else
            if(.not.allocated(tdhfed))   allocate(tdhfed  (norbit,norbit))
            if(.not.allocated(tdhffock)) allocate(tdhffock(norbit,norbit))
```

```
            do i=1,norbit
               read(io,503) (temp1(i,j),j=1,norbit)
            end do
            read(io,*) line
            do i=1,norbit
               read(io,503) (temp2(i,j),j=1,norbit)
            end do
            do i=1,norbit
               do j=1,norbit
                  tdhfed(i,j) = temp1(i,j)+II*temp2(i,j)
               end do
            end do
         end if

         tdhffirst = .false.

         deallocate(temp1)
         deallocate(temp2)

         close(io)


         end subroutine
```

A.7 pitduhf.f

```
!     ##############################################################
!     ##  This code is written by Kush Patel.                     ##
!     ##                                                          ##
!     ##  Prints various matrices and calls the TDHF              ##
```

```
!     ##  iterator to update rho according to the            ##
!     ##  equation:                                          ##
!     ##                                                     ##
!     ##       (ih)(d rho/dt) = <[F(rho),rho]> = i L rho      ##
!     ##                                                     ##
!     #########################################################

      subroutine pitduhf
      use sizes
      use atoms
      use atomid
      use files
      use iounit
      use orbits
      use piorbs
      use bndstr
      use units
      use tdhfvars

      implicit none

!     pitdhf variables
      integer i,j,k,m
      integer info
      complex*16 tau(norbit),work(8*norbit),fockcopy(norbit,norbit)
      real*8 d(norbit),e(norbit-1), p
!     variables for reconstructing fock
      integer iorb,jorb
      integer iatn,jatn
      complex*16 s1a,s2a,s1b,s2b
      real*8 hcii, hcij
```

```
36        real*8 brij,erij
          real*8 bebond,eebond
          real*8 ebb,ebe,abnz,aeth,ble,blb
          real*8 xij,yij,zij,rij,rijsq
          real*8 gii,gij,g11,g11sq,g12,g14
41        real*8 ovlap,covlap
          real*8 cionize,iionize,jionize
          real*8 ip(norbit)
   !      variables for population analysis
          real*8 Smatrix(norbit,norbit), pop(norbit)
46 !      variables for iterative output
          integer lext
          character*7 ext
          integer freeunit
          logical exist
51 !      variables for total energy calculation
          complex*16 energy, paij, pbij
   !      Dipole calculation
          real*8 hcom(3)
          real*8 ecom(3)
56        real*8 qii
          real*8 dipl(3)
   !-------Testing-----------
   !      Seeing if original method of calculating energy
   !      results in regular fluctuations
61        complex*8 xi,xj,xk,xg,xcor
          complex*8 pii,pjj,pij
   !-------------------------
   !      Electron current
          real*8 currax, curray, curraz
66        real*8 currbx, currby, currbz
```

```
      real*8 cc1,cc2,cc3

  300 format(2E17.6, 3X)
  301 format(8E16.5)

!     open an output file
      outiter = outiter + 1
      printq = mod(outiter,outfreq).eq.0
      printq = printq.or.(outiter.eq.1)
      lext = 6
      if(printq) then
         call numeral(outiter,ext,lext)
         tdhfout = freeunit()
         inquire(file=filename//'.tdhf.'//ext(1:lext),exist=exist)
         if(exist) then
            open(unit=tdhfout,
     &       file=filename(1:leng)//'.tdhf.'//ext(1:lext),status='old')
            rewind(unit=tdhfout)
         else
            open(unit=tdhfout,
     &       file=filename(1:leng)//'.tdhf.'//ext(1:lext),status='new')
         end if

  309    format(A18,E16.5)
         write(tdhfout,*) '## Begin piTDHF iteration ##'
         write(tdhfout,*) ''
         write(tdhfout,309) ' Coupling radius: ', tdhfcr
         write(tdhfout,309) ' Time Step:       ', tdhfdt
         write(tdhfout,*) ''
      end if
```

```
!-------------------------------------------------------------
!     Following code develops the gamma matrix as well
!     as the hc matrix. These are the various integrals
!     and the Hatree Core matrix
!     Taken straight from the piscf subroutine
!     of picalc.f
!-------------------------------------------------------------
!     initialize some constants and parameters
      cionize = -11.16d0 / evolt

!     set the bond energies, alpha values and ideal bond length
!     parameter for carbon-carbon pibond type parameters
      ebe  = 129.37d0
      ebb  = 117.58d0
      aeth =   2.309d0
      abnz =   2.142d0
      ble  =   1.338d0
      blb  =   1.397d0

!     assign empirical one-center Coulomb integrals, and
!     first or second ionization potential depending on
!     whether the orbital contribures one or two electrons
      do i = 1, norbit
         iorb = iorbit(i)
         tdhfgamma(i,i) = emorb(iorb)
         ip(i) = worb(iorb) + (1.0d0-qorb(iorb))*emorb(iorb)
      end do

!     calculate two-center repulsion integrals
!     according to Ohno's semi-empirical formula
```

```
      do i = 1, norbit
         iorb = iorbit(i)
131      gii = tdhfgamma(i,i)
         do j = i+1, norbit
            jorb = iorbit(j)
            g11 = 0.5d0 * (gii+tdhfgamma(j,j))
            g11sq = 1.0d0 / g11**2
136         xij = x(iorb) - x(jorb)
            yij = y(iorb) - y(jorb)
            zij = z(iorb) - z(jorb)
            rijsq = (xij**2 + yij**2 + zij**2) / bohr**2
            g12 = 1.0d0 / sqrt(rijsq + g11sq)
141         tdhfgamma(i,j) = g12
            tdhfgamma(j,i) = g12
         end do
      end do

146   !     the first term in the sum to find alpha is the first
      !     or second ionization potential, then the two-center
      !     repulsion integrals are added
      do i = 1, norbit
         hcii = ip(i)
151      do j = 1, norbit
            if(i.ne.j) then
               jorb = iorbit(j)
               hcii = hcii - qorb(jorb)*tdhfgamma(i,j)
            end if
156      end do
         tdhfhc(i,i) = hcii
      end do
```

```
!     get two-center repulsion integrals via Ohno's formula
!      do k = 1, nbpi
!         i = ibpi(2,k)
!         j = ibpi(3,k)
       do i = 1, norbit-1
       do j = i+1, norbit
         iorb = iorbit(i)
         jorb = iorbit(j)
         iatn = atomic(iorb)
         jatn = atomic(jorb)
         xij = x(iorb) - x(jorb)
         yij = y(iorb) - y(jorb)
         zij = z(iorb) - z(jorb)
         rij = sqrt(xij**2 + yij**2 + zij**2)
         rijsq = rij**2 / bohr**2
         g11 = 0.5d0 * (tdhfgamma(i,i) + tdhfgamma(j,j))
         g11sq = 1.0d0 / g11**2
         g12 = tdhfgamma(i,j)

!     compute the bond energy using a Morse potential
         erij = aeth * (ble-rij)
         brij = abnz * (blb-rij)
         eebond = (2.0d0*exp(erij)-exp(2.0d0*erij)) * ebe / hartree
         bebond = (2.0d0*exp(brij)-exp(2.0d0*brij)) * ebb / hartree

!     compute the carbon-carbon resonance integral using
!     the Whitehead and Lo formula
         g14 = 1.0d0 / sqrt(4.0d0*rijsq+g11sq)
         hcij = 1.5d0*(bebond-eebond) - 0.375d0*g11
     &        + (5.0d0/12.0d0)*g12 - g14/24.0d0
```

```
!     if either atom is non-carbon, then factor the resonance
!     integral by overlap ratio and ionization potential ratio
         if(iatn.ne.6 .or. jatn.ne.6) then
            call overlap(iatn,jatn,rij,ovlap)
            call overlap(6,6,rij,covlap)
            hcij = hcij * (ovlap/covlap)
            iionize = ip(i)
            if(qorb(iorb) .ne. 1.0d0) then
               if(iatn .eq.  7) iionize = 0.595d0 * iionize
               if(iatn .eq.  8) iionize = 0.525d0 * iionize
               if(iatn .eq. 16) iionize = 0.890d0 * iionize
            end if
            jionize = ip(j)
            if(qorb(jorb) .ne. 1.0d0) then
               if(jatn .eq.  7) jionize = 0.595d0 * jionize
               if(jatn .eq.  8) jionize = 0.525d0 * jionize
               if(jatn .eq. 16) jionize = 0.890d0 * jionize
            end if
            hcij = hcij * (iionize+jionize)/(2.0d0*cionize)
         end if
!-----Test Code 11.04.16-------------------------------------
!     This code is the pitdhf copy of the similar thing in fullpi.f.
!     Applies Yukowa scaling to hcij terms of nonbonded atoms
!------------------------------------------------------------
         bonded = .false.
         do k=1, nbpi
            iorb = ibpi(2,k)
            jorb = ibpi(3,k)
            bonded = (iorb.eq.i .and. jorb.eq.j) .or. bonded
         end do
         if(.not.bonded) then
```

```
            hcij = hcij*(exp(-1.0d0*rij/tdhfcr))/rij
         end if
!-------------------------------------------------------------

!     set symmetric elements to the same value
         tdhfhc(i,j) = hcij
         tdhfhc(j,i) = hcij
      end do
      end do

!     reconstruct the Fock Matrix
      do i=1, norbit
         do j=1, norbit
            s1a = (0.0d0, 0.0d0) ! Total Density matrix element
            s1b = (0.0d0, 0.0d0) ! rho(k,k)*J(i,k)
            s2a = -1.0d0*tdhfgamma(i,j)*tdhfeda(i,j)
            s2b = -1.0d0*tdhfgamma(i,j)*tdhfedb(i,j)

            if(i.eq.j) then
               do k=1, norbit
                  s1a = tdhfeda(k,k) + tdhfedb(k,k)
                  s1b = s1b + s1a*tdhfgamma(i,k)
               end do
            end if

            tdhffocka(i,j) = tdhfhc(i,j) + s2a + s1b
            tdhffockb(i,j) = tdhfhc(i,j) + s2b + s1b
         end do
      end do

!
```

```
!     diagonalize the fock matrices to obtain eigenvalues
!     for the superoperator
!
      do m=1,2
!
!     print density matrices
!
         if(printq) then
            if(m.eq.1) then
               write(tdhfout,*) 'TDHF Electron Density Matrix (alpha)'
            else
               write(tdhfout,*) 'TDHF Electron Density Matrix (beta)'
            end if
            do i=1, norbit
               do j=1, norbit
                  if(m.eq.1) then
                     write(tdhfout,300,advance='no') tdhfeda(i,j)
                  else
                     write(tdhfout,300,advance='no') tdhfedb(i,j)
                  end if
               end do
               write(tdhfout,*) ''
            end do
            write(tdhfout,*) ''

!     Print the diagonal elements
 307        format(2E20.9,3X)
            if(m.eq.1) then
               write(tdhfout,*) 'TDHF Density Matrix Diagonal (alpha)'
            else
               write(tdhfout,*) 'TDHF Density Matrix Diagonal (beta)'
```

```
            end if
         end if
286      s1a = (0.0d0,0.0d0)
         do j=1, norbit
            if(m.eq.1) then
               s1a = s1a + tdhfeda(j,j)
               if(printq) write(tdhfout,307) tdhfeda(j,j)
291         else
               s1a = s1a + tdhfedb(j,j)
               if(printq) write(tdhfout,307) tdhfedb(j,j)
            end if
         end do
296
         if(printq) then
302         format(A12,2F17.6)
            write(tdhfout,*) ''
            write(tdhfout,302) 'Trace: ', s1a
301         write(tdhfout,*) ''

            s1a = (0.0d0,0.0d0)
            do i=1, norbit
               s1a = s1a + tdhfeda(i,i) + tdhfedb(i,i)
306         end do

            if(m.eq.2) then
303            format(A19,2F17.6)
               write(tdhfout,303) 'Trace (full): ', s1a
311            write(tdhfout,*) ''
            end if
         end if
!
```

```
!     copy fock matrix to tdhfcopy
!
      if(m.eq.1) then
         if(printq) write(tdhfout,*) 'Fock matrix (alpha)'
            do i=1, norbit
               do j=1, norbit
                  fockcopy(i,j) = tdhffocka(i,j)
                  if(printq) then
                     write(tdhfout,300,advance='no') tdhffocka(i,j)
                  end if
               end do
               if(printq) write(tdhfout,*) ''
            end do
         else
            if(printq) write(tdhfout,*) 'Fock matrix (beta)'
            do i=1, norbit
               do j=1, norbit
                  fockcopy(i,j) = tdhffockb(i,j)
                  if(printq) then
                     write(tdhfout,300,advance='no') tdhffockb(i,j)
                  end if
               end do
               if(printq) write(tdhfout,*) ''
            end do
         end if
         if(printq) write(tdhfout,*) ''
!
!     occasionally recalculate the eigenvalues
!
         if(mod(outiter,100).eq.1 .or. res1) then
            res1 = .false.
```

```
346                   if(tdhfdebug) then
                         write(debugout,*) 'Recalculating Super Eigenvalues'
                      end if
      !
      !     tridiagonalize
351   !
                call zhetrd('U',norbit,fockcopy,norbit,d,e,tau,
          &             work,8*norbit,info)
                if(tdhfdebug) then
                   write(debugout,*) 'Info: ', info
356                write(debugout,*) ''
                   write(debugout,*) 'Post Tridiagonal Fock'
                   do i=1,norbit
                      do j=1, norbit
                         if(m.eq.1) then
361                         write(debugout,300,advance='no') tdhffocka(i,j)
                         else
                            write(debugout,300,advance='no') tdhffockb(i,j)
                         end if
                      end do
366                   write(debugout,*)''
                   end do
                   write(debugout,*)''

                   write(debugout,*) 'tri-diagonal matrix'
371                write(debugout,*) 'diagonal     superdiagonal'
                   do i=1, norbit
                      write(debugout,301,advance='no') d(i)
                      if(i.ne.norbit) then
                         write(debugout,301,advance='no') e(i)
376                   end if
```

```
                write(debugout,*) ''
            end do
            write(debugout,*) ''
        end if

!     get the eigenvalues
        if(m.eq.1) then
            superevmax = 0.0d0
            superevmin = 0.0d0
        end if

        call zsteqr('N',norbit,d,e,'null',norbit,'null',info)
        if(printq) then
            if (m.eq.1) then
                write(tdhfout,*) 'Fock (alpha) Eigenvalues: '
            else
                write(tdhfout,*) 'Fock (beta) Eigenvalues: '
            end if
            write(tdhfout,301) (d(i),i=1,norbit)
            write(tdhfout,*)''
        end if
        superevmax = max(superevmax,d(norbit)-d(1))
        superevmin = min(superevmin,d(1)-d(norbit))
        if(printq) then
            write(tdhfout,*) 'Supereigenvalues: {', superevmin,', ',
     &           superevmax, '}'
            write(tdhfout,*) ''
        end if
        superevmax = 1.5d0*superevmax
        superevmin = 1.5d0*superevmin
```

```
            end if

    !
411 !       calculate total electronic energy
    !
            if(printq) then
               if(m.eq.2) then
                  energy = (0.0d0,0.0d0)
416               s1a = (0.0d0,0.0d0)
                  s1b = (0.0d0,0.0d0)
                  do i=1, norbit
                     do j=1, norbit
                        paij = tdhfeda(i,j)
421                     pbij = tdhfedb(i,j)
                        s1a = s1a + 0.5d0*paij*(tdhfhc(j,i)+tdhffocka(j,i))
                        s1b = s1b + 0.5d0*pbij*(tdhfhc(j,i)+tdhffockb(j,i))
                     end do
                  end do
426               energy = s1a + s1b
     305          format(A25,2E16.5)
                  write(tdhfout,305) 'Total alpha Energy: ', s1a
                  write(tdhfout,305) 'Total beta Energy: ' , s1b
                  write(tdhfout,305) 'Total Electronic Energy: ', energy
431               write(tdhfout,*) ''
               end if
            end if

            if(m.eq.2 .and. probcurr .and. printq) then
436            write(tdhfout,*) 'Probability Current'
               write(tdhfout,*) ' qx    qy    qz    ax    ay    az bx by bz'
               do i=1, norbit
```

```
                iorb = iorbit(i)
                currax = 0.0d0
                curray = 0.0d0
                curraz = 0.0d0
                currbx = 0.0d0
                currby = 0.0d0
                currbz = 0.0d0
                do j=1, norbit
                   jorb = iorbit(j)
                   cc1 = 2.0d0*tdhfhc(j,i)*aimag(tdhfeda(j,i))
                   cc2 = 2.0d0*tdhfhc(j,i)*aimag(tdhfedb(j,i))
                   currax = currax + cc1* (x(jorb)-x(iorb))
                   curray = curray + cc1* (y(jorb)-y(iorb))
                   curraz = curraz + cc1* (z(jorb)-z(iorb))
                   currbx = currbx + cc2* (x(jorb)-x(iorb))
                   currby = currby + cc2* (y(jorb)-y(iorb))
                   currbz = currbz + cc2* (z(jorb)-z(iorb))
                end do
  306           format(9E16.5)
                write(tdhfout,306) x(iorb),y(iorb),z(iorb),currax,curray,
     &               curraz,currbx,currby,currbz
              end do
            end if
          end do

  !       call tdhfiterate and update rho
          call tdhfiterate(tdhfeda,tdhffocka,superevmax,
     &      superevmin,tdhfdt,norbit,9)
          call tdhfiterate(tdhfedb,tdhffockb,superevmax,
     &      superevmin,tdhfdt,norbit,9)
```

```
!-----Test Code 10 Dec 17-----------------------------------
!      Impose zero for imaginary parts on diagonal
       do i=1,norbit
          tdhfeda(i,i) = realpart(tdhfeda(i,i))
          tdhfedb(i,i) = realpart(tdhfedb(i,i))
       end do
!-------------------------------------------------------------

!
!      update the nonplanar pi bond orders (pnpl)
!      this is the same code that's at the end of picalc
!
 401   format(5i5,2f12.2)
 402   format(3A5,2A15)
       if(printq) write(tdhfout,402) 'Bond','i','j','old','new'
       do k=1, nbpi
          i = ibpi(2,k)
          j = ibpi(3,k)
          p = pnpl(k)

!         take just the real part of electron density
!          pnpl(k) = realpart( tdhfed(i,j) )
!         take the real part of the electron density
!          pnpl(k) = zabs( tdhfeda(i,j) + tdhfedb(i,j))
          pnpl(k) = realpart( tdhfeda(i,j) + tdhfedb(i,j) )
          pbpl(k) = pnpl(k) * tdhfhc(i,j)/(-0.0757d0)

          i = ibnd(1,ibpi(1,k))
          j = ibnd(2,ibpi(1,k))
          if(printq)
```

```
501       &         write(tdhfout,401) k,ibpi(2,k),ibpi(3,k),i,j,p,pnpl(k)
          end do
          if(printq) write(tdhfout,*)''

    !-------------------------
506       if(printq) then
             xi = 0.0d0
             xj = 0.0d0
             xk = 0.0d0
             xg = 0.0d0
511          xcor = 0.0d0

             write(tdhfout,*) 'Original Type Energy Calculation'
             do i=1, norbit
                do j=1, norbit
516                 pii = tdhfeda(i,i) + tdhfedb(i,i)
                    pjj = tdhfeda(j,j) + tdhfedb(j,j)
                    pij = tdhfeda(i,j) + tdhfedb(i,j)
                    xi = xi + pij*tdhfhc(i,j)
                    xj = xj + 0.50d0*pij*tdhfgamma(i,j)
521                 xk = xk - 0.25d0*pij*tdhfgamma(i,j)
                    xcor = xcor - 0.5d0*(pij*pij - pii*pjj)*tdhfgamma(i,j)
                end do
             end do
             do i=1, norbit-1
526             do j=i+1, norbit
                    xg = xg + tdhfgamma(i,j)
                end do
             end do
             write(tdhfout,*) 'OEnergy:  ', xi+xj+xk+xg+xcor
531          write(tdhfout,*) 'Core:     ', xi
```

```
            write(tdhfout,*) 'Coulomb:  ', xj
            write(tdhfout,*) 'Exchange: ', xk
            write(tdhfout,*) 'Nuclear:  ', xg
            write(tdhfout,*) 'Xcorrect: ', xcor
!-------------------------

            write(tdhfout,*)
            write(tdhfout,*)'## End of piTDHF iteration ##'
            write(tdhfout,*)''
            write(tdhfout,*)''

!     Close the output file
            flush(tdhfout)
            close(unit=tdhfout)

!     Write restart information
            call writerestart()

        end if

        end subroutine
```

A.8 redirects.f

```
!     The following lines of code must be inserted into the indicated
!     files. This code is written for TINKER version 7.1.2. Users must
!     ensure compatibility with newer versions.

!========================
!     dynamic.f
!========================
```

```
!     insert this section in the header to inlude usage of variables
!--------------------
      use tdhfvars
      use civars
!--------------------

!     insert this section before the integration steps are started
!-----------------------------------------------------------------------
!     call tdhfinit to see if TDHF is requested and initialize necessary
!     parameters
!
      call tdhfinit
!     Copy dt to a variable TDHF will use
      if(usetdhf) then
         tdhfdt = dt
      end if
!-----------------------------------------------------------------------
!     call ciinit to see if CI theory/excited calculations are requested
!     and initialize the necessary parameters
c
      call ciinit
!-----------------------------------------------------------------------



!========================
!     picalc.f
!========================
!     insert this section in the header to inlude usage of variables
!--------------------
      use tdhfvars
      use civars
```

```
!--------------------

!     insert this section immediately after the number of calls to the
!     picalc routine is incremented and before quantum calculations
!     begin
!-----------------------------------------------------------------------
!     If using TDHF or CI theory call fullpi instead
      if(usetdhf.or.usesinglet.or.usetriplet) then
         call fullpi
         return
      end if
!-----------------------------------------------------------------------
```

A.9 tdhfinit.f

```
!
!     ######################################################
!     ##          Written by Kush Patel - 2/01/16         ##
!     ######################################################
!
!     ################################################################
!     ##                                                            ##
!     ##  subroutine tdhfinit  --  checks for keyword tdhf and      ##
!     ##                           and initializes necessary        ##
!     ##                           values                           ##
!     ##                                                            ##
!     ################################################################
!
!     "tdhfinit" checks for the keyword "TDHF" which decides
!     whether the tchebychev propagator will be used or not
!
```

```
      subroutine tdhfinit
      use sizes
      use keys
      use tdhfvars
      logical exist, resumeq
      integer i, next, freeunit
      character*20 keyword
      character*120 record, string

      usetdhf = .false.
      tdhfdebug = .false.
      printq = .false.
      probcurr = .false.
      useurhf = .false.
      resumeq = .false.
      res1 = .false.
      outiter = 0
      outfreq = 1

      do i=1,nkey
         next = 1
         record = keyline(i)
         call gettext (record,keyword,next)
         call upcase (keyword)
         if(keyword(1:5) .eq. 'TDHF ') then
            usetdhf  = .true.
            tdhffirst = .true.
         else if(keyword(1:7) .eq. 'TDHFCR ') then
            string = record(next:120)
            tdhfcr = 0.0d0
```

```
            read(string,*,err=10,end=10) tdhfcr
 10         continue
50        else if(keyword(1:10).eq.'TDHFDEBUG␣') then
            tdhfdebug = .true.
            debugout = freeunit()
!     open the debugging output
            inquire(file='tdhf.debug',exist=exist)
55          if(exist) then
               open(unit=debugout,file='tdhf.debug',status='old')
               rewind(unit=debugout)
            else
               open(unit=debugout,file='tdhf.debug',status='new')
60          end if
          else if(keyword(1:8).eq.'USEURHF␣') then
            useurhf = .true.
          else if(keyword(1:11).eq.'PRINTEVERY␣') then
            string = record(next:120)
65          read(string,*,err=20,end=20) outfreq
            if(outfreq.lt.1) outfreq = 1
 20         continue
          else if(keyword(1:9).eq.'PROBCURR␣') then
            probcurr = .true.
70        else if(keyword(1:8).eq.'RESUME␣') then
            resumeq = .true.
            res1 = .true.
          end if
        end do
75
        if(resumeq) call tdhfload()

        return
```

```
      end
```

A.10 tdhfvars.f

```
!
!     ################################################################
!     ##                                                            ##
!     ##  module tdhfvars  --  contents of the PITDHF               ##
!     ##                       calculation                          ##
!     ##                                                            ##
!     ################################################################
!
!
!     maxkey    maximum number of lines in the keyword file
!
!     nkey      number of nonblank lines in the keyword file
!     keyline   contents of each individual keyword file line
!
!     usetdhf     logical that indicates whether TDHF is used
!     tdhffirst   logical switch that indicates whether the dynamics
!                 are in the first time step
!     tdhfdebug   logical for deciding whether to print debugging
!                 information
!     tdhfdt      time step for dynamics
!     tdhfout     output file unit number
!     tdhf_ nfill fill level for the orbitals (# of electrons / 2)
!     tdhfeda     tdhf electron density matrix (spin up)
!     tdhfedb     tdhf electron density matrix (spin down)
!     tdhffocka   tdhf fock matrix (spin up)
!     tdhffockb   tdhf fock matrix (spin down)
!     tdhfgamma   tdhf gamma matrix
```

```
!     tdhfh!      tdhf core matrix
!     bonded      logical that indicates whether a pair of atoms
!                 are bonded
!     tdhfcr      coupling radius for the exponetial scaling term (Angstrom)


      module tdhfvars
      implicit none
      logical usetdhf, tdhffirst, tdhfdebug, printq
      logical probcurr, useurhf, res1
      integer tdhfout, debugout, tdhf_nfill, outiter
      integer outfreq
      real*8  tdhfdt, superevmin, superevmax
      complex*16, allocatable :: tdhfeda(:,:)
      complex*16, allocatable :: tdhfedb(:,:)
      complex*16, allocatable :: tdhffocka(:,:)
      complex*16, allocatable :: tdhffockb(:,:)
      complex*16, allocatable :: tdhfgamma(:,:)
      complex*16, allocatable :: tdhfhc(:,:)

      logical bonded
      real*8  tdhfcr

      complex*16, allocatable :: tdhfed(:,:)
      complex*16, allocatable :: tdhffock(:,:)

      save
      end
```

A.11 mpacf.py

```
"""
Program for computing velocity autocorrelation of TINKER
files (multiprocessor)

Written by Kush Patel, May 2019


Tinker velocity files have names in the form:
  root.###v

Input for this program is a JSON file with the following
parameters defined:
  root (string)
   - Root title of the TINKER coordinate file
  outtag (string)
   - User defined output label for section of system
     autocorrelated
   - output file will be named "[root].[outtag].acf"
  atoms (array of integers)
   - Indices of atoms to consider in autocorrelation

Example:
[{
   "root": "TPP",
   "outtag": "diaryl",
   "atoms: [35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45,
            57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67]
}]


------------------------------------------------------------
```

```
! This code is non-general and written for use in a specific
! directory and ensemble nomenclature. Furthermore, it
! includes velocity files of a prior calculation for
! pre-excitation dynamics. Users should adjust this code to
! their directory setup.
-----------------------------------------------------------
"""


#!/usr/bin/env python


import multiprocessing as mp
import time
import os
import glob
import sys
import numpy as np
import json

def autocorr( tau ):
        sm = 0.0
        for i in range(0,nfiles-tau):
                for j in atomlist:
                        sm += np.sum( vels[i,j]*vels[i+tau,j] )
        return sm/(nfiles-tau)

def importvel(fn):
        tfn = "temp.vel."+str(os.getpid())
        os.system('cp␣'+fn+'␣'+tfn )
        os.system("sed␣-i␣'s/D/e/g'␣"+tfn)
```

```
        vels = np.loadtxt(tfn,skiprows=1,usecols=(2,3,4))
        os.remove(tfn)
        return vels

if __name__ == '__main__':
        print("Loading parameters..")
        jsonfile = sys.argv[1]
        with open(jsonfile) as jf: jstr = jf.read()
        prms = json.loads(jstr)[0]

        try:
                root = prms["root"]
        except KeyError:
                print("No root name found")
                sys.exit()
        try:
                alst = prms["atoms"]
        except KeyError:
                alst = -1
        try:
                outtag = "."+prms["outtag"]
        except KeyError:
                outtag = ""

        print("parameters found:")
        print("root          : " + root)

        try:
                nprocs = int(os.environ["SLURM_JOBS_CPUS_PER_NODE"])
        except KeyError:
                nprocs = 16
```

```
        print( "nprocs␣␣␣␣␣␣␣␣:␣"+ str(nprocs)  )

95

        simdirs = glob.glob("../"+root+"/sim0*/")
        simdirs = [ str(sd) for sd in simdirs ]

100     timefiles = [ str( root+"."+str(i).zfill(3)+"v" )
                for i in range(1,10001) ]

        for sd in simdirs:
                simitr = sd.split('/')[2]
105             simitr = int(simitr[4:])

                shortitr = str(simitr/100)
                outfile  = root+"/"+root+"."+shortitr+outtag+".acf"

110             # check if the output file exists
                # if so, proceed. (notify main out)
                # otherwise, perform the ACF
                if os.path.isfile(outfile):
                        print( outfile + "␣already␣exists.␣Skipping..." )
115                     continue

                prefiles = [ str( root+"pre/"+root+"."+str(i).zfill(5)+"v" )
                        for i in range(simitr-2000,simitr) ]
                localfiles = [ sd+f for f in timefiles  ]
120             velfiles = prefiles + localfiles

                global vels
                t1 = time.time()
                p = mp.Pool(processes=nprocs)
```

```
vels = np.asarray(p.map( importvel, velfiles ))
p.terminate()
t2 = time.time()
print('load time     : ' + str(t2-t1))


natoms = len(vels[0])
print("natoms        : " + str(natoms))

global atomlist
if alst==-1 or len(alst)<1:
        atomlist = range(natoms)
else:
        atomlist = alst
print("atomlist      : ")
print(np.asarray(atomlist))


global nfiles
nfiles = len(vels)
p = mp.Pool(processes=nprocs)
acf = p.map(autocorr, range(nfiles))
p.terminate()

wrtr = open(outfile,"w")
for i in range(nfiles):
        wrtr.write("%7i %15.6e\n" % (i,acf[i]/acf[0]) )
wrtr.close()
```

Bibliography

[1] Rudolph A. Marcus. On the theory of oxidation-reduction reactions involving electron transfer. I. *The Journal of Chemical Physics*, 24(5):966–978, 1956.

[2] Rudolph A. Marcus. Electrostatic free energy and other properties of states having nonequilibrium polarization. I. *The Journal of Chemical Physics*, 24(5):979–989, 1956.

[3] Rudolph A. Marcus. Electron transfer reactions in chemistry theory and experiment. *Journal of Electroanalytical Chemistry*, 438(1-2):251–259, 1997.

[4] Rudolph A. Marcus. On the theory of oxidation-reduction reactions involving election transfer v. comparison and properties of electrochemical and chemical rate constants - correction. *The Journal of Physical Chemistry*, 67(12):2889–2889, 1963.

[5] Arun K. Manna and Barry D. Dunietz. Communication: Charge-transfer rate constants in zinc-porphyrin-porphyrin-derived dyads: A Fermi golden rule first-principles-based study. *Journal of Chemical Physics*, 141(12), 2014.

[6] Arun K. Manna, Desinghu Balamurugan, Margaret S. Cheung, and Barry D. Dunietz. Unraveling the mechanism of photoinduced charge transfer in carotenoid-porphyrin-C60 molecular triad. *Journal of Physical Chemistry Letters*, 6(7):1231–1237, 2015.

[7] Xunmo Yang and Eric R. Bittner. Intramolecular charge- and energy-transfer rates with reduced modes: Comparison to marcus theory for donor-bridge-acceptor systems. *Journal of Physical Chemistry A*, 118(28):5196–5203, 2014.

[8] Rudolph A. Marcus. Exchange reactions and electron transfer reactions including isotopic exchange. Theory of oxidation-reduction reactions involving electron transfer. Part 4.—A statistical-mechanical basis for treating contributions from solvent, ligands, and inert salt. *Discuss. Faraday Soc.*, 29(i):21–31, 1960.

[9] John R. Miller. Intramolecular long-distance electron transfer in radical anions. the effects of free energy and solvent on the reaction rates. *Journal of the American Chemical Society*, 106(10):3047–3049, 1984.

[10] John C. Tully. Molecular dynamics with electronic transitions. *The Journal of Chemical Physics*, 93(2):1061–1071, 1990.

[11] Paul Ehrenfest. Bemerkung über die angenäherte Gültigkeit der klassischen Mechanik innerhalb der Quantenmechanik. *Zeitschrift für Physik*, 45(7-8):455–457, 1927.

[12] John C. Tully and Richard K. Preston. Trajectory surface hopping approach to nonadiabatic molecular collisions: the reaction of H + with D 2. *The Journal of Chemical Physics*, 55(2):562–572, 1971.

[13] Normand C. Blais and Donald G. Truhlar. Trajectory-surface-hopping study of Na(3p2P)+H2→Na(3s2S)+H2(v′,j′,θ). *The Journal of Chemical Physics*, 79(3):1334–1342, 1983.

[14] Gérard Parlant and Eric A Gislason. An exact trajectory surface hopping procedure: Comparison with exact quantal calculations. *The Journal of Chemical Physics*, 91(7):4416–4418, 1989.

[15] David F. Coker and Li Xiao. Methods for molecular dynamics with nonadiabatic transitions. *The Journal of Chemical Physics*, 102(1):496–510, 1995.

[16] Oleg V. Prezhdo and Peter J. Rossky. Mean-field molecular dynamics with surface hopping. *The Journal of Chemical Physics*, 107(3):825–834, 1997.

[17] Craig C. Martens. Surface hopping without momentum jumps: a quantum-trajectory-based approach to nonadiabatic dynamics. *Journal of Physical Chemistry A*, 123(5):1110–1128, 2019.

[18] Uwe Müller and Gerhard Stock. Surface-hopping modeling of photoinduced relaxation dynamics on coupled potential-energy surfaces. *Journal of Chemical Physics*, 107(16):6230–6245, 1997.

[19] Jian Yun Fang and Sharon Hammes-Schiffer. Improvement of the internal consistency in trajectory surface hopping. *Journal of Physical Chemistry A*, 103(47):9399–9407, 1999.

[20] Nathan S. Lewis and Daniel G. Nocera. Powering the planet: Chemical challenges in solar energy utilization. *Proceedings of the National Academy of Sciences*, 103(43):15729–15735, 2006.

[21] Lingxian Meng, Yamin Zhang, Xiangjian Wan, Chenxi Li, Xin Zhang, Yanbo Wang, Xin Ke, Zuo Xiao, Liming Ding, Ruoxi Xia, Hin-Lap Yip, Yong Cao, and Yongsheng Chen. Organic and solution-processed tandem solar cells with 17.3% efficiency. *Science (New York, N.Y.)*, 361(6407):1094–1098, 2018.

[22] Best Research-Cell Efficiency Chart, 2019.

[23] Martin Knupfer. Exciton binding energies in organic semiconductors. *Applied Physics A: Materials Science and Processing*, 77(5):623–626, 2003.

[24] Allen Kelley, Kush Patel, and Eric R. Bittner. Quantum simulations of charge separation at a model donor-acceptor interface: Role of delocalization and local packing. *Advances in Condensed Matter Physics*, 2018(December):1–10, 2018.

[25] Xiche Hu, Ana Damjanovic, Thorsten Ritz, and Klaus Schulten. Architecture and mechanism of the light-harvesting apparatus of purple bacteria. *Proceedings of the National Academy of Sciences*, 95(11):5935–5941, 1998.

[26] Kenneth R. Miller. Three-dimensional structure of a photosynthetic membrane. *Nature*, 300(5887):53–55, 1982.

[27] Thomas Walz and Robin Ghosh. Two-dimensional crystallization of the light-harvesting I-reaction centre photounit from Rhodospirillum rubrum. *Journal of Molecular Biology*, 265(2):107–111, 1997.

[28] Teresa G. Monger and William W. Parson. Singlet-triplet fusion in Rhodopseudomonas sphaeroides chromatophores. A probe of the organization of the photosynthetic apparatus. *BBA - Bioenergetics*, 460(3):393–407, 1977.

[29] Lothar Germeroth, Hartmut Michel, Friedrich Lottspeich, and Bruno. Robert. Unexpected similarities of the B800–850 light-harvesting complex from rhodospirillum molischianum to the B870 light-harvesting complexes from other purple photosynthetic bacteria. *Biochemistry*, 32(21):5615–5621, 1993.

[30] Jeanette Aagaard and William R. Sistrom. Control of synthesis of reaction center bacteriochlorophyll in photosynthetic bacteria. *Photochemistry and Photobiology*, 15(2):209–225, 1972.

[31] Rienk van Grondelle, Jan P. Dekker, Tomas Gillbro, and Villy Sundstrom. Energy transfer and trapping in photosynthesis. *BBA - Bioenergetics*, 1187(1):1–65, 1994.

[32] Johann Deisenhofer and James R. Norris, editors. *The Photosynthetic Reaction Center, Volume II*. Elsevier, 1993.

[33] Steven G. Boxer and Gerhard L. Closs. A covalently bound dimeric derivative of pyrochlorophyllide a. a possible model for reaction center chlorophyll. *Journal of the American Chemical Society*, 98(17):5406–5408, 1976.

[34] Michael R. Wasielewski, Martin H. Studier, and Joseph J. Katz. Covalently linked chlorophyll a dimer: A biomimetic model of special pair chlorophyll. *Proceedings of the National Academy of Sciences*, 73(12):4282–4286, 1976.

[35] Devens Gust, Thomas A. Moore, Paul A. Liddell, Gregory A. Nemeth, Lewis R. Makings, Ana L. Moore, Donna Barrett, Peter J. Pessiki, and Rene V. Bensasson. Charge separation in carotenoporphyrin-quinone triads: synthetic, conformational, and fluorescence lifetime studies. *Journal of the American Chemical Society*, 109(3):846–856, 1987.

[36] Alisdair N. Macpherson, Paul A. Liddell, Su Lin, Lori Noss, Gilbert R. Seely, Janice M. DeGraziano, Ana L. Moore, Thomas A. Moore, and Devens Gust. Ultrafast Photoinduced electron transfer in rigid porphyrin-quinone dyads. *Journal of the American Chemical Society*, 117(27):7202–7212, 1995.

[37] Darius Kuciauskas, Su Lin, Gilbert R. Seely, Ana L. Moore, Thomas A. Moore, Devens Gust, Tatiana Drovetskaya, Christopher A. Reed, and Peter D.W. Boyd. Energy and photoinduced electron transfer in porphyrin-fullerene dyads. *Journal of Physical Chemistry*, 100(39):15926–15932, 1996.

[38] Paul A. Liddell, Darius Kuciauskas, John P. Sumida, Boaz Nash, Dorothy Nguyen, Ana L. Moore, Thomas A. Moore, and Devens Gust. Photoinduced charge separation and charge recombination to a triplet state in a carotene-porphyrin-fullerene triad. *Journal of the American Chemical Society*, 119(6):1400–1405, 1997.

[39] Jeffrey L. Bahr, Darius Kuciauskas, Paul A. Liddell, Ana L. Moore, Thomas A. Moore, and Devens Gust. Driving force and electronic coupling effects on photoinduced electron transfer in a fullerene-based molecular triad. *Photochemistry and Photobiology*, 72(5):598, 2000.

[40] Darius Kuciauskas, Paul A Liddell, Su Lin, Simon G Stone, Ana L. Moore, Thomas A Moore, and Devens Gust. Photoinduced Electron Transfer in Carotenoporphyrin-Fullerene Triads: Temperature and Solvent Effects. *The Journal of Physical Chemistry B*, 104(18):4307–4321, 2000.

[41] Devens Gust, Thomas A. Moore, and Ana L. Moore. Mimicking Photosynthetic Solar Energy Transduction. *Accounts of Chemical Research*, 34(1):40–48, 2001.

[42] Sergei N. Smirnov, Paul A. Liddell, Ivan V. Vlassiouk, Alexey Teslja, Darius Kuciauskas, Charles L. Braun, Ana L. Moore, Thomas A. Moore, and Devens Gust. Characterization of the giant transient dipole generated by photoinduced electron transfer in a carotene-porphyrin-fullerene molecular triad. *The Journal of Physical Chemistry A*, 107(38):7567–7573, 2003.

[43] Gerdenis Kodis, Paul A. Liddell, Ana L. Moore, Thomas A. Moore, and Devens Gust. Synthesis and photochemistry of a carotene-porphyrin-fullerene model photosynthetic reaction center. *Journal of Physical Organic Chemistry*, 17(9):724–734, 2004.

[44] Darius Kuciauskas, Paul A. Liddell, Su-Chun Hung, Su Lin, Simon Stone, Gilbert R. Seely, Ana L. Moore, Thomas A. Moore, and Devens Gust. Structural effects on photoinduced electron transfer in carotenoid-porphyrin-quinone triads. *The Journal of Physical Chemistry B*, 101(3):429–440, 2002.

[45] Donatella Carbonera, Marilena Di Valentin, Carlo Corvaja, Giancarlo Agostini, Giovanni Giacometti, Paul A. Liddell, Darius Kuciauskas, Ana L. Moore, Thomas A. Moore, and Devens Gust. EPR investigation of photoinduced radical pair formation and decay to a triple state in a carotene-porphyrin-fullerene triad. *Journal of the American Chemical Society*, 120(18):4398–4405, 1998.

[46] Carlo Andrea Rozzi, Sarah Maria Falke, Nicola Spallanzani, Angel Rubio, Elisa Molinari, Daniele Brida, Margherita Maiuri, Giulio Cerullo, Heiko Schramm, Jens Christoffers, and Christoph Lienau. Quantum coherence controls the charge separation in a prototypical artificial light-harvesting system. *Nature Communications*, 4(1):1602, 2013.

[47] Miguel Marques. `octopus`: a first-principles tool for excited electron–ion dynamics. *Computer Physics Communications*, 151(1):60–78, 2003.

[48] Wendu Ding, Christian F. A. Negre, Leslie Vogt, and Victor S. Batista. Single molecule rectification induced by the asymmetry of a single frontier orbital. *Journal of Chemical Theory and Computation*, 10(8):3393–3400, 2014.

[49] David Zsolt Manrique, Cancan Huang, Masoud Baghernejad, Xiaotao Zhao, Oday A. Al-Owaedi, Hatef Sadeghi, Veerabhadrarao Kaliginedi, Wenjing Hong, Murat Gulcur, Thomas Wandlowski, Martin R. Bryce, and Colin J. Lambert. A quantum circuit rule for interference effects in single-molecule electrical junctions. *Nature Communications*, 6:1–8, 2015.

[50] Xiang Sun, Pengzhi Zhang, Yifan Lai, Kyle L. Williams, Margaret S. Cheung, Barry D. Dunietz, and Eitan Geva. Computational study of charge-transfer dynamics in the carotenoid-porphyrin-c60 molecular triad solvated in explicit tetrahydrofuran and its spectroscopic signature. *Journal of Physical Chemistry C*, 122(21):11288–11299, 2018.

[51] Qiang Shi and Eitan Geva. Nonradiative electronic relaxation rate constants from approximations based on linearizing the path-integral forward-backward action. *Journal of Physical Chemistry A*, 108(29):6109–6116, 2004.

[52] Arieh Aviram and Mark A. Ratner. Molecular rectifiers. *Chemical Physics Letters*, 29(2):277–283, 1974.

[53] Robert M. Metzger. Unimolecular electrical rectifiers. *Chemical Reviews*, 103(9):3803–3834, 2003.

[54] Rudolph Pariser and Robert G. Parr. A semi-empirical theory of the electronic spectra and electronic structure of complex unsaturated molecules. II. *The Journal of Chemical Physics*, 21(5):767–776, 1953.

[55] John A. Pople. Electron interaction in unsaturated hydrocarbons. *Transactions of the Faraday Society*, 49:1375, 1953.

[56] Jay W. Ponder and Frederic M. Richards. An efficient newton-like method for molecular mechanics energy minimization of large molecules. *Journal of Computational Chemistry*, 8(7):1016–1024, 1987.

[57] Attila Szabo and Neil Ostlund. *Modern quantum chemistry : introduction to advanced electronic structure theory / Attila Szabo, Neil S. Ostlund*. Dover Publications, Inc., Mineola, first edition, 1996.

[58] Eric R. Bittner. *Quantum dynamics: Applications in biological and materials systems*. CRC Press, 2009.

[59] Hillel Tal-Ezer and Ronnie Kosloff. An accurate and efficient scheme for propagating the time dependent Schrodinger equation. *The Journal of Chemical Physics*, 81(9):3967, 1984.

[60] Ronnie Kosloff. Time-dependent quantum-mechanical methods for molecular dynamics. *Journal of Physical Chemistry*, 92(8):2087–2100, 1988.

[61] Donald H. Lo and Michael A. Whitehead. Accurate heats of atomization and accurate bond lengths.: I. Benzenoid hydrocarbons. *Canadian Journal of Chemistry*, 46(12):2027–2040, 1968.

[62] Marcus D. Hanwell, Donald E. Curtis, David C. Lonie, Tim Vandermeersch, Eva Zurek, and Geoffrey R. Hutchison. Avogadro: an advanced semantic chemical editor, visualization, and analysis platform. *Journal of Cheminformatics*, 4(1):17, 2012.

[63] Norman L. Allinger, Young H. Yuh, and Jenn Huei Lii. Molecular mechanics. The MM3 force field for hydrocarbons. 1. *Journal of the American Chemical Society*, 111(23):8551–8566, 1989.

[64] Craig C. Martens. Nonstationary time-series analysis of many-body dynamics. *Physical Review A*, 45(9):6914–6917, 1992.

[65] Yusuke Yamauchi, Hiromi Nakai, and Yoshiki Okada. Short-time Fourier transform analysis of ab initio molecular dynamics simulation: Collision reaction between NH4 +(NH 3)2 and NH3. *Journal of Chemical Physics*, 121(22):11098–11103, 2004.

[66] Mari Tamaoki, Yusuke Yamauchi, and Hiromi Nakai. Short-time fourier transform analysis of Ab initio molecular dynamics simulation: Collision reaction between CN and C4H6. *Journal of Computational Chemistry*, 26(5):436–442, 2005.

[67] Yusuke Yamauchi and Hiromi Nakai. Hybrid approach for ab initio molecular dynamics simulation combining energy density analysis and short-time Fourier transform: Energy transfer spectrogram. *The Journal of Chemical Physics*, 123(3):034101, 2005.

[68] Takao Otsuka and Hiromi Nakai. Wavelet transform analysis ofab initio molecular dynamics simulation: Application to core-excitation dynamics of BF3. *Journal of Computational Chemistry*, 28(6):1137–1144, 2007.

[69] Yusuke Yamauchi, Shiho Ozawa, and Hiromi Nakai. Ab initio molecular dynamics simulation of the energy-relaxation process of the protonated water dimer. *The Journal of Physical Chemistry A*, 111(11):2062–2066, 2007.

[70] Tomoko Akama and Hiromi Nakai. Short-time Fourier transform analysis of real-time time-dependent Hartree-Fock and time-dependent density functional theory calculations with Gaussian basis functions. *Journal of Chemical Physics*, 132(5), 2010.

[71] Jaideva C. Goswami and Andrew K. Chan. *Fundamentals of Wavelets: Theory, Algorithms, and Applications*. Wiley, New York, 1999.

[72] Stephane Mallat. *A Wavelet Tour of Signal Processing*. Elsevier, San Diego, second edition, 2009.

[73] Tunna Baruah and Mark R. Pederson. Density functional study on a light-harvesting carotenoid-porphyrin-C60 molecular triad. *Journal of Chemical Physics*, 125(16):1–6, 2006.

[74] Metin Aydin. Comparative study of the structural and vibroelectronic properties of porphyrin and its derivatives. *Molecules*, 19(12):20988–21021, 2014.

[75] Gregory D. Scholes. Quantum-coherent electronic energy transfer: Did nature think of it first? *Journal of Physical Chemistry Letters*, 1(1):2–8, 2010.

[76] Eric R. Bittner and Carlos Silva. Noise-induced quantum coherence drives photocarrier generation dynamics at polymeric semiconductor heterojunctions. *Nature Communications*, 5, 2014.

[77] Guangqi Li, Niranjan Govind, Mark A. Ratner, Christopher J. Cramer, and Laura Gagliardi. Influence of Coherent tunneling and incoherent hopping on the charge transfer mechanism in linear donor-bridge-acceptor systems. *Journal of Physical Chemistry Letters*, 6(24):4889–4897, 2015.

[78] Krystyna E. Wilk, Stephen J. Harrop, Lucy Jankova, Diana Edler, Gary Keenan, Francis Sharples, Roger G. Hiller, and Paul M. G. Curmi. Evolution of a light-harvesting protein by addition of new subunits and rearrangement of conserved elements: Crystal structure of a cryptophyte phycoerythrin at 1.63-A resolution. *Proceedings of the National Academy of Sciences*, 96(16):8901–8906, 1999.

[79] Tiago Barros and Werner Kühlbrandt. Crystallisation, structure and function of plant light-harvesting Complex II. *Biochimica et Biophysica Acta (BBA) - Bioenergetics*, 1787(6):753–772, 2009.

[80] Elisabetta Collini, Cathy Y. Wong, Krystyna E. Wilk, Paul M. G. Curmi, Paul Brumer, and Gregory D. Scholes. Coherently wired light-harvesting in photosynthetic marine algae at ambient temperature. *Nature*, 463(7281):644–647, 2010.

[81] Vladimir Lankevich and Eric R. Bittner. Relating free energy and open-circuit voltage to disorder in organic photovoltaic systems. *Journal of Chemical Physics*, 149(24), 2018.

[82] Junzi Liu, Yong Zhang, and Wenjian Liu. Photoexcitation of light-harvesting C-P-C60 triads: A FLMO-TD-DFT study. *Journal of Chemical Theory and Computation*, 10(6):2436–2448, 2014.

[83] Metin Aydin. DFT and Raman spectroscopy of porphyrin derivatives: Tetraphenylporphine (TPP). *Vibrational Spectroscopy*, 68:141–152, 2013.

[84] Kirill B. Agapev, Ivan I. Vrubel, Roman G. Polozkov, and Vadim K. Ivanov. The model of the fullerene C60 and its ions C60+, C60– pseudopotentials for molecular dynamics purposes. *The European Physical Journal D*, 72(11):1–6, 2018.

[85] David Mendive-Tapia, Morgane Vacher, Michael J. Bearpark, and Michael A. Robb. Coupled electron-nuclear dynamics: Charge migration and charge transfer initiated near a conical intersection. *The Journal of Chemical Physics*, 139(4):044110, 2013.

[86] Milan Delor, Theo Keane, Paul A. Scattergood, Igor V. Sazanovich, Gregory M. Greetham, Michael Towrie, Anthony J.H.M. Meijer, and Julia A. Weinstein. On the mechanism of vibrational control of light-induced charge transfer in donor-bridge-acceptor assemblies. *Nature Chemistry*, 7(9):689–695, 2015.

[87] Eric R. Bittner. An effective Hamiltonian approach for Donor-Bridge-Acceptor electronic transitions: Exploring the role of bath memory. *Condensed Matter Physics*, 19(2):1–9, 2016.