2021

Understanding the electronics of donor acceptor columnar liquid crystals

Victoria K. Sauve

Understanding the Electronics of Donor Acceptor Columnar Liquid Crystals

Victoria K Sauvé

Joseph J. Reczek
Denison University
Department of Chemistry and Biochemistry

In collaboration with
Mohammad Balooch Qarai
Francis Spano
Temple University

Abstract

Donor acceptor columnar liquid crystals (DACLCs) are at the forefront of the development of new sustainable materials for energy storage. Their unique optical properties give rise to their popularity Theoretical modeling of these systems using Density Functional Theory (DFT) can provide useful insight into the CT mechanism of different donor acceptor pairs such as naphthalenediimide (NDI) and diaminonaphthalene (DAN). However, modeling these systems requires care in the selection of functionals. The B3LYP functional is the most basic double hybrid functional. Functionals such as the CAM-B3LYP and the ωB97X-D include more specifications about long range exchange-correlation to apply to systems with more complicated noncovalent interactions such as pi-pi interactions, which are important in modeling the donor-acceptor interactions in DACLCs. Theoretical modeling of the absorption spectrum using semi-empirical wavefunction based methods can provide computationally less expensive to model the larger system. The electronic properties of the CT can be seen in the absorption spectrum of the DAN and NDI mixture. In this work, a comparison of DFT functionals is performed to determine the best model of the DACLC system (Section 1). Then, simulation of the experimental spectrum by manipulating parameters of the Hamiltonian operator is performed to provide a scaffold for the prediction of new DACLC and provide insight to the electronic interactions in a DA stack (Section 2).

Table of Contents

Introduction

Global demand for new and low-cost technologies, products, and energy continues to flourish and Earth's ability to provide diminishes so the search for sustainable development of these materials is imperative. The emerging principles of supramolecular chemistries provides the platform for the discovery of these materials at a lower cost. Liquid crystals, materials in the stable phase between crystalline solids and isotropic liquids, have become important in research and development because of their capabilities of self-organization and sensitivities to external stimuli[1]. Donor-acceptor columnar liquid crystals (DACLCs) are formed from aromatic molecules that self-assemble in face-to-face stacking (Figure 1) and are of interest for the development of photovoltaics and organic electronics.

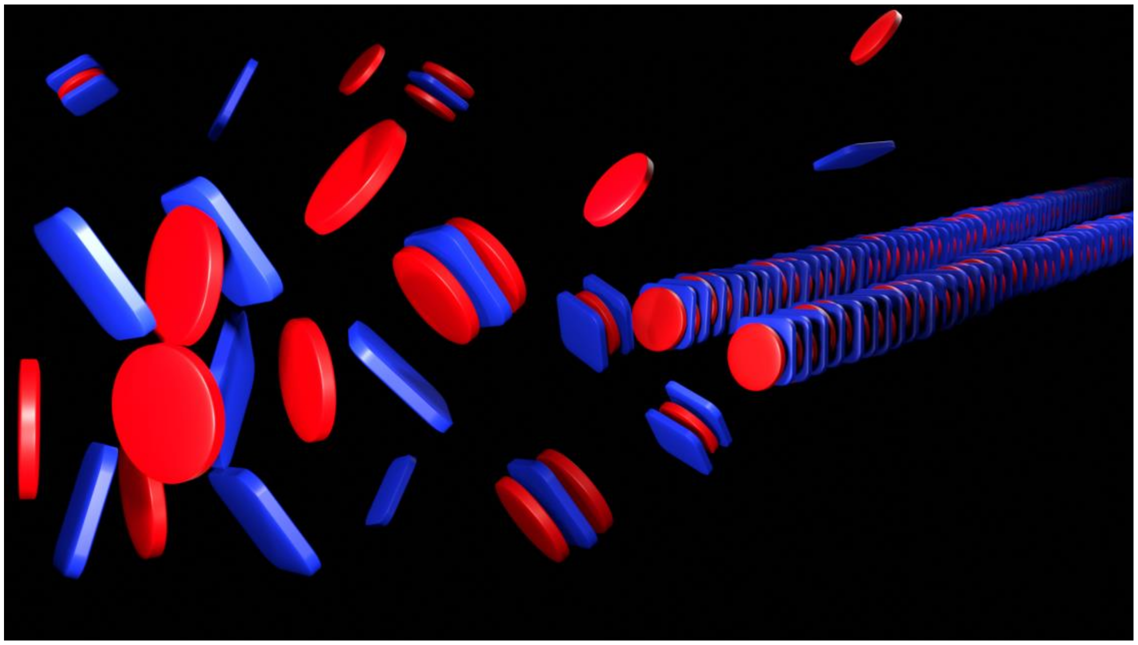


Figure 1. Model of donor-acceptor column formation. Donor (DAN) and acceptor (NDI) structures.

This face-to-face stacking results from one aromatic molecule containing an electron poor (acceptor) π-surface and the other containing an electron rich π-surface. This phenomenon results from weak noncovalent interactions known as charge transfer (CT). An example of such a pairing is with 1,5-diaminonaphthalene (DAN), the electron-rich donor and naphthalenediimide (NDI), the electron-poor acceptor (Figure 2).

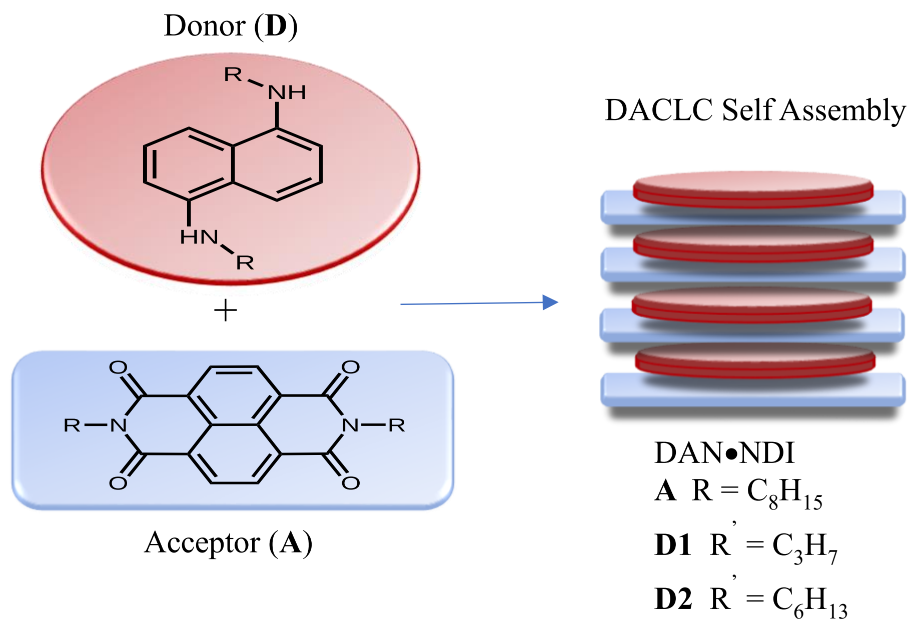


Figure 2. Schematic of self-assembly and alternation of DACLC components.

The nature of the charge transfer properties in donor-acceptor systems has been well-studied as the broad optical bands can be seen in the UV-vis spectrum[2]. The properties of a donor-acceptor system can be further understood with access to the CT band; however, it is best to be able to predict the nature of the CT properties in a given system before spending time and resources synthesizing the compounds required. Computational and theoretical methods can be used to predict these opto-electric properties and may be further developed and adapted to model our systems of interest.

Computational chemistry can be used to solve different problems that arise in chemistry. This can be done by defining a system, such as identifying the number of protons, electrons, and neutrons and identifying how they interact in space using either quantum or classical mechanics.

From there, different properties of how atoms or molecules interact in space can be predicted. In order to view from a quantum mechanical standpoint, computational chemistry must deal with different ways to solve the time-independent Schrodinger equation as shown in Equation 1.

$$\hat{H}\psi(R,r) = E\psi(R,r) \quad (1)$$

Where $\hat{H}$ is the Hamiltonian operator, ψ is the wavefunction based on the positions of the nuclei (R) and electrons (r) in the system, and E is the energy of the wavefunction. However, the equation cannot be completely solved, only approximated because it is impossible to determine the relative positions of nuclei and electrons as they are all dependent on each other. For example, if there are two electrons and a nucleus in a system, without already knowing the exact position of the nucleus, the position of the electrons cannot be known, but the position of the nucleus cannot be determined without knowing the positions of the electrons. Computational chemistry often deals with different uses of approximations to determine the wavefunction to solve the Schrodinger equation. A few important approximations include the Born-Oppenheimer approximation, the Hartree-Fock approximation, and the Linear Combination of Atomic Orbitals (LCAO) approximation. The Born-Oppenheimer approximation allows the wavefunction to separate the wavefunction of the nuclei (nuclear wavefunction) from that of the electrons (electronic wavefunction), and then to fix the position of the nuclei. The Hartree-Fock approximation allows the electronic wavefunction to be separated into a product of function that depend on only the position of one electron. Electron-electron repulsion for each electron is approximated as the average repulsion of all other electrons on one electron in the system for each. To solve this, a self-consistent field is utilized where a set of electron functions are assumed, and based on this, the average electron-electron repulsion term is determined. Then, base a new set of electronic functions based off of this value. This process can be repeated until the energy calculated using each electronic wavefunction stops decreasing. This approximation does not allow the wavefunction to describe how electron movements can impact the position of other electrons instantaneously such as electron-electron repulsion, but also electron correlation. Computational chemistry utilizes different approaches to better approximate electron-electron interactions to better model systems (All information in this paragraph)[3].

I. Density Functional Theory Study

Introduction:

Density functional theory (DFT) computes energies based on the electron density rather than electronic wavefunctions[3]. This utilizes less computing power as the total electron density function relies on 3 variables rather than the 3n variables required for the electronic wavefunctions, resulting in DFT providing a more direct route to computing molecular energies. Two theorems dictate how DFT works. First, the Hohenberg—Kohn existence theorem proves that there is a density functional that gives the exact energy. This shows that electron density follows the variational principle. For a given electron density, the energy will be greater than or equal to the exact energy of the system. This is similar to how HF molecular orbitals are solved for in that the system is solved for iteratively until the energy is minimized. However, this is done with the density instead of individual electron positions. The functionals involve the nuclear attraction terms, the classical electron-electron repulsion terms, and finally, as an improvement upon the Hartree-Fock functions, the exchange-correlation term. DFT's ability to solve for electron correlation is important because it can do this using the same computation power as HF, but improves on what is yielded[3].

Different molecular interactions benefit from the use of specific basis sets and functionals. Some systems are more susceptible to errors such as counterpoint errors than others. With this understood, it is important to identify the correct basis sets and functionals to model different systems of interest. Specifically, accurately modeling the electrostatics for charge transfer in pi stacked systems require in depth computational focus on the intermolecular interactions involved[3].

Basis sets are used to specify the atomic orbitals in a system. These can be expanded into the molecular orbitals using a combination of gaussian functions to mimic the slater type orbital. Theoretical modeling of a system at the minimum involves one basis set for each formally or partially occupied orbital in an atom; this is known a single zeta. However, single zeta is usually inadequate as it does not yield accurate distances between core electrons and valence electrons necessary for delocalization. To remedy this issue, basis sets can be doubled, or even tripled. Most basis sets involve split valence basis sets, which focus on the valence electrons by splitting them into inner and outer valence electrons. This is signified by the dash (-) in the function. For example, in the 6-31G basis function, the gaussian functions for the core electrons are detailed on the right, 6 Gaussian functions are used to model the core electrons, and the valence is then split into inner and outer valence electrons, 3 Gaussian functions for the inner and 1 for the outer valence electrons[3].

Each functional differs in how it deals with the exchange-correlation term. The exchange correlation term is separated into an exchange term and a correlation term, and with that, their own energies. In the simplest treatment of the two terms is local density approximations (LDA), which assumes that the exchange energy can be solved using a constant density value. This makes the assumption of complete uniformity. Local spin density approximation (LSDA) assumes similarly but includes the presence of alpha and beta densities and that they are not equal. Improvements upon this involve including the variational aspect of density by adding the derivative of the density function into the functional. Such approximations are known are generalized gradient approximations (GGA). GGA were improved with the meta-GGA functional, which add in dependence of the Laplacian, which led to hyper-GGA with added dependence on the exact HF exchange. Double hybrid functionals further these functionals by including unoccupied Kohn Sham orbitals[3]. The most basic hybrid functional is the B3LYP, with the "B" denoting the exchange term by Becke, and the "LYP", denoting the use of the correlation functional by Lee, Yang, and Parr. Becke's exchange term involves the use of an LSDA with the density derivative. The LYP term uses the Laplacian of the density for the correlation term. One problem arises in modeling the exchange-correlation hole, which at the limit, only involves exchange and no correlation[3]. This is improved upon by including a long-range correction using nonlocal HF terms denoted as long range and short range; however, HF is only applied to the long range. The use of the nonlocal HF was first used in the M06 suite of functionals, which were then used for systems with noncovalent interactions, such as pi-pi stacking as seen in the literature[4]. The omegaB97 functionals then incorporates both long- and short-range HF corrections to its system[3]. Finally, dispersion corrected DFT, signified with "D" has been incorporated into many models to more accurately model London dispersion interactions[5]. Previous research has found that the ωB97X-D has been the best functional to model donor-acceptor systems[6,7].

Results and Discussion

All calculations were performed on Spartan software. Three functionals were tested to determine the functional that predicts the HOMO to LUMO energy levels. They were performed first by calculating the equilibrium geometry with the molecules at the ground state in the gas phase using the basis set 6-311G**. This was done for the NDI (Table 1, Figure 3) and DAN (Table 2, Figure 4) monomers, dimer (Table 3, Figure 5), dimer with two unpaired electrons (Table 4, Figure 6), trimer (Table 5, Figure 7), and trimer with two unpaired electrons (Table 6, Figure 8). The values were then compared to the energy gap in the excitation spectrum taken in solution of NDI (Figure 3), DAN (Figure 4), and the mixture of the two (Figure 5).

The energy gaps from DFT calculations were found by subtracting the HOMO energy level from the LUMO energy level. The energy gap was calculated from the experimental absorption spectra with Equation 2. Although there are many more than two to three donor-acceptor pairs in a stack, dimer and trimer simulations were done to simulate a stack as DFT calculations of a larger system would cost too much computationally.

$$Band\ Gap = \frac{1}{Wavelength(nm) * 10^{-7}cm} \div 8065eV \qquad (2)$$

NDI has a band gap of 3.26 eV as shown in the experimental spectrum (Figure 4). The functional yielding the closest energy gap between the HOMO and LUMO energy levels is the B3LYP functional with a band gap of 3.2 eV. Similarly, the B3LYP functional results in the most accurate band gap for the DAN molecule with the DFT calculation providing a band gap of 4.2 eV, while the experimental shows a band gap of 3.56 eV. Among dimer calculations, the CAM-B3LYP produced a band gap of 2.5 eV, closest to the excitation energy in the experimental around 2.57 eV. For the dimer with 2 unpaired electrons, the B3LYP produced a band gap of 1.7 eV, which is close to the CT peak at 1.86 eV in the experimental spectrum. The trimer calculation with the B3LYP functional also produced a band gap of 1.7 eV. Finally, the ωB97X-D functional performing the calculation for the trimer with two unpaired electrons resulted in the closest band gap calculation for the excitation around 482 nm, or 2.57 eV as this calculation showed a band gap of 2.1 eV. However, the calculation with the CAM-B3LYP functional produced a value closer to the CT at 1.4 eV. Additionally, the trimer calculation produced an interesting result in the orbital diagram as it shows electron delocalization across the stack.

Because previous research has shown the usefulness of time-dependent DFT calculations in modeling the electronics of pi-pi stacked systems[10], the UV-Vis spectrum simulator in Spartan was utilized to determine which functional could best predict the experimental absorption spectrum. This was done using different functionals to calculate the absorption spectrum of the DAN and NDI dimer. The ωB97X-D functional provides the most closely matching spectrum, with the CT band around 667 nm (Figure 12).

Table 1. HOMO and LUMO calculations of NDI found using three different functionals using the 6-311G** basis set. Molecule was simulated in the gas phase at the ground state.

Functional	HOMO (eV)	LUMO (eV)	Energy Gap (eV)
ωB97X-D	-9	-2.2	-6.8
CAM-B3LYP	-8.4	-2.8	-5.6
B3LYP	-7.2	-3.8	-3.4

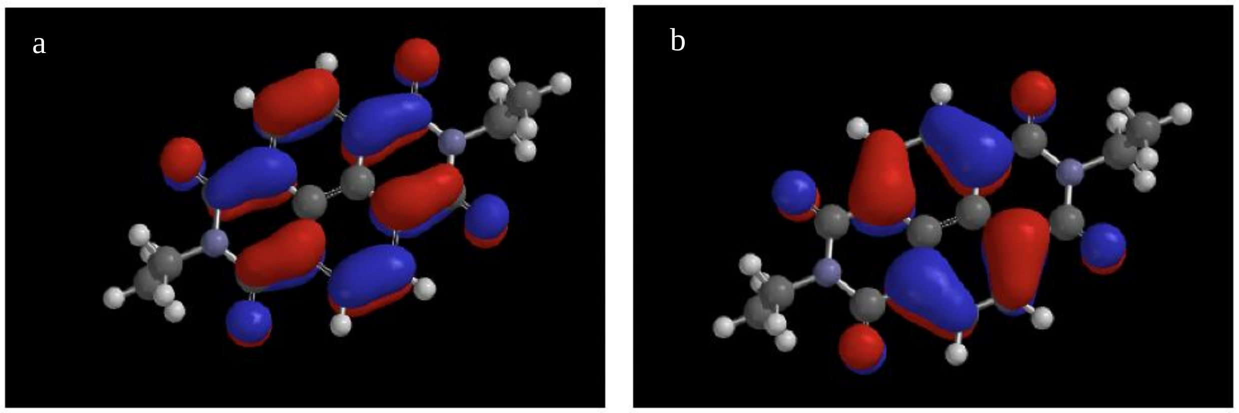


Figure 3. LUMO (a) and HOMO (b) orbital diagrams for NDI.

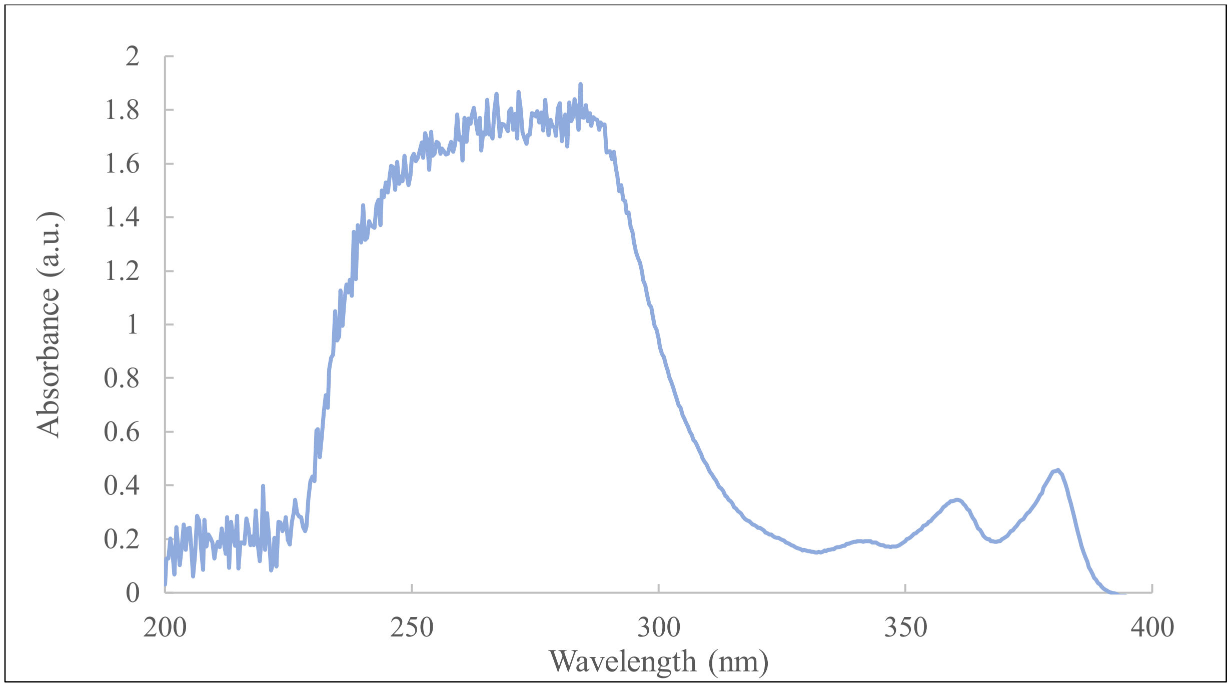


Figure 4. Absorption spectrum of NDI solution. Taken using the Ocean Optics spectrophotometer. Excitation (A*) at 380 nm (3.26 eV)

Table 2. HOMO and LUMO calculations of DAN found using three different functionals using the 6-311G** basis set. Molecule was simulated in the gas phase at the ground state.

Functional	HOMO (eV)	LUMO	Energy Gap (eV)
ωB97X-D	-7	0.8	-7.8
CAM-B3LYP	-6.5	0.1	-6.6
B3LYP	-5.2	-1	-4.2

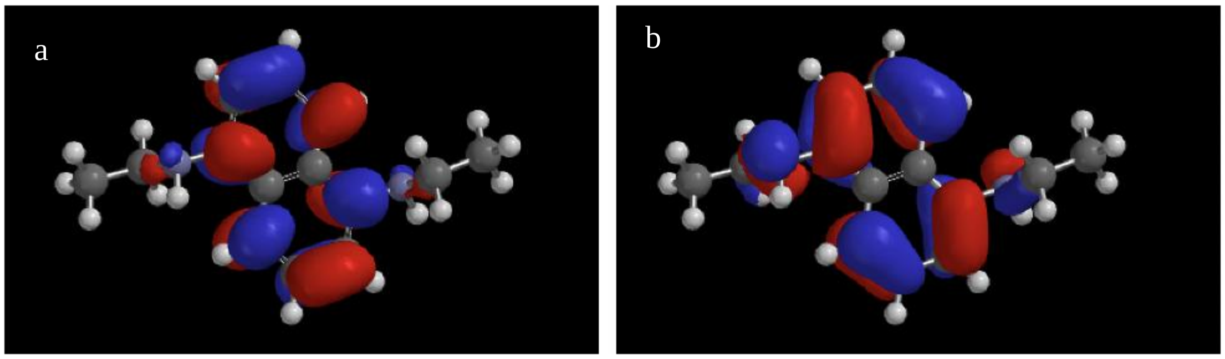


Figure 5. LUMO (a) and HOMO (b) orbital diagrams for DAN.

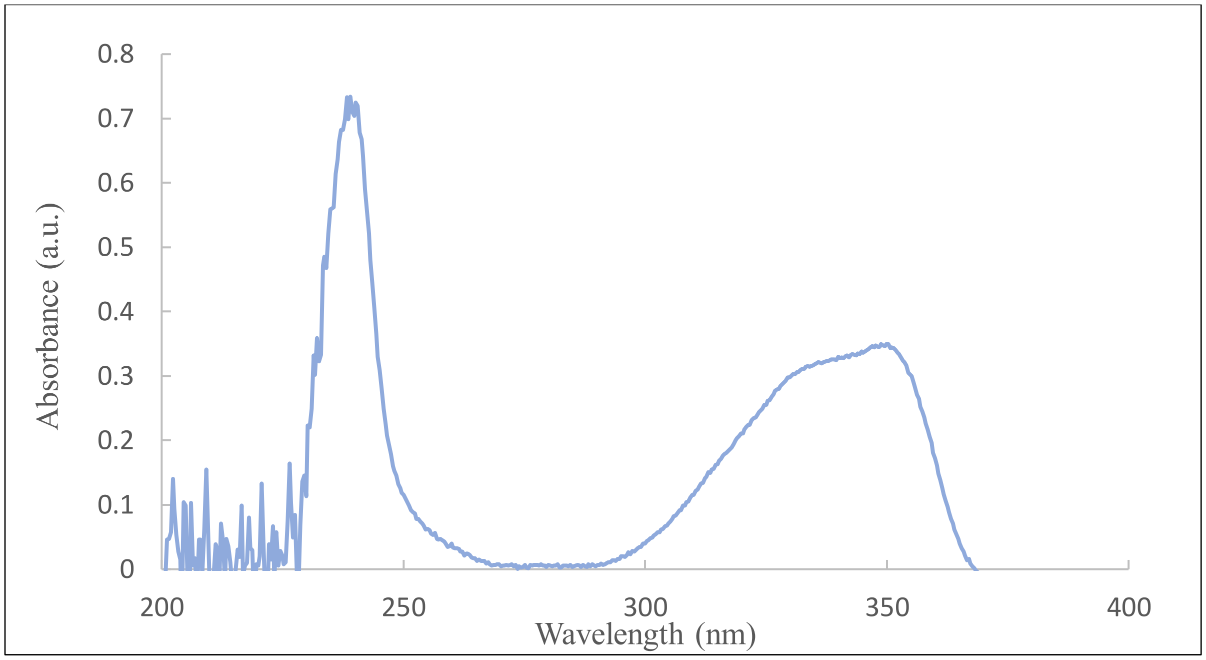


Figure 6. Absorption spectrum of DAN solution. Taken using the Ocean Optics spectrophotometer. Excitation (D*) at 350 nm (3.54 eV).
B3LYP yields best results for monomers

Table 3. HOMO and LUMO calculations of dimer found using three different functionals using the 6-311G** basis set. Molecule was simulated in the gas phase at the ground state.

Functional	HOMO (eV)	LUMO (eV)	Energy Gap (eV)
ωB97X-D	-6.7	-1.8	-4.9
CAM-B3LYP	-6.2	-2.5	-2.5
B3LYP	-5.3	-3.6	-3.6

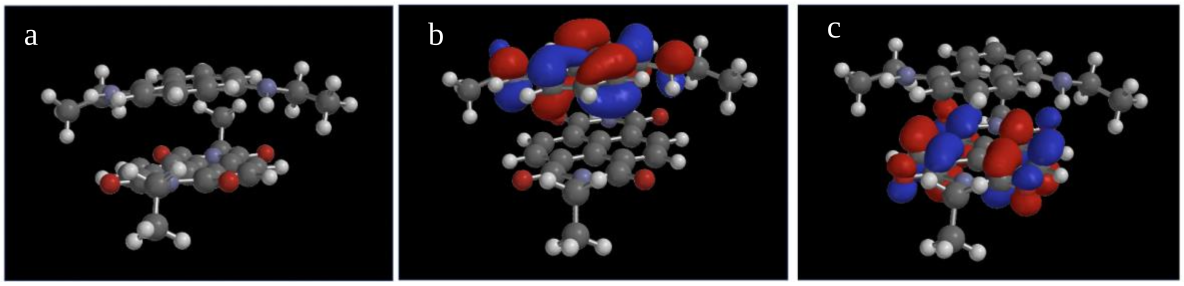


Figure 7. Equilibrium geometry (a), LUMO (b) and HOMO (c) orbital diagrams for NDI+DAN dimer.

Table 4. HOMO and LUMO calculations of dimer found using three different functionals using the 6-311G** basis set. Molecule was simulated in the gas phase at the ground state with 2 unpaired electrons.

Functional	HOMO (eV)	LUMO (eV)	Energy Gap (eV)
ωB97X-D	-4.7	-0.2	-4.5
CAM-B3LYP	-4.2	-0.8	-3.4
B3LYP	-3.4	-1.7	-1.7

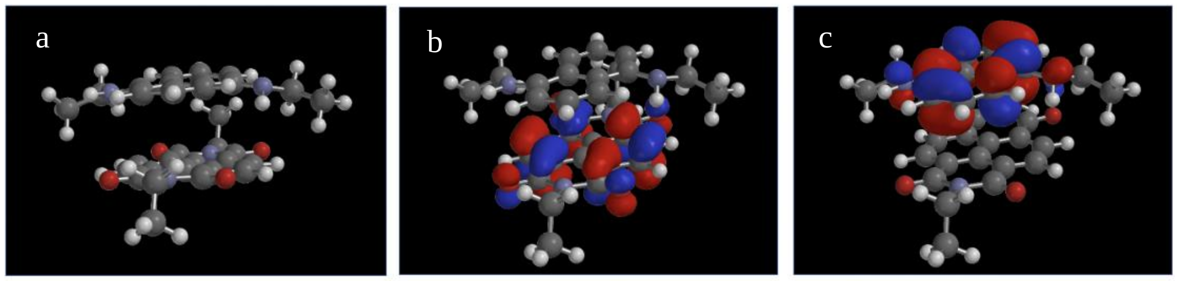


Figure 8. Equilibrium geometry (a), LUMO (b) and HOMO (c) orbital diagrams for NDI+DAN dimer with two unpaired electrons.

Table 5. HOMO and LUMO calculations of trimer found using three different functionals using the 6-31G* basis set. Molecule was simulated in the gas phase at the ground state.

Functional	HOMO (eV)	LUMO (eV)	Energy Gap (eV)
ωB97X-D	-7.0	-1.7	-5.3
CAM-B3LYP	-6.5	-2.3	-4.2
B3LYP	-5.3	-3.6	-1.7

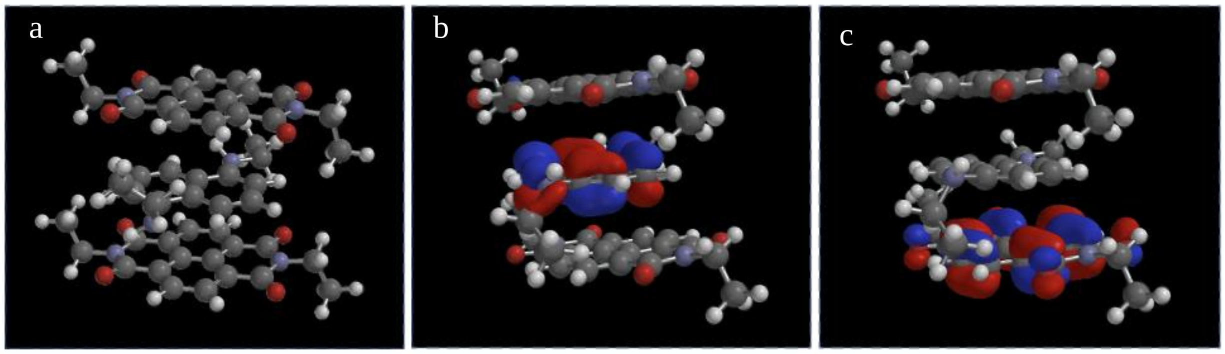


Figure 9. Equilibrium geometry (a), LUMO (b) and HOMO (c) orbital diagrams for NDI+DAN+NDI trimer.

Table 6. HOMO and LUMO calculations of trimer found using three different functionals using the 6-31G* basis set. Molecule was simulated in the gas phase at the ground state with 2 unpaired electrons.

Functional	HOMO (eV)	LUMO (eV)	Energy Gap (eV)
ωB97X-D	-4.7	-2.6	-2.1
CAM-B3LYP	-4.3	-2.9	-1.4
B3LYP	-3.9	-3.4	-0.5

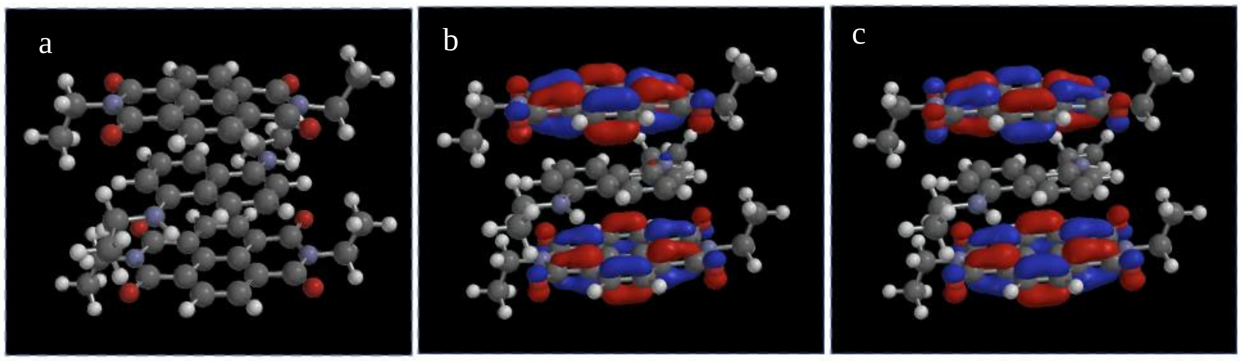


Figure 10. Equilibrium geometry (a), LUMO (b) and HOMO (c) orbital diagrams for NDI+DAN+NDI trimer with 2 unpaired electrons.

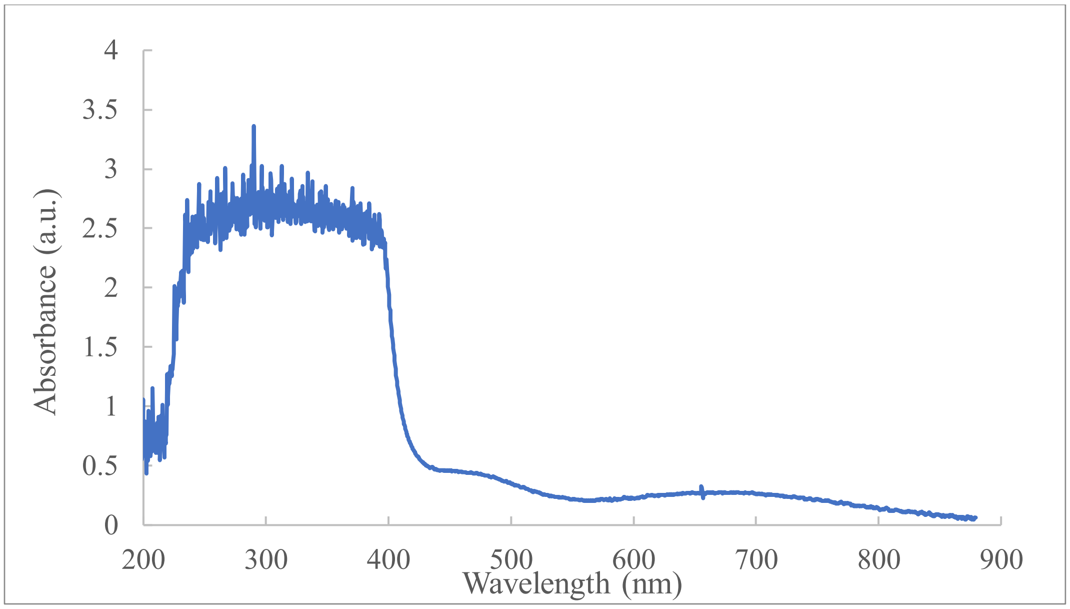
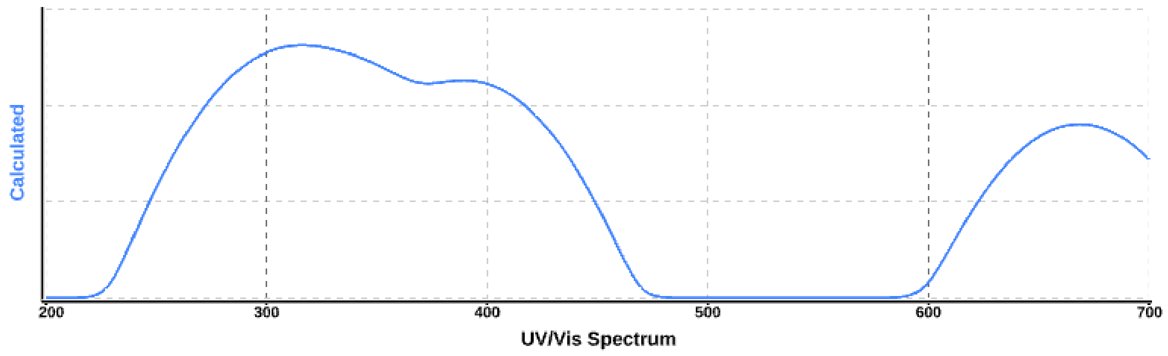
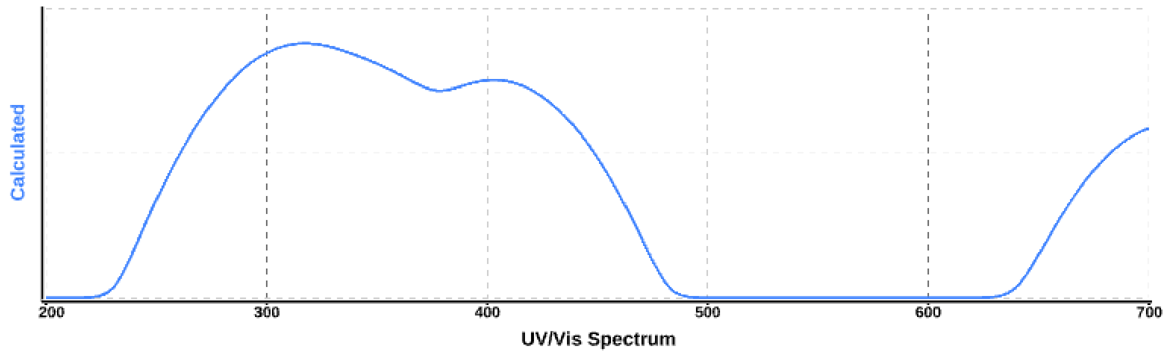


Figure 11. Absorption spectrum of DAN and NDI solution mixture. Excitation at 482 nm (2.57 eV) and CT around 667 nm (1.86 eV)





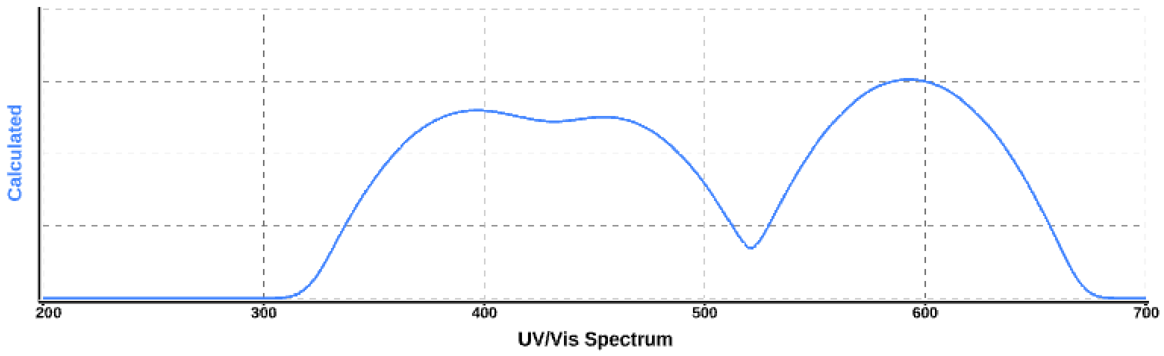


Figure 12. UV-Vis spectra of dimer calculated using the ωB97X-D functional (b), the CAM-B3LYP functional (c), and the B3LYP functional (d). The ωB97X-D functional provides a peak closest to 667 nm out of the three functionals.

Conclusion

The DFT study shows that the B3LYP functional produces HOMO-LUMO band gaps that match most closely with the band gap found experimentally in the absorption spectrum. The functionals for the dimer and trimer calculations produced results that are difficult to interpret with significant meaning. The dimer calculation with the CAM-B3LYP functional produced HOMO and LUMO energy levels with a band gap most closely matching the excitation energy in the experimental spectrum. Other functionals for the trimer calculations produced band gaps closely matching the CT energy level in the experimental spectrum. It is hard to know if the calculations are providing results that are meant to be of the excitation, the CT, or a mixture of the two. It is also worth noting that the literature suggests that the ωB97X-D produces the best results for donor acceptor systems[6,7], which was reflected in the UV-Vis study. Without more in depth knowledge about DFT methods, it is difficult to determine the best functional for the extended system. Additionally, the calculation designates specific HOMO and LUMO levels, but in the system, orbitals are much more complicated and undergo mixing.

II. Absorption Spectrum Simulations

Introduction

Although density functional theory provides information about the HOMO and LUMO gaps, stacking geometries, and excitation spectra of smaller systems, such as dimers and trimers of the DA system, more in-depth theory is necessary to understand the excitation and charge transfer properties of large-scale systems. Additionally, it would be useful to understand the polarization of the CT and exciton as shown in the polarized absorption spectrum found experimentally (Figure 13). Previous work has developed a model to simulate the absorption spectra of hybrid organic-inorganic perovskites, which feature a donor-acceptor energy coupling, which can be compared to the CT coupling in the DACLCs[8]. The basis set is created utilizing a 3rd nearest neighbor approach, meaning, if one chromophore in the chain is excited, the first, second, and third are approximated to be impacted by it. Then, the Hamiltonian is formed using this basis set. The researchers then changed parameters to fit an experimental excitation spectrum. Once they found a suitable fit, they were able to predict the spectra of different donors and acceptors[8]. This model can be shifted by moving to a model in which the nearest neighbor is the only one impacted by a charge transfer aspect (D+, A-).

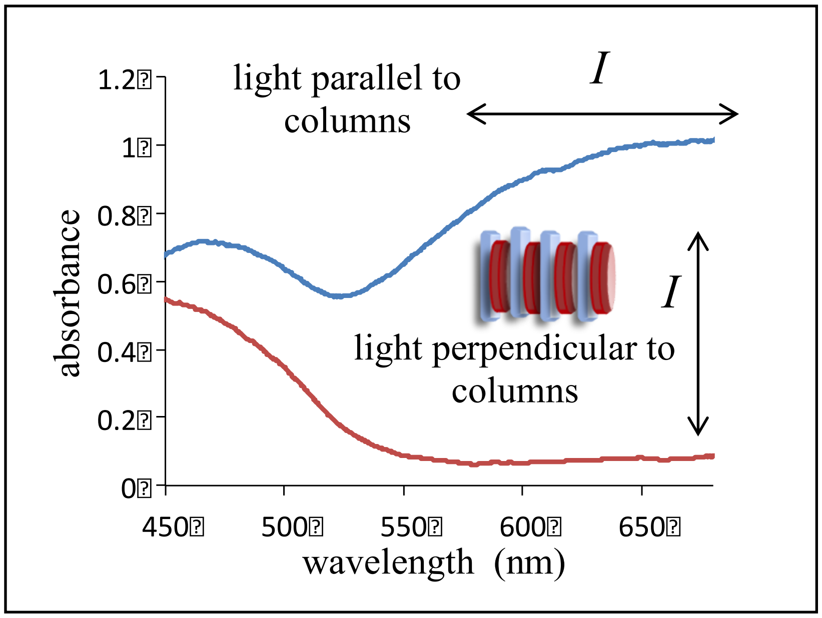


Figure 13. UV/Vis absorbance demonstrating CT dichroism of DACLC under LPL

Methods

The simple model is based on a Frenkel Exciton/CT Holstein model based on a linear array of donors and acceptors. An exciton is a delocalized excitation, where the excitation of one molecule induces the excitation of another fragment in the linear array, which in this simulation, the excitation can be delocalized as far as the third nearest neighbor. In a CT, an electron is actually transferred from the donor to the acceptor. The overall Hamiltonian operator is shown in Equation 3, with the $\hat{H}'$shown in Equation 3 and $\hat{H}^{(0)}$ is simply the monomer energy (donor excitation).

$$\hat{H} = \hat{H}^{(0)} + \hat{H}' \ (3)$$

$$\hat{H}' = -E_{D^*}|g\rangle\langle g| + t\sum_{j=2,4,6,\ldots}^{N_F}\{|g\rangle\langle j+1,j| + |g\rangle\langle j-1,j| + h.c.\} \ (4)$$

In this equation, -E_{D^*} is the energy of the donor excitation, found experimentally in the UV-Vis, g is the energy of the ground state, t is the coupling constant between the ground state of the donor and the CT state, j is the donor fragment that is undergoing the CT with acceptors before and after it (j-1,j+1). The Hamiltonian operator can be put into matrix form and then diagonalized by finding the eigenvalues of the matrix, these steps can be found in Equation 5. The oscillator strength is another value of importance as it was found to be necessary for the calculation of the correctly polarized absorption spectrum as mentioned later. The equations for this calculation can be found in Appendix 1. The results of this equation are reflected by μ_A, μ_D, and μ_CT. Another part of this calculation are the delta values (Appendix 1). Once this computation is performed, the system of equations that relates the positions of each particle can be more easily solved for.

$$\boldsymbol{H'} = \begin{pmatrix} -E_g & t & t & t & t \\ t & -\Delta & 0 & 0 & 0 \\ t & 0 & -\Delta & 0 & 0 \\ t & 0 & 0 & -\Delta & 0 \\ t & 0 & 0 & 0 & -\Delta \end{pmatrix} \tag{5}$$

$$det(\boldsymbol{H'} - a\boldsymbol{I}) = 0.0$$

$$(\Delta + a)^3\left(-\left(E_g + a\right)(\Delta + a) + 2t^2\right) = 0.0$$

$$a_{1,2} = \frac{1}{2}\left[-(E_g + \Delta) \pm \sqrt{(E_g + \Delta)^2 - 4(E_g\Delta - 2t^2)}\right]$$

$$a_{3,4,5} = -\Delta$$

There are several parameters in the code (Appendix 2) that are used to form the Hamiltonian operators for the exciton, CT, and ground state Hamiltonians. The connection of these parameters to the different CT states can be seen Figure 8. The main parameter that goes into the Hamiltonian operator in the J_{DA} term (JDA in the code). This represents the coupling between the donor and the acceptor, which means the coulombic interaction between the charges and allows the excitation to be a linear combination of all states. The operator for the CT state includes the energy when the donor is negatively charged and the acceptor is positively charged (wdnap in code, ΔEA in Figure), the energy when the donor is positively charged and the acceptor is negatively charged (wdpan in code, ΔIP in Figure), and U-V (VMU in code). The wdnap parameter is found using DFT calculations (B3LYP, 6-311G**) by finding the differences in the HOMOs of the donor and acceptor. Additionally, the wdpan parameter is found by finding the difference in the LUMOs of the donor and acceptor. U-V is a positive quantity that shows two factors: first, U represents the binding energy of the exciton when the electron and hole are located on the same site second, V is the binding energy (energy of attraction) between the electron and hole that are on different sites; electron is on the nearest neighbor (acceptor). The quantity of U-V is always positive because the binding energy of the electron and hole when they are on the same side is always larger than when on different sides. Finally, the ground state Hamiltonian incorporates the coupling of the CT state to the ground state.

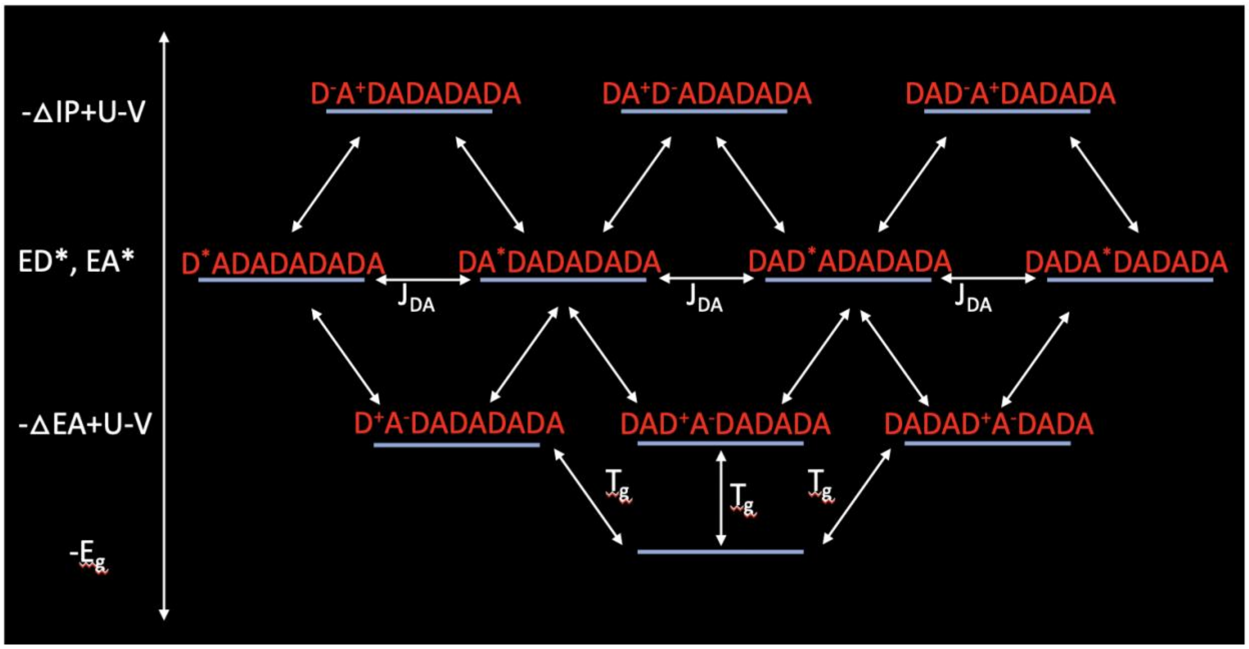


Figure 8. Schematic of simulated electronic states for DACLC in absorption spectrum. E_g corresponds to the monomer energy of the donor, ED* and EA* correspond to the excitation of the donor and acceptor, respectively. Excitation energies were found using Ocean Optics spectrophotometer. ΔEA corresponds to the difference in the HOMOs and ΔIP corresponds to the difference between the LUMOS of the donor and acceptor found using DFT with the ωB97X-D functional and the 6-311+G** basis set. J_{DA} and T_g were the coupling constants that

were found by fitting. Finally, U-V was a fitted parameter where U is the energy to separate e- and hole pair on the donor and V is the energy of attraction between electron on donor and hole on the acceptor. Fitted parameters in the Hamiltonian allowed finding the line of best fit for the experimental spectrum.

Other fitting parameters of note used in the code include the number of fragments (N), The linewidth of the lower energy peak (gam LE), the linewidth of the higher energy peak (gam HE), and the energy at which the splitting between the high energy and low energy peaks take place (wcut). The parameters gam LE and gam HE determine the intensity of the peaks in the spectrum. All values in the code are converted to cm^{-} in terms of the donor excitation energy. This is done by dividing a value in cm^{-} by 1400, setting the donor energy to zero (Ed*=0 in the code).

The code is run using the Terminal in Mac OSX using the gfortran, lapack, and blas packages. After the directory is set, the code is compiled (Line 1) and then run (Line 2).

```
gfortran DAGcopolymerVKS.f95 -o DAGcopolymerVKS -llapack -lblas (Line 1)

./DAGcopolymerVKS (Line 2)
```

Once the simulation parameters from the experimental spectra and DFT calculations were put into place, the simulation was run, and the fit was compared to the that found experimentally. Then, the other parameters were changed based on how they control the spectrum to get the best fit.

Results and Discussion

Using the Fortran code written by Mohammad Balooch Qarai, different parameters used in solving the Hamiltonian were changed to understand how they impacted the system. As the ground state energy is increased (Tg), the CT band, which starts red shifted from the exciton band, becomes blue shifted (Figure 9). As the ground state energy is increased, separation between the orbitals is still maintained so the upper orbital resulting from the CT and ground mixing eventually passes the excited state of the donor (Figure 10). When increasing the U-V parameter, the gap between the exciton and the CT bands became smaller (Figure 11). As J_{DA} is increased, the exciton peak becomes blue shifted (Figure 12).

These parameters were some of the most important in forming numerically calculated UV-Vis spectra that matched the experimental spectrum (Figure 13, 14, 15). Additionally, these spectra when polarized show the expected polarization, with the lower energy being the x-polarized CT band, and the higher energy being the y-polarized exciton (Figure 8).

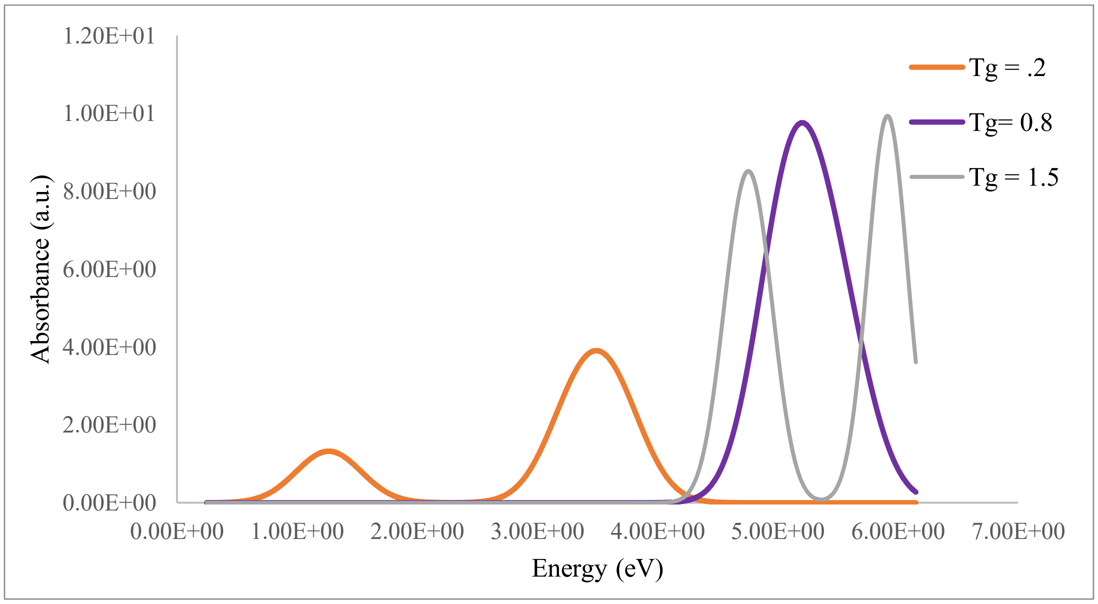
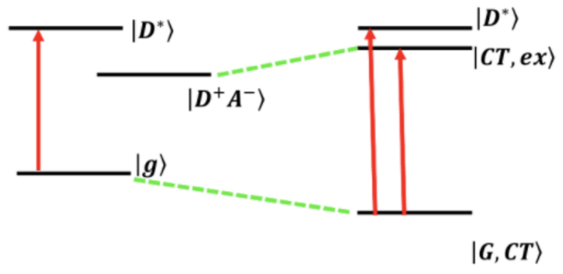


Figure 9. Absorbance as a function of energy at different ground state energies (Tg in eV).



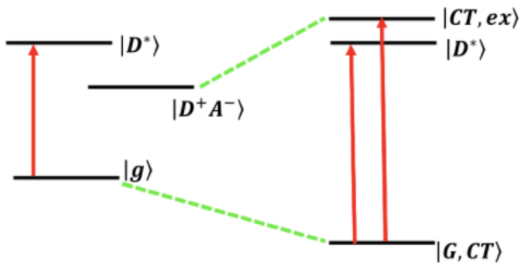


Figure 10. Representation of how changing the ground state energy impacts the CT band. (Created by Mohammad Balooch Qarai)

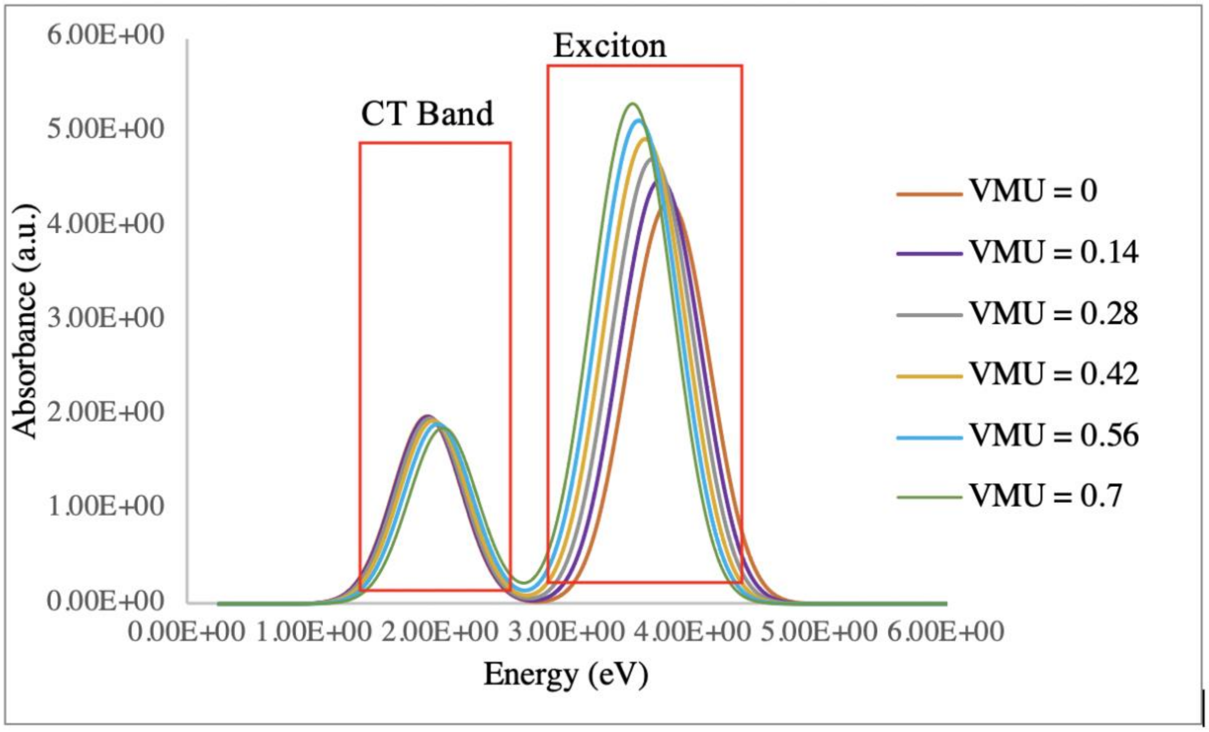


Figure 11. Energy as a function of absorbance at different values of U-V (eV)

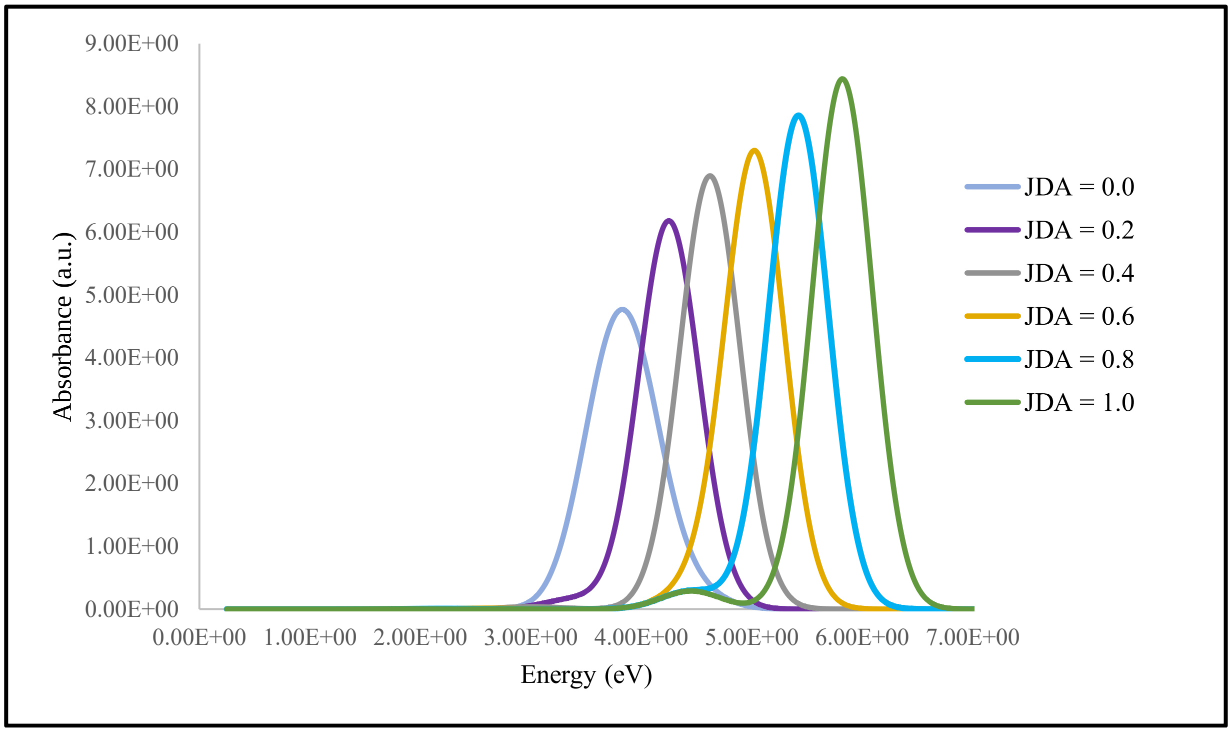


Figure 12. Energy as a function of absorbance at different values of J_{DA} (eV) with the CT at zero to better observe how it impacts the exciton peak.

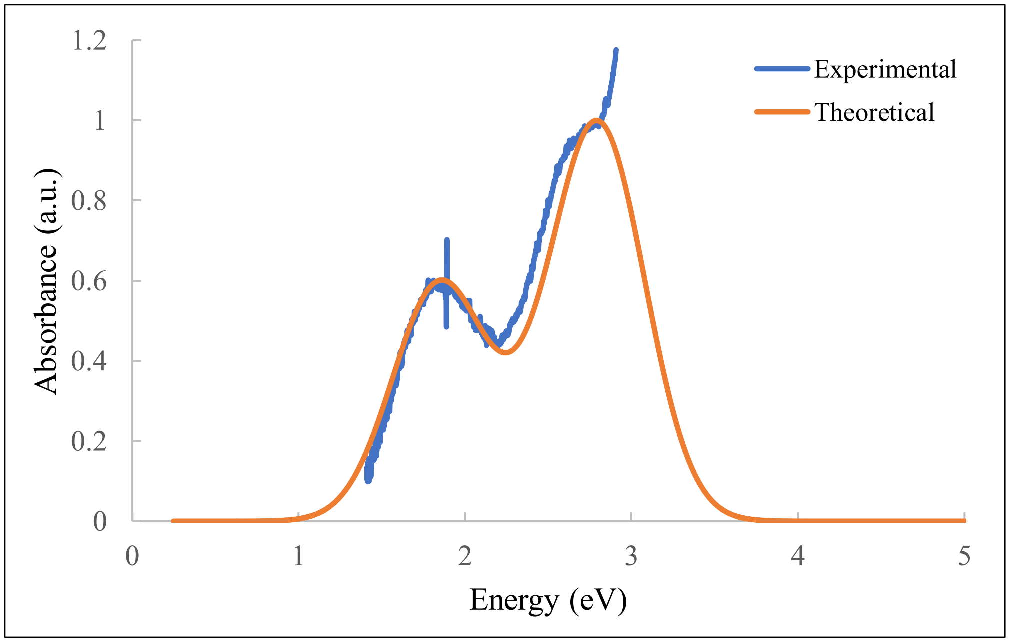


Figure 13. Energy as a function of absorbance from theoretical calculation and experimental result. Parameters used in the theoretical calculation are shown in Table 1.

Table 7. Parameters used in the simulated UV-Vis spectrum from Figure 5.

Parameters	Values
N	10
Vmax	0
DeltaIP (eV)	2
DeltaEA (eV)	3
U-V (eV)	1.39
Wvib (eV)	0.1736
ED* (eV)	0
EA*=ED* (eV)-x	x= -0.28
JDA (eV)	0.09
μ_A	1
μ_D	1
μ_CT	4.5
Tg (eV)	0.286
gam LE (eV)	0.39
gam HE (eV)	0.39
wcut (eV)	2.79
Monomer E	16000

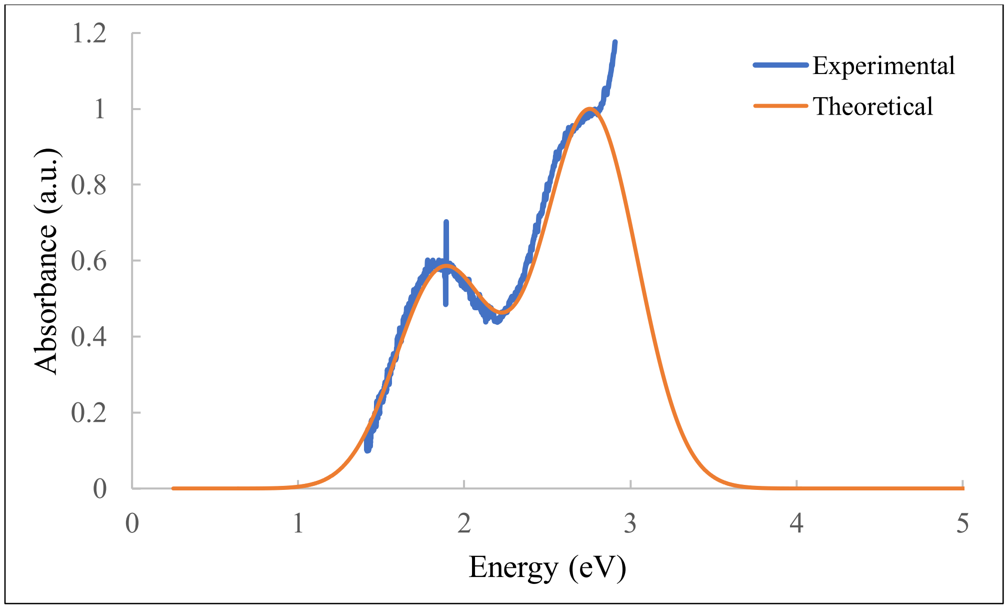


Figure 14. Energy as a function of absorbance from theoretical calculation and experimental result. Parameters used in the theoretical calculation are shown in Table 2.

Table 8. Parameters used in the simulated UV-Vis spectrum from Figure 6.

Parameters	**Values**
N	10
Vmax	0
DeltaIP (eV)	2
DeltaEA (eV)	3
VMU (eV)	1.48
Wvib (eV)	0.1736
ED* (eV)	0
EA*=ED* (eV)	-0.28
JDA (eV)	0.09
µ_A	1
µ_D	1
µ_CT	4.5
Tg (eV)	0.286
gam LE (eV)	0.39
gam HE (eV)	0.399
wcut (eV)	2.79
Monomer E	16000

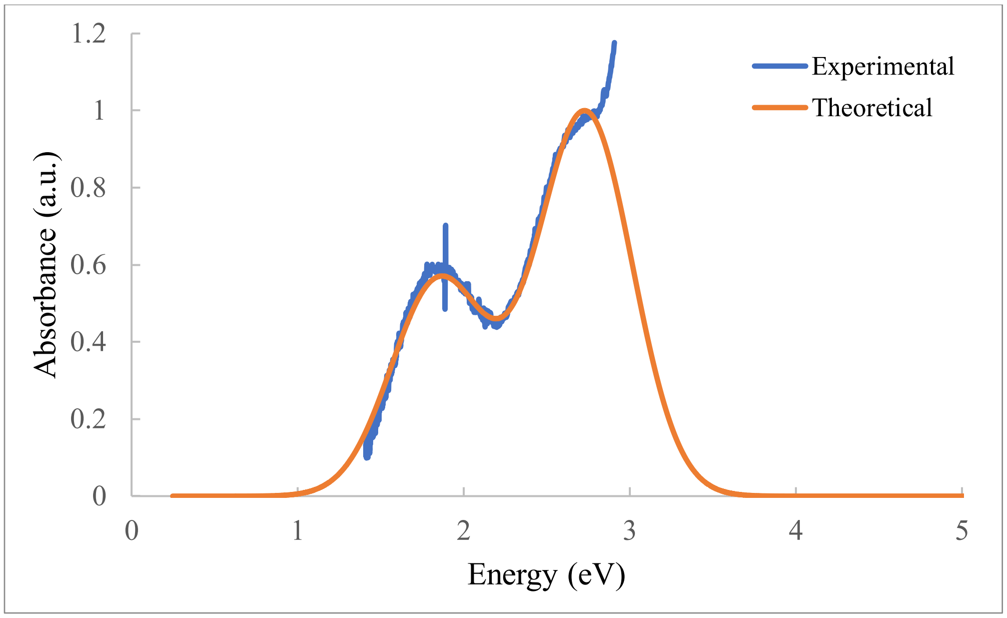


Figure 15. Energy as a function of absorbance from theoretical calculation and experimental result. Parameters used in the theoretical calculation are shown in Table 3.

Table 9. Parameters used in the simulated UV-Vis spectrum from Figure 7.

Parameters	Values
N	10
Vmax	0
DeltaIP (eV)	2
DeltaEA (eV)	3
VMU (eV)	1.48
Wvib (eV)	0.1736
ED* (eV)	0
EA*=ED* (eV)	-0.28
JDA (eV)	0.09
μ_A	1
μ_D	1
μ_CT	4.5
Tg (eV)	0.279
gam LE (eV)	0.39
gam HE (eV)	0.399
wcut (eV)	2.79
Monomer E	16000

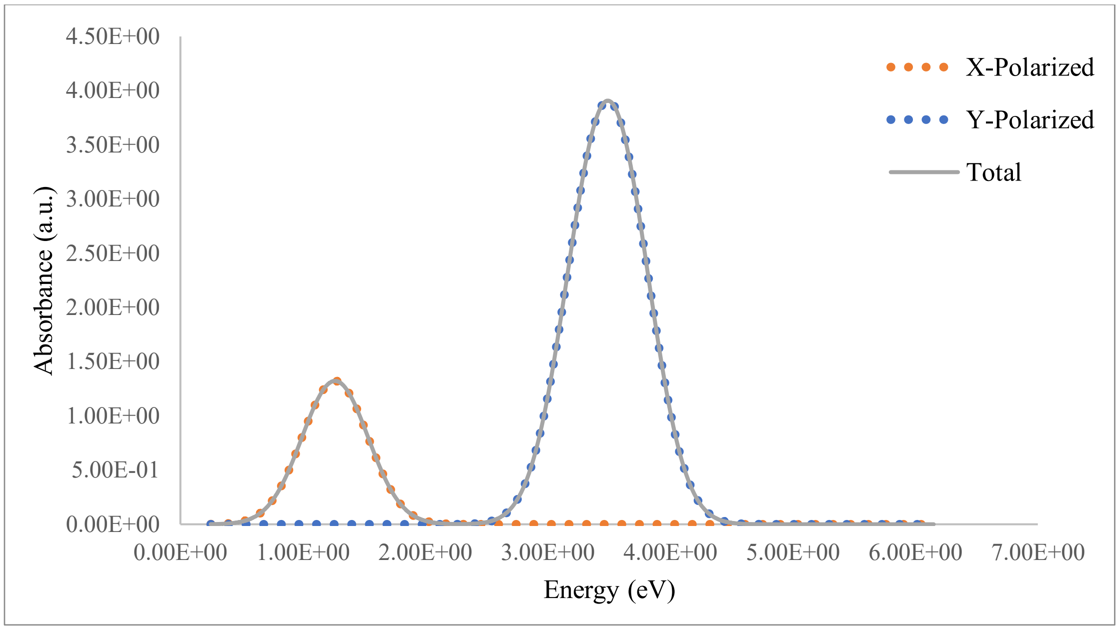


Figure 16. Energy as a function of absorbance from theoretical calculation. This shows that the lower energy is the x-polarized band and that the higher energy is the y-polarized band.

Conclusion

The results of this study produced incredibly well fittings of the experimental spectrum by manipulating the Hamiltonian operators of the different states. Results showed that the polarization of the CT was correctly predicted as shown in experiments. Additionally, the simulation produced results that informed our understanding of the orbital mixing in the donor acceptor pi delocalized system. Before adding the oscillator strength, allowing a coupling between the CT and ground state and adding the T_{ground} coupling constant, the simulation failed to produce results that allowed for the polarization of the CT band parallel to the stack, which has been shown to be true experimentally. This shows that there is orbital mixing between the ground state and the CT that was not previously known. The simulation also showed that the exciton peak is perpendicular to the donor-acceptor stack which has not yet been strongly supported by experiment. Further studies with polarized spectra must be done to confirm this. The simulation can be further explored by fitting the thin film spectra, rather than solution spectra and also by modeling the absorption spectrum of other donor-acceptor fragments. Finally, this simulation can hopefully be used to shed further light on the electronics of the pi delocalized system in DACLCs and perhaps help to predict the opto-electronic properties of novel donor-acceptor pairings that have not yet been synthesized.

III. Future work: Thin film study

Preliminary results for modeling of the thin film of the NDI + DAN mixture are shown in Figure 17. Additionally, experimental solution spectra and DFT calculations were found for new fragments including Mellitic Triimide (Figure 18)[12] and Anthracene (Figure 20). Again, initial results are shown in Figures 19 and 21. After these initial studies were conducted, it was realized that the wrong functional was used for monomer calculations. Further study will be done after implementing the corrected DFT values using the B3LYP functional (Table).

N	10
DeltaIP (eV)	2
DeltaEA (eV)	3
U-V (eV)	1.26
ED* (eV)	2.3806
EA*=ED* (eV)	-0.29
JDA (eV)	0.1
μ_A	1
μ_D	1
μ_CT	4
Tg (eV)	0.2899
gam LE (eV)	0.3992
gam HE (eV)	0.4426
wcut (eV)	2.66

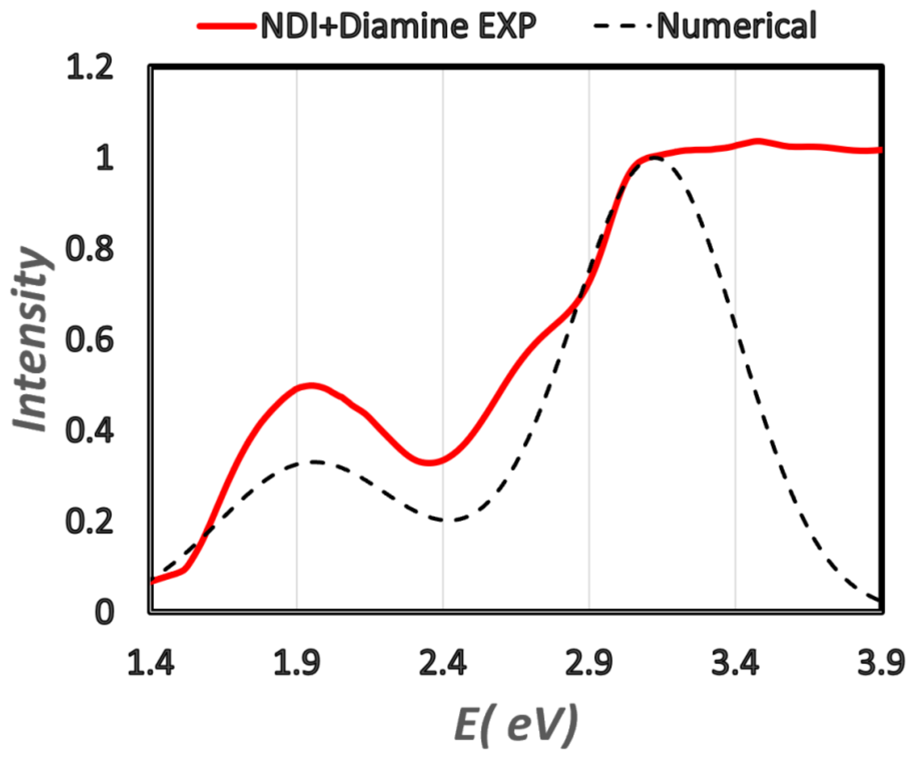
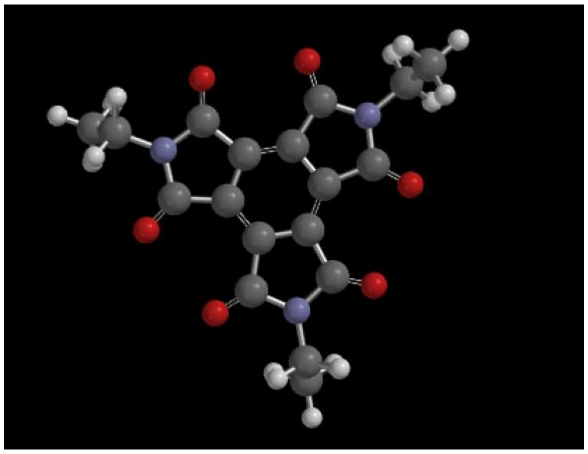
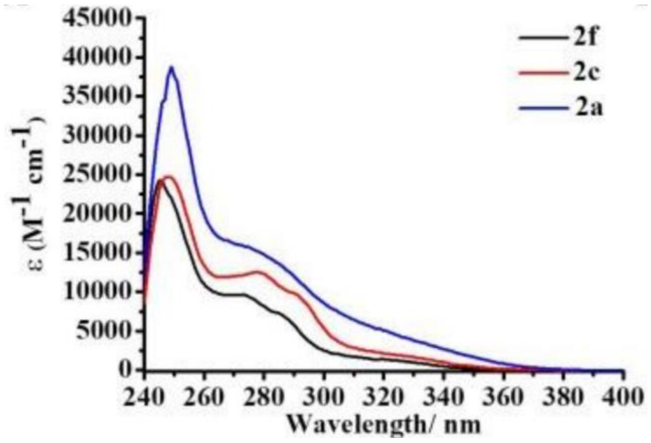


Figure 17. Numerical results (Left) of simulation for the NDI and DAN thin film absorption spectrum. Comparison of theoretical results (Numerical) with the experimental absorption spectrum.





Functional	Basis Set	HOMO (eV)	LUMO (eV)
ωB97-X	6-311+G**	-10.1	-1.8

Figure 18. MTI structure (Left) and MTI absorption spectrum (2c, 4.27 eV) (Right). DFT calculations of HOMO and LUMO orbitals (Bottom).

N	10
Vmax	0
DeltaIP (eV)	2.6
DeltaEA (eV)	3.1
U-V (eV)	1.26
ED* (eV)	1.8971
EA*=ED* (eV)+X	0.73
JDA (eV)	0.1
μ_A	1
μ_D	1
μ_CT	4
Tg (eV)	0.3177
gam LE (eV)	0.3992
gam HE (eV)	0.4426
wcut (eV)	2.66

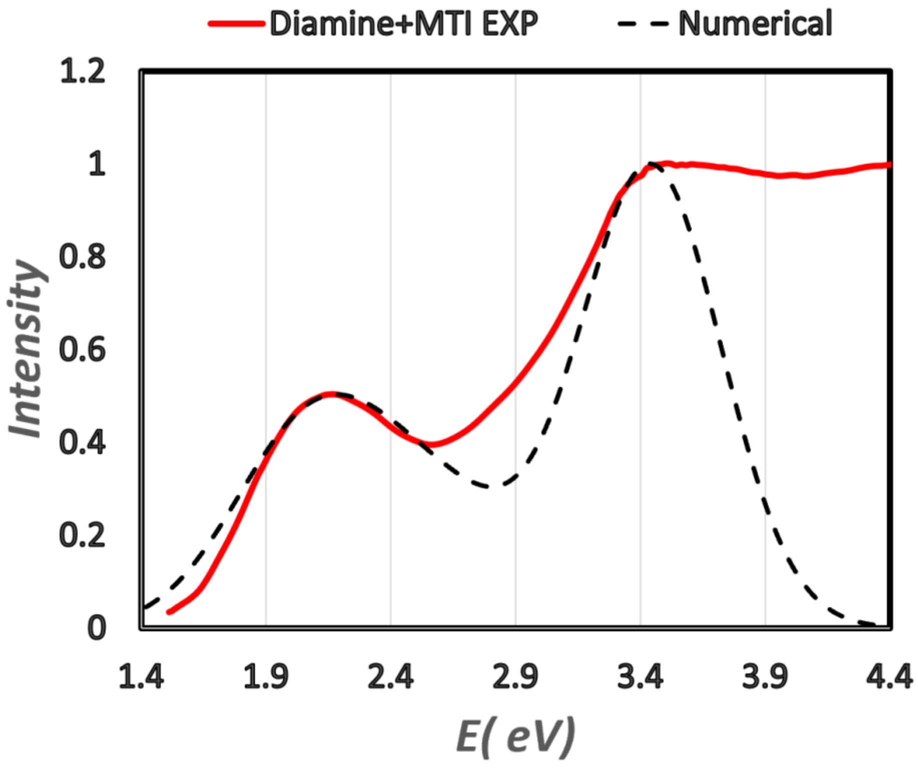
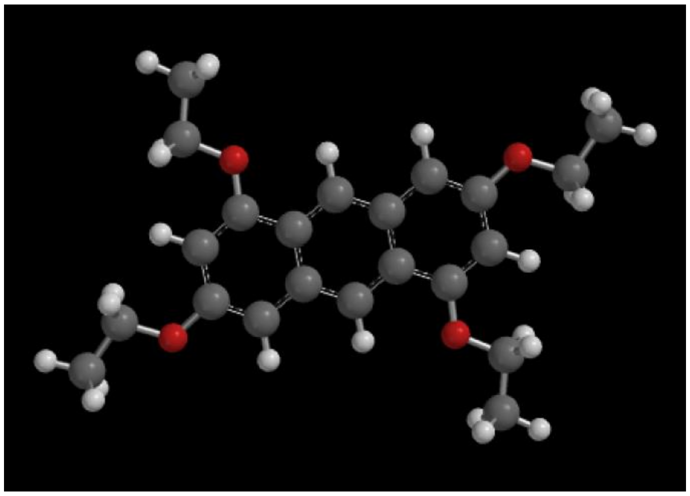
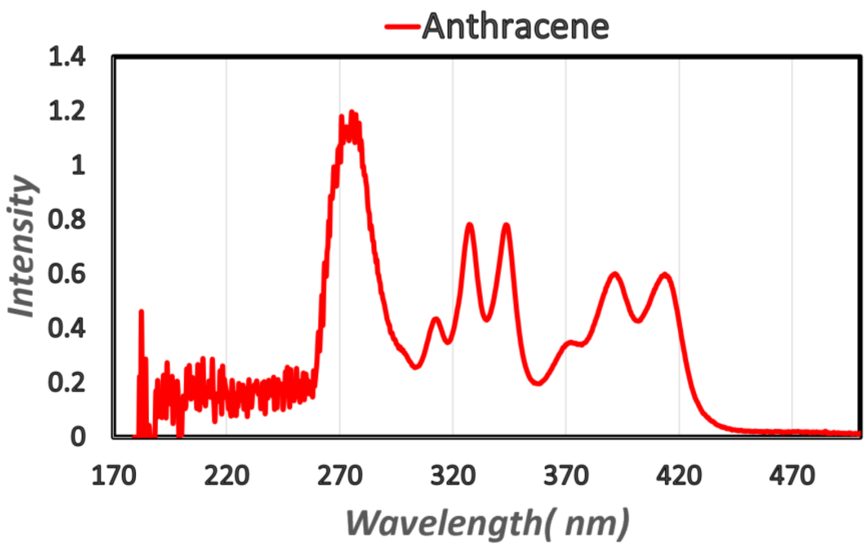


Figure 19. Numerical results (Left) of simulation for the MTI and DAN thin film absorption spectrum. Comparison of theoretical results (Numerical) with the experimental absorption spectrum.





Functional	Basis Set	HOMO (eV)	LUMO (eV)
ωB97X-D	6-311+G**	-6.4	0.2

Figure 20. MTI structure (Left) and anthracene absorption spectrum (excitation at 3.0 eV) (Right). DFT calculations of HOMO and LUMO orbitals (Bottom).

N	10
Vmax	0
DeltaIP (eV)	2.6
DeltaEA (eV)	3.1
U-V (eV)	1.26
ED* (eV)	1.7731
EA*=ED* (eV)+X	0.25
JDA (eV)	0.1
μ_A	1
μ_D	1
μ_CT	4
Tg (eV)	0.3125
gam LE (eV)	0.2083
gam HE (eV)	0.4166
wcut (eV)	2.73

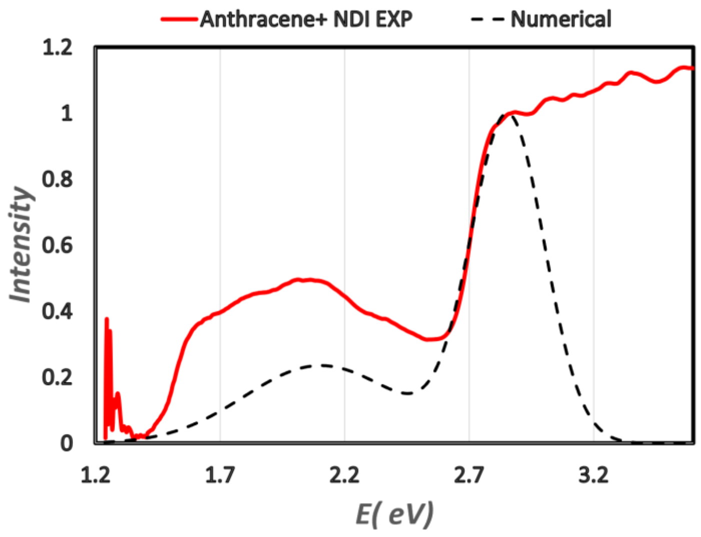


Figure 21. Numerical results (Left) of simulation for the NDI and anthracene thin film absorption spectrum. Comparison of theoretical results (Numerical) with the experimental absorption spectrum.

Table 10. HOMO and LUMO energy levels for MTI and anthracene.

Molecule	HOMO (eV)	LUMO (eV)
MTI	-7.9	-3.6
Anthracene	-4.7	-1.5

References

[1]Leight, Katie R.; Esarey, Brooke E.; Murray, Alex E.; **Reczek, Joseph J.** "Modular and Predictable Tuning of Absorption Properties in Aromatic Donor-Acceptor Materials" *Chem. Mater.* **2012,** *24*, 3318-3328.

[2]Blackburn, A. K.; Sue, A. C. H.; Shveyd, A. K.; Cao, D.; Tayi, A. Narayanan, A.; Rolczynski, B. S.; Szarko, J. M.; Bozdemir, O. A.; Wakabayashi, R.; Lehrman, J. A.; Kahr, B.; Chen, L. X.; Nassar, M. S.; Stupp, S. I.; Stoddart, J. F. Lock-Arm supramolecular ordering: A construction set for cocrystallizing organic charge transfer complexes *J. Am. Chem. Soc.* **2014**, 136, 17224-17235.

[3]Bachrach, S. M. *Computational Organic Chemistry. Wiley*. 2014, 2nd Edition.

[4]Woller, T.; Banerjee, A.; Sylvetsky, N.; Santra, G.; Deraet, X.; Proft, F. D.; Martin, J. M. L.; Alonso, M. Performance of electronic structure methods for the description of Huckel Mobius interconversions in extended pi-systems. *J. Phys. Chem. A* **2020**, 124, 12, 2380-2397

[5]Chai, J.; Head-Gordon, M. Long-range corrected double-hybrid density functionals. *J. Chem. Phys.* **2009**, 131, 174105.

[6]Safia, H.; Ismahan, L.; Abdelkrim, G.; Mouna, C.; Leila, N.; Fatiha, M. Density functional theories study of the interactions between host β-Cyclodextrin and guest 8-Anilinonaphthalene-1-sulfonate: Molecular structure, HOMO, LUMO, NBO,. QTAIM,. And NMR analyses. *Journal of Molecular Liquids* **2019**, 280, 218-229.

[7]Shen, X.; Han, G.; Yi, Y. The nature of excited states in dipolar donor/fullerene complexes for organic solar cells: evolution with the donor stack size. *Physical Chemistry Chemical Physics* **2016**, 23, 15955-15963

[8]Janke, S. M.; Qarai, M. B.; Blum, V.; Spano, F. C. Frenkel-Holstein Hamiltonian applied to absorption spectra of quarterthiophene-based 2D hybrid organic-inorganic perovskites *J. Chem. Phys*. **2020**, 152, 144702.

[9]Ariana Gray Bé, Cheryl Tran, Riley Sechrist, and **Joseph J. Reczek**, Strongly dichroic organic films via controlled assembly of modular aromatic charge-transfer liquid crystals *Org. Lett.* **2015**, *17*, 4834-4837.

[10]Hossen, T.; Sahu, K. Photo-induced electron transfer or proton-coupled electron transfer in methylbipyridine/phenol complexes: a time-dependent density function theory investigation J. Phys. Chem. A **2019**, 123, 8122-8129.

[11]Tuo, De-Hui, et al. "Benzene Triimides: Facile Synthesis and Self-Assembly Study." *Chinese Journal of Chemistry* **2019**, 37.7, 684-688.

Acknowledgments

Dr. Joseph J. Reczek

Mohammad Balooch Qarai

Francis Spano

Dr. Jordan Fantini

Dr. Jordan Katz

Dr. Annabel Edwards

Dr. Timothy Troyer

The Reczek Group

Matthew Kelsey

Denison University Department of Chemistry and Biochemistry

Temple University Department of Chemistry

The Anderson Fund

The W. H. Woodland Prizes in Chemistry

Appendix 1

Oscillator strength:

$$\hat{\mu} = \sum_{j=2,4,6,\dots}^{N_F} \{\vec{\mu}_D|g\rangle\langle j-1| + \vec{\mu}_A|g\rangle\langle j| + \vec{\mu}_{D^+A^-}\{|j+1,j\rangle\langle j+1,j| + |j-1,j\rangle\langle j-1,j|\} + h.c.\}$$

Ground state:

$$|\psi_1\rangle = C_1^{(1)}|g,0;g,0;\dots;g,0\rangle + \sum_{all\ M.P.BS} C_{M.P.S}^{(1)}\,|M.P.BS\rangle$$
$$|\psi_{\delta>1}\rangle = C_1^{(\delta)}|g,0;g,0;\dots;g,0\rangle + \sum_{all\ M.P.BS} C_{M.P.S}^{(\delta)}\,|M.P.BS\rangle$$

$$\langle\psi_1|\hat{\mu}|\psi_{\delta>1}\rangle = \textstyle\sum_{1PE} C_1^{(1)}C_{1PE}^{(\delta)}\mu_n\langle 0|\tilde{v}\rangle + \sum_{1PE} C_{1PE}^{(1)}C_1^{(\delta)}\mu_n\langle 0|\tilde{v}\rangle + \sum_{1PE} C_{1PE}^{(1)}C_{1PE}^{(\delta)}\mu_n$$
$$+ \sum_{\boldsymbol{CTnn}} \boldsymbol{C_{D^+A^-}^{(1)} C_{D^+A^-}^{(\delta)} \mu_{D^+A^-}}$$

Delta function:

$$\delta E_{CT}(t) = a_2 - a_1 = \sqrt{(E_g+\Delta)^2 - 4(E_g\Delta - 2t^2)}$$

$$\delta E_{D^*}(t) = 0.0 - a_1 = \frac{1}{2}\left(E_g+\Delta\right) + \frac{1}{2}\sqrt{(E_g+\Delta)^2 - 4(E_g\Delta - 2t^2)}$$

$$\boldsymbol{E_g = 17.0;\ \Delta = 5.7607}$$

Appendix 2

Code is also found in a separate file on the Google Drive

```
!Copolymer code works; March 25,2020 (started writing on Feb 29, 2020)!
!------------- specify #Maximum Quanta--------------------!
Module common_variables
  implicit none
  Character                                            :: Answer
  Double Precision                        :: dw, w, dt, wavenumber,
wavelength, AbmaxX, AbmaxY, DeltaEps, weV
  Double Precision                        :: Abmaxtot, Fmax
  Double Precision, parameter                        :: t=4.5   !time!
  Double Precision, External   :: FC
  Double Precision, EXTERNAL   :: Repulsion
  Double Precision, EXTERNAL   :: Dopant
  Double Precision, EXTERNAL   :: EnergyCT
  Double Precision, EXTERNAL   :: Tefunction
  Double Precision, EXTERNAL   :: Thfunction
  Double Precision, EXTERNAL   :: Jfunction
  Double Precision, EXTERNAL   :: Dipoles
  !Double Precision, External   :: disorder_table
  Double Precision             :: lambP, lambN, lambEP, lambEN, NPROB,
rand1, rand2, rand1N, rand2N
  Double Precision             :: randR, theta, rand1Nnew, Meandist,
difference
  integer :: i1, j1, i2, j2, f1, f2, i11, j11, i22, j22, i, j, vp1, vn1,
vp2, vn2, neh, f11, f22
  integer :: i3, i33, j3, j33, v3, v33, x3, kount5, kount4, kount3, vp11,
vn11, vp22, vn22
  integer :: x4, v4, x5
  integer :: INFO, LWORK, kount1, kount2, x1, x2, t1, v1, v2, v11, v22,
Kbase, configuration
  integer :: whole
  Double precision                                     :: wholeEl,
wholemean, wholedif, standD, wholePmax
  Double precision                                     :: DelJTintra,
DelJTinter
  integer, parameter
      :: Realization = 1000                      !realization!
  integer, parameter
      :: N = 10
  integer, parameter
      :: Cell = 1
  integer, parameter                                   :: vibmax = 0
  integer, parameter                                   :: vibmaxT = 0
  integer, parameter                                   :: B1 =
N*(vibmax+1)  !1PE states)!
  integer, parameter                                   :: B2 = (N*(N-
1)*vibmax*(vibmax+1))/2   !2PE states!
```

```
  integer, parameter                                    :: B3 =
(N)*(vibmax+1)*(vibmax+2)       !CTn.n states!
  integer, parameter                                    :: B4 = ((N)*(N-
2)*vibmaxT*(vibmaxT+1)*(vibmaxT+2))/3       !CTn.nvib states!
  integer, parameter                                    :: B5 = ((N-
2)*(2*N-3)*vibmax*(vibmax-1)*(2*vibmax+2))/12       !3PE states!
  integer, parameter                                    :: B =
(B1+B2+B3+B4+B5)
  integer, parameter                                    :: LDA = B
  integer, parameter                                    :: Z= 10000
  integer, parameter                                    :: y= 10
  Double Precision                                      :: beta = 2.35d0
  Double Precision                                      :: sigma = 0.5
  Double Precision                                      :: randmean = 0.0
  Double Precision                                      :: distance =
2.1d0
  Double Precision                                      :: danion = 4.1d0
  Double Precision                                      :: lamb = 0.0d0
  Double Precision                                      :: Sfactor = 1.0
  Double Precision, parameter                           :: VMU = 2.88
  Double Precision                                      :: wdstar = 0.0
  Double Precision                                      :: wastar = -1.959
  Double Precision                                      :: wdnap = 14.98
  Double Precision, parameter                           :: wdpan = -13.83
  !Double Precision                                      :: wdndp = 0.0
  Double Precision, parameter                           :: monomer_E =
21000.d0
  Double Precision, parameter                           :: wcm = 1400.0d0
  Double Precision, parameter                           :: Etha = 0.0
!ionicity coefficient!
  Double Precision                                      :: wGround = -
(((monomer_E/wcm)*(1.0-Etha))-((-wdpan-VMU)*Etha))
  Double Precision, parameter                           :: MuAInitial =
1.0
  Double Precision                                      :: MuD = 1.0
  Double Precision, parameter                           :: MuA = 1.0
  Double Precision                                      :: MuCT = 4.5
!It is along x-axis which is the pi-stack axis! Exciton TDMs are along y
exis!
  Double Precision                                      :: Angle = 0.0
!Angle between dipoles in degree! MuD is considered along y-axis!
  Double Precision                                      :: Alfadegree =
180.0                   !it is a constant to convert from degree to radin!
  !Double Precision                                      :: Teinter = 2.46
  !Double Precision                                      :: Thinter = 2.46
  Double Precision                                      :: Tground = 1.6
  Double Precision                                      :: Teintra = 0.0
  Double Precision                                      :: Thintra = 0.0
  Double Precision, parameter                           :: JDAInitial =
0.40
  Double Precision                                      :: JDA =
(MuA/MuAInitial)*JDAInitial
  Double Precision, parameter                           :: JDD = 0.0
  Double Precision                                      :: JAA = 0.0
  Double Precision                                      :: D = 0.0
  Double Precision                                      :: wvib = 1.0
  Double Precision                                      :: Wmin = 2000.0
!for wavenumber!
  Double Precision                                      :: Wmax = 49600.0
! for wavenumber!
  Double Precision                                      :: Wcut =
19506.590                   ! for wavenumber!
  !Double Precision                                      :: Wmin = 0.0d0
  !Double Precision                                      :: Wmax = 15.0d0
  Double Precision                                      :: gamLE = 2.26
  Double Precision                                      :: gamHE = 2.28
  Real, parameter                                       :: PI = 3.1415927
  complex*16, parameter                                 :: XJ =
(0.d0,1.d0)
  Double Precision, dimension(:,:), allocatable                     :: H,
HS, OLH1, OLH2, HX, HY
  Double Precision, dimension(:,:), allocatable                     ::
HSLL, HSLS, HSSL, HSSS
  Double Precision, dimension(:,:), allocatable                     ::
HLLOff, HLSOff, HSSOff, HSLOff
  Double Precision, dimension(:,:,:), allocatable       ::
disorder_elements, wholeP
  Double Precision, dimension(:,:), allocatable                     ::
H1PE, H2PE, H3PE, HCTnn, HCTnnv
  Double Precision, dimension(:,:), allocatable                     ::
OL1CT2P, OL22PCT, OL1CT1P, OL21PCT
  Double Precision, dimension(:,:), allocatable                     ::
OL1CTV2P, OL22PCTV, OL1CTV3P, OL23PCTV
  Double Precision, dimension(:,:), allocatable                     ::
OL1CTVCT, OL2CTCTV, OL12P1P, OL21P2P
  Double Precision, dimension(:,:), allocatable                     ::
OL13P2P, OL22P3P
  Integer, dimension(:,:), allocatable                  :: indx1
  Integer, dimension(:,:,:,:), allocatable              :: indx2
  Integer, dimension(:,:,:,:), allocatable              :: indx3
  Integer, dimension(:,:,:,:,:,:), allocatable          :: indx4
  Integer, dimension(:,:,:,:,:,:), allocatable          :: indx5
  Double Precision, dimension(:), allocatable           :: WORK, OS, Freq,
AbX, AbY, Abtot
  Double Precision, dimension(:,:), allocatable           :: LSX, LSY
  Double Precision, dimension(:), allocatable           :: nA, nD1, nD2,
SummX, SummY, FX, FY
  Double Precision, dimension(:), allocatable           :: COF1PE, COF2PE,
COFCTnn, COFCTnnv, COF3PE
  Double Precision, dimension(:), allocatable           :: COFDpAm,
HGround
  Real(kind=8), dimension(:), allocatable               :: Eign
  !-----------------------------disorderTable-----------------------------
------!
  integer                                               :: Vx, Vy
  !Double Precision                                      :: rand
  !Double Precision                                      :: dlarnd
  !External                                       :: dlarnd
  integer                                                  :: config
  !integer, parameter                                        :: idist = 3
  !integer                                                   ::
iseed(4)=(/47,3093,1041,77/)
  !-------------------External Subroutines-------------------------------
---------------!
  EXTERNAL                                DSYEV
  EXTERNAL                                PRINT_MATRIX
  !EXTERNAL                                disorder_table
  !--------------------------------Logical Parameters--------------------
------------------!
  !logical    :: LSDiagonal = .true.
  !logical    :: SLDiagonal = .true.
  !logical    :: SSDiagonal = .true.
  !logical    :: LLOffDiagonal = .true.
  !logical    :: LSOffDiagonal = .true.
  !logical    :: SLOffDiagonal = .true.
  !logical    :: SSOffDiagonal = .true.
  !--------------------Intrinsic Functions------------------------------
----------------!
  Intrinsic                               INT, MIN
  !---------------------------------------------------------------------
------!
end module common_variables
!---------------------------------------------Start the main program------
-------------------------------------!
Program Copolymer
  use common_variables
  Implicit none

!-----------------------Start Executive part-----------------------------
-------------!
dw  = (Wmax - Wmin)/(Z-1)

lambP = sqrt(0.5)*lamb        !Ground neutral to Excited Cation!
lambN = sqrt(0.5)*lamb        !!Ground neutral to Excited Anion!
lambEP = lamb - lambP         !Excited Frenkel to Excited Cation!
lambEN = lamb - lambN         !Excited Frenkel to Excited Anion!


allocate(H(B+1,B+1))
allocate(HX(B,B))
allocate(HY(B,B))
!allocate(disorder_elements(Realization,N,N))
!allocate(wholeP(Realization,N,N))
allocate(OL1CT1P(B3,B1), OL21PCT(B1,B3))
allocate(OL12P1P(B2,B1), OL21P2P(B1,B2))
allocate(OL13P2P(B5,B2), OL22P3P(B2,B5))
allocate(OL1CT2P(B3,B2), OL22PCT(B2,B3))
allocate(OL1CTV2P(B4,B2), OL22PCTV(B2,B4))
allocate(OL1CTV3P(B4,B5), OL23PCTV(B5,B4))
allocate(OL1CTVCT(B4,B3), OL2CTCTV(B3,B4))
!allocate(HSLL(B,B), HSLS(B,B), HSSL(B,B), HSSS(B,B))
!allocate(HLLOff(B,B), HLSOff(B,B), HSSOff(B,B), HSLOff(B,B))
```

```
allocate(indx1(N,vibmax+1))
allocate(H1PE(B1,B1), H2PE(B2,B2), HCTnn(B3,B3), HCTnnv(B4,B4),
H3PE(B5,B5))
allocate(indx2(N,vibmax,N,vibmax))
allocate(indx3(N,vibmax+1,N,vibmax+1))
allocate(indx4(N,vibmaxT,N,vibmaxT,N,vibmaxT))
allocate(indx5(N,vibmax-1,N,vibmax-1,N,vibmax-1))
allocate(HS(B+1,B+1), Eign(B+1), SummX(B), SummY(B), FX(B), FY(B),
Freq(B), OS(B-1), LSX(Z,B), LSY(Z,B))
allocate(nA(Cell), nD1(Cell), nD2(Cell), AbX(Z), AbY(Z), Abtot(Z),
HGround(B3))
allocate(COF1PE(B+1), COF2PE(B+1), COFCTnn(B+1), COFCTnnv(B+1),
COF3PE(B+1), COFDpAm(B+1))
!-------------------------------HR factor for cation and anion relative to
S0 and S1--------------------------!

!--------------------------------------------------------------------------
----------------------------------------!
!-------------------------------Form the disorder table-------------------
---------------------!
!call disorder_table()

!---------------------------------Coordinate table for DAD units----------
--------------------------------------!
do i1 = 1,Cell
  nA(i1) = 3*i1-1
  nD1(i1) = 3*i1-2
  nD2(i1) = 3*i1
end do

print *, 'JDA=',JDA,''
print *, 'wGround=',wGround,''
!--------------------------------------------------------------------------
--------------------!
print *, 'Now Hamiltonian will be formed'
!-------------Formation of Hamiltonian-------------------------------------
----!
!----------------------Index-----------------------------------------------
-------!
!-------------------------1PE index-----------------------------------!
                kount1 = 0
                do i1 =1,N
                  do j1 =1,(vibmax+1)
                    v1 = j1 - 1
                      kount1 = kount1 + 1
                      indx1(i1,j1) = kount1
                            !print *, 'index <',i1,'/',i2,'> and
<',v1,'/',v2,'> is ',indx1(i1,j1,i2,j2),''
                    end do
                end do
                print *, 'kount1=',kount1,''
!------------------------Formation of the 1PE Hamiltonian----------------
----!
                    t1 = 0
                    do i1 =1,N
                      do j1 =1,(vibmax+1)
                        v1 = j1 - 1
                          x1 = indx1(i1,j1)
                          do i11 =1,N
                            do j11 =1,(vibmax+1)
                              v11 = j11 - 1
                                x2 = indx1(i11,j11)
                                if (x2.eq.x1) then
                                      if (MOD(i1,2).eq.0) then
                                         H1PE(x1,x2) = (wastar + D) +
(wvib*(v1*1.0))
                                      else
                                         H1PE(x1,x2) = (wdstar + D) +
(wvib*(v1*1.0))
                                       end if
                                else if ((iabs(i11-i1)==1).or.(iabs(i11-
i1)==N-1).or.(iabs(i11-i1)==2))  then      !for Periodic!
                                !else if (iabs(i11-i1)==1)  then        !
for open!
                                  H1PE(x1,x2) =
Jfunction(i1,i11,JDA,JDD,JAA,N)*FC(t1,v1,lamb)*FC(t1,v11,lamb)
                                else
                                  H1PE(x1,x2) = 0.0
                                end if
                          end do
                        end do
                    end do
                  end do
                !  print *, 'here is H1PE'
                  !do x1 = 1,B1
                                !  write(*,59) (H1PE(x1,x2), x2=1,B1)

                !  end do
                !  59         format (8f9.4)
!------------------------------2PE index-------------------------------------
----!
                kount2 = 0
                do i1 =1,N
                  do j1 =1,vibmax
                    v1 = j1 - 1
                  do i2=1,N
                    if (i2.eq.i1) cycle
                    do j2 =1,vibmax
                      v2 = j2
                      if ((v1+v2)>vibmax) cycle
                              kount2 = kount2 + 1
                              indx2(i1,j1,i2,j2) = kount2
                          !    print *, indx2(i1,j1,i2,j2,i3,j3)
                      end do
                    end do
                  end do
                end do
                print *, 'kount2=',kount2,''
print *, 'TEST'
!-----------------------------------Formation of 2PE Hamiltonian---------
-!
!---------------------------------------------------------------------
------!
              t1 = 0
              do i1 =1,N
                do j1 =1,vibmax
                  v1 = j1 - 1
                do i2=1,N
                  if (i2.eq.i1) cycle
                  do j2 =1,vibmax
                    v2 = j2
                    if ((v1+v2)>vibmax) cycle
                          x1 = indx2(i1,j1,i2,j2)
                          do i11 =1,N
                            do j11 =1,vibmax
                              v11 = j11 - 1
                            do i22=1,N
                              if (i22.eq.i11) cycle
                              do j22 =1,vibmax
                                v22 = j22
                                if ((v11+v22)>vibmax) cycle
                                x2 = indx2(i11,j11,i22,j22)
      if (x2.eq.x1) then
        if (MOD(i1,2).eq.0) then
           H2PE(x1,x2) = (wastar + D) + (wvib*(v1*1.0 + v2*1.0))
        else
           H2PE(x1,x2) = (wdstar + D) + (wvib*(v1*1.0 + v2*1.0))
        end if
      else if ((iabs(i11-i1)==1).or.(iabs(i11-i1)==N-1).or.(iabs(i11-
i1)==2)) then           !for periodic!
      !else if (iabs(i11-i1)==1) then           !for open!
            if ((i22==i2).and.(v22==v2)) then
                 H2PE(x1,x2) =
Jfunction(i1,i11,JDA,JDD,JAA,N)*FC(t1,v1,lamb)*FC(t1,v11,lamb)
            else if ((i22==i1).and.(i11==i2)) then
                 H2PE(x1,x2) =
Jfunction(i1,i11,JDA,JDD,JAA,N)*FC(v22,v1,lamb)*FC(v2,v11,lamb)

            else
                 H2PE(x1,x2) = 0.0
            end if
      else
        H2PE(x1,x2) = 0.0
      end if
                      end do
                    end do
                  end do
                end do
              end do
            end do
          end do
        end do
```

```
  print *, 'TEST2'
                    !print *, 'here is H2PE'
                    !do x1 = 1,B2
                                !write(*,64) (H2PE(x1,x2), x2=1,B2)

                    !end do
                    !64         format (18f9.4)
!---------------------------------------------------------------------------
--------------------------!
!----------------------------------------------3PE index--------------------
--------------!
                kount5 = 0
                do i1 =1,N
                  do j1 =1,(vibmax-1)
                    v1 = j1 - 1
                  do i2=1,N
                    if (i2.eq.i1) cycle
                    do j2 =1,(vibmax-1)
                      v2 = j2
                      do i3=1,N
                        !if ((i3.eq.i2).and.(i3.eq.i1)) cycle
                        if ((i3.LE.i2)) cycle
                        if ((i3.eq.i1)) cycle
!if (((iabs(i3-i1).ne.1).or.(iabs(i3-i1).ne.N-1)).and.((iabs(i2-
i1).ne.1).or.(iabs(i2-i1).ne.N-1))) cycle
!if ((iabs(i3-i1).EQ.1).or.(iabs(i3-i1).EQ.N-1).or.(iabs(i2-
i1).EQ.1).or.(iabs(i2-i1).EQ.N-1)) then
if ((iabs(i3-i1).EQ.1).or.(iabs(i2-i1).EQ.1)) then
                      do j3 =1,(vibmax-1)
                        v3 = j3
                      if ((v1+v2+v3)>vibmax) cycle
                            kount5 = kount5 + 1
                            indx5(i1,j1,i2,j2,i3,j3) = kount5
                            !print *, indx5(i1,j1,i2,j2,i3,j3)
                            !print *, 'index
<',i1,'/',i2,'/',i3,'/',v1,'/',v2,'/',v3,'> is
',indx5(i1,j1,i2,j2,i3,j3),''
                          end do
                        end if
                        end do
                      end do
                    end do
                  end do
                end do
                print *, 'kount5=',kount5,''
!----------------------------------------------Formation of 3PE
Hamiltonian----------------------------------!
                        t1 = 0
                        do i1 =1,N
                          do j1 =1,(vibmax-1)
                            v1 = j1 - 1
                          do i2=1,N
                            if (i2.eq.i1) cycle
```

```
                              do j2 =1,(vibmax-1)
                                v2 = j2
                                do i3=1,N
                                  if ((i3.LE.i2)) cycle
                                  if ((i3.eq.i1)) cycle
  if ((iabs(i3-i1).EQ.1).or.(iabs(i2-i1).EQ.1)) then
                                do j3 =1,(vibmax-1)
                                  v3 = j3
                                if ((v1+v2+v3)>vibmax) cycle
                                x1 = indx5(i1,j1,i2,j2,i3,j3)
                                do i11 =1,N
                                  do j11 =1,(vibmax-1)
                                    v11 = j11 - 1
                                  do i22=1,N
                                    if (i22.eq.i11) cycle
                                    do j22 =1,(vibmax-1)
                                      v22 = j22
                                      do i33=1,N
                                        if ((i33.LE.i22)) cycle
                                        if ((i33.eq.i11)) cycle
if ((iabs(i33-i11).EQ.1).or.(iabs(i22-i11).EQ.1)) then
                                      do j33 =1,(vibmax-1)
                                        v33 = j33
                                      if ((v11+v22+v33)>vibmax) cycle
                                      x2 = indx5(i11,j11,i22,j22,i33,j33)
                                      if (x2.eq.x1) then
                                        if (MOD(i1,2).eq.0) then
                                           H3PE(x1,x2) = (wastar + D) +
(wvib*(v1*1.0 + v2*1.0 + v3*1.0))
                                        else
                                           H3PE(x1,x2) = (wdstar + D) +
(wvib*(v1*1.0 + v2*1.0 + v3*1.0))
                                        end if
                                      else if ((iabs(i11-
i1)==1).or.(iabs(i11-i1)==N-1).or.(iabs(i11-i1)==2)) then         !for
periodic!
                                      !else if (iabs(i11-i1)==1) then
!for open!
                                        if
((i11==i2).and.(i22==i1).and.(i33==i3).and.(v33==v3)) then
                                          H3PE(x1,x2) =
Jfunction(i1,i11,JDA,JDD,JAA,N)*FC(v2,v11,lamb)*FC(v22,v1,lamb)
                                        else if
((i11==i2).and.(i33==i1).and.(i22==i3).and.(v22==v3)) then
                                          H3PE(x1,x2) =
Jfunction(i1,i11,JDA,JDD,JAA,N)*FC(v2,v11,lamb)*FC(v33,v1,lamb)
                                        else if
((i11==i3).and.(i22==i1).and.(i33==i2).and.(v33==v2)) then
                                          H3PE(x1,x2) =
Jfunction(i1,i11,JDA,JDD,JAA,N)*FC(v3,v11,lamb)*FC(v22,v1,lamb)
                                        else if
((i11==i3).and.(i33==i1).and.(i22==i2).and.(v22==v2)) then
                                          H3PE(x1,x2) =
Jfunction(i1,i11,JDA,JDD,JAA,N)*FC(v3,v11,lamb)*FC(v33,v1,lamb)
```

```
                                      else if
((i22==i2).and.(v22==v2).and.(i33==i3).and.(v33==v3)) then
                                        H3PE(x1,x2) =
Jfunction(i1,i11,JDA,JDD,JAA,N)*FC(t1,v11,lamb)*FC(t1,v1,lamb)
                                      else if
((i22==i3).and.(v22==v3).and.(i33==i2).and.(v33==v2)) then
                                        H3PE(x1,x2) =
Jfunction(i1,i11,JDA,JDD,JAA,N)*FC(t1,v11,lamb)*FC(t1,v1,lamb)
                                      else
                                        H3PE(x1,x2) = 0.0
                                      end if
                                    else
                                      H3PE(x1,x2) = 0.0
                                    end if
                                  end do
                                end if
                                end do
                              end do
                            end do
                          end do
                        end do
                      end do
                    end if
                    end do
                  end do
                end do
              end do
            end do
            !print *, 'here is H3PE'
            !do x1 = 1,B5
                            !write(*,89) (H3PE(x1,x2), x2=1,B5)

            !end do
            !89         format (12f9.3)
!--------------------------------------------------------------------------
----------------------!
!-------------------------------------------CTnn index--------------------
---------------------------!
                            kount3 = 0
                            do i1=1,N
                              do j1=1,(vibmax+1)
                                vp1 = j1 - 1
                                do i2=1,N
                                  if (i2==i1) cycle
                                  if ((iabs(i2-i1).EQ.1).or.(iabs(i2-
i1).EQ.N-1)) then
                                    do j2=1,(vibmax+1)
                                      vn2 = j2 - 1
                                    if ((vp1+vn2)>vibmax) cycle
                                          kount3 = kount3 + 1
                                          indx3(i1,j1,i2,j2) = kount3
                                          !print *, indx3(i1,j1,i2,j2)
                                  end do
                                end if
```

```
                                    end do
                                  end do
                                end do
                                print *, 'kount3=',kount3,''
!------------------------------------------Formation of CTnn-----------
------------------------!
                t1 = 0
                do i1=1,N
                  do j1=1,(vibmax+1)
                    vp1 = j1 - 1
                    do i2=1,N
                      if (i2==i1) cycle
                      if ((iabs(i2-i1).EQ.1).or.(iabs(i2-i1).EQ.N-1))
then
                        do j2=1,(vibmax+1)
                          vn2 = j2 - 1
                        if ((vp1+vn2)>vibmax) cycle
                            x1 = indx3(i1,j1,i2,j2)
                             !print *, 'indexX1
<',i1,'/',vp1,'/',i2,'/',vn2,'>'
                             !print *, 'X1=',x1,''
                            do i11=1,N
                              do j11=1,(vibmax+1)
                                vp11 = j11 - 1
                                do i22=1,N
                                  if (i22==i11) cycle
                                  if ((iabs(i22-i11).EQ.1).or.(iabs(i22-
i11).EQ.N-1)) then
                                    do j22=1,(vibmax+1)
                                      vn22 = j22 - 1
                                    if ((vp11+vn22)>vibmax) cycle
                                        x2 = indx3(i11,j11,i22,j22)
                                            if (x2.eq.x1) then

                    HCTnn(x1,x2) =
(EnergyCT(i1,i2,wdnap,wdpan,N,Sfactor,VMU) + D) +
(Wvib*((vp1*1.0)+(vn2*1.0)))
                                            else if
((i11==i1).and.(vp11==vp1).and.(iabs((i22-i2))==1)) then
!electron moves!
                                          HCTnn(x1,x2) =
Tefunction(i2,i22,Teintra,N)*FC(t1,vn1,lambN)*FC(t1,vn22,lambN)
                                            else if
((i11==i1).and.(vp11==vp1).and.(iabs((i22-i2))==N-1)) then           !e-
BC!
                                          HCTnn(x1,x2) =
Tefunction(i2,i22,Teintra,N)*FC(t1,vn1,lambN)*FC(t1,vn22,lambN)
                                            else if ((iabs(i11-
i1)==1).and.(i22==i2).and.(vn22==vn2)) then             !hole moves!
                                          HCTnn(x1,x2) =
Thfunction(i1,i11,Thintra,N)*FC(t1,vp2,lambP)*FC(t1,vp11,lambP)
                                            else if ((iabs(i11-i1)==N-
1).and.(i22==i2).and.(vn22==vn2)) then              !h-BC!
```

```
                                           HCTnn(x1,x2) =
Thfunction(i1,i11,Thintra,N)*FC(t1,vp2,lambP)*FC(t1,vp11,lambP)
                                             else
                                               HCTnn(x1,x2) = 0.0
                                             end if
                                end do
                              end if
                              end do
                             end do
                           end do
                         end do
                       end if
                       end do
                     end do
                   end do
                   !print *, 'here is HCTnn'
                   !do x1 = 1,B3
                                  !write(*,99) (HCTnn(x1,x2), x2=1,B3)

                   !end do
                   !99         format (18f9.3)
!------------------------------------------------------------------------
---------------------------------------------!
!-----------------------------Ground state to ionic D+A- electronic
coupling---------------------------------------------!
!notice that only the CTnn states will have nonezero coupling terms which
would be multiplied with FC factors!
!CTnnv states will not couple!
                         t1 = 0
                         do i1=1,N
                           do j1=1,(vibmax+1)
                             vp1 = j1 - 1
                             do i2=1,N
                               if (i2==i1) cycle
                               if ((iabs(i2-i1).EQ.1).or.(iabs(i2-
i1).EQ.N-1)) then
                                 do j2=1,(vibmax+1)
                                   vn2 = j2 - 1
                                 if ((vp1+vn2)>vibmax) cycle
                                     x1 = indx3(i1,j1,i2,j2)
                                     if (MOD(i2,2).eq.0) then
                                     HGround(x1) =
Tground*FC(t1,vp1,lambP)*FC(t1,vn2,lambN)
                                     else
                                     HGround(x1) = 0.0
                                     end if
                                   end do
                                 end if
                               end do
                             end do
                           end do
!-----------------------------------------------------------CTnnv index-----
---------------------------------------------!
                       kount4 = 0
```

```
                        do i1 =1,N
                          do j1 =1,vibmaxT
                            vp1 = j1 - 1
                          do i2=1,N
                            if (i2.eq.i1) cycle
                            if ((iabs(i2-i1).EQ.1).or.(iabs(i2-i1).EQ.N-
1)) then
                            do j2 =1,vibmaxT
                              vn2 = j2 -1
                              do i3=1,N
                                !if ((i3.eq.i1).or.(i3.eq.i2)) cycle
                                if (i3.EQ.i1) cycle
                                if (i3.EQ.i2) cycle
                              do j3 =1,vibmaxT
                                v3 = j3
                              if ((vp1+vn2+v3)>vibmaxT) cycle
                                    kount4 = kount4 + 1
                                    indx4(i1,j1,i2,j2,i3,j3) = kount4
                                    !print *, 'index
<',i1,'/',i2,'/',i3,'/',vp1,'/',vn2,'/',v3,'> is
',indx4(i1,j1,i2,j2,i3,j3),''
                                  end do
                                end do
                              end do
                            end if
                            end do
                          end do
                        end do
                        print *, 'kount4=',kount4,''
!---------------------------------------------Formation of CTnnv-----------
-----------------------!
                t1 = 0
                do i1 =1,N
                  do j1 =1,vibmaxT
                    vp1 = j1 - 1
                  do i2=1,N
                    if (i2.eq.i1) cycle
                    if ((iabs(i2-i1).EQ.1).or.(iabs(i2-i1).EQ.N-1)) then
                    do j2 =1,vibmaxT
                      vn2 = j2 -1
                      do i3=1,N
                        if (i3.EQ.i1) cycle
                        if (i3.EQ.i2) cycle
                      do j3 =1,vibmaxT
                        v3 = j3
                      if ((vp1+vn2+v3)>vibmaxT) cycle
                      x1 = indx4(i1,j1,i2,j2,i3,j3)
                      !print *, 'indexX1
<',i1,'/',vp1,'/',i2,'/',vn2,'/',i3,'/',v3,'>'
                      !print *, 'X1=',x1,''
                      do i11 =1,N
                        do j11 =1,vibmaxT
                          vp11 = j11 - 1
                        do i22=1,N
```

```
                          if (i22.eq.i11) cycle
                          if ((iabs(i22-i11).EQ.1).or.(iabs(i22-i11).EQ.N-
1)) then
                          do j22 =1,vibmaxT
                            vn22 = j22 - 1
                            do i33=1,N
                              if (i33.EQ.i11) cycle
                              if (i33.EQ.i22) cycle
                            do j33 =1,vibmaxT
                              v33 = j33
                            if ((vp11+vn22+v33)>vibmaxT) cycle
                            x2 = indx4(i11,j11,i22,j22,i33,j33)
                            !print *, 'indexX2
<',i11,'/',vp11,'/',i22,'/',vn22,'/',i33,'/',v33,'>'
                            !print *, 'X1=',x2,''
                            if (x2.eq.x1) then
            HCTnnv(x1,x2) = (EnergyCT(i1,i2,wdnap,wdpan,N,Sfactor,VMU) +
D) + (Wvib*((vp1*1.0)+(vn2*1.0)+(v3*1.0)))
                            else if
((i11==i1).and.(vp11==vp1).and.(iabs((i22-i2))==1)) then
!electron moves!
                              if ((i33==i3).and.(v33==v3)) then
                                    HCTnnv(x1,x2) =
Tefunction(i2,i22,Teintra,N)*FC(t1,vn1,lambN)*FC(t1,vn22,lambN)
                              else if ((i33==i2).and.(i22==i3)) then
                                    HCTnnv(x1,x2) =
Tefunction(i2,i22,Teintra,N)*FC(v33,vn1,lambN)*FC(v3,vn22,lambN)
                              end if
                            else if
((i11==i1).and.(vp11==vp1).and.(iabs((i22-i2))==N-1)) then
!electron BC!
                              if ((i33==i3).and.(v33==v3)) then
                                    HCTnnv(x1,x2) =
Tefunction(i2,i22,Teintra,N)*FC(t1,vn1,lambN)*FC(t1,vn22,lambN)
                              else if ((i33==i2).and.(i22==i3)) then
                                    HCTnnv(x1,x2) =
Tefunction(i2,i22,Teintra,N)*FC(v33,vn1,lambN)*FC(v3,vn22,lambN)
                              end if
                            else if ((iabs(i11-
i1)==1).and.(i22==i2).and.(vn22==vn2)) then             !hole moves!
                              if ((i33==i3).and.(v33==v3)) then
                                    HCTnnv(x1,x2) =
Thfunction(i1,i11,Thintra,N)*FC(t1,vp1,lambP)*FC(t1,vp11,lambP)
                              else if ((i33==i1).and.(i11==i3)) then
                                    HCTnnv(x1,x2) =
Thfunction(i1,i11,Thintra,N)*FC(v33,vp1,lambP)*FC(v3,vp11,lambP)
                              end if
                            else if ((iabs(i11-i1)==N-
1).and.(i22==i2).and.(vn22==vn2)) then             !hole BC!
                              if ((i33==i3).and.(v33==v3)) then
                                    HCTnnv(x1,x2) =
Thfunction(i1,i11,Thintra,N)*FC(t1,vp1,lambP)*FC(t1,vp11,lambP)
                              else if ((i33==i1).and.(i11==i3)) then
```

```
                                HCTnnv(x1,x2) =
Thfunction(i1,i11,Thintra,N)*FC(v33,vp1,lambP)*FC(v3,vp11,lambP)
                              end if
                            else
                              HCTnnv(x1,x2) = 0.0
                            end if
                          end do
                        end do
                      end do
                    end if
                    end do
                  end do
                end do
                end do
                end do
                end do
              end if
                end do
                end do
                end do

          !      print *, 'here is HCTnnv'
          !      do x1 = 1,B4
            !                    write(*,65) (HCTnnv(x1,x2), x2=1,B4)

          !      end do
          !      65         format (4f9.3)
!---------------------------------------------------------------------------
----------------------!
!----------------------------------------------OL12P1P  and  OL21P2P------
-------------------------!
                    t1 = 0
                    do i1 =1,N
                      do j1 =1,vibmax
                        v1 = j1 - 1
                      do i2=1,N
                        if (i2.eq.i1) cycle
                        do j2 =1,vibmax
                          v2 = j2
                          if ((v1+v2)>vibmax) cycle
                                x1 = indx2(i1,j1,i2,j2)
                                do i11 =1,N
                                  do j11 =1,(vibmax+1)
                                    v11 = j11 - 1
                                      x2 = indx1(i11,j11)
                                    if ((iabs(i11-i1)==1).or.(iabs(i11-
i1)==N-1).or.(iabs(i11-i1)==2)) then
                                      if (i11==i2) then
                                        OL12P1P(x1,x2) =
Jfunction(i1,i11,JDA,JDD,JAA,N)*FC(t1,v1,lamb)*FC(v2,v11,lamb)
                                        OL21P2P(x2,x1) = OL12P1P(x1,x2)
                                      else
                                        OL12P1P(x1,x2) = 0.0
                                        OL21P2P(x2,x1) = OL12P1P(x1,x2)
```

```
                                              end if
                                            end if
                                            end do
                                          end do
                                        end do
                                      end do
                                    end do
                                  end do

                                     !print *, 'here is OL12P1P'
                                     !do x1 = 1,B2
                                                     !write(*,76) (OL12P1P(x1,x2),
x2=1,B1)

                                     !end do
                                     !76         format (9f9.3)

                                     !print *, 'here is OL21P2P'
                                     !do x1 = 1,B1
                                                     !write(*,46) (OL21P2P(x1,x2),
x2=1,B2)

                                     !end do
                                     !46         format (18f9.3)

!-----------------------------------------------OL13P2P  and  OL22P3P------
--------------------------!
                                     t1 = 0
                                     do i1 =1,N
                                       do j1 =1,(vibmax-1)
                                         v1 = j1 - 1
                                       do i2=1,N
                                         if (i2.eq.i1) cycle
                                         do j2 =1,(vibmax-1)
                                           v2 = j2
                                           do i3=1,N
                                             if ((i3.LE.i2)) cycle
                                             if ((i3.eq.i1)) cycle
       if ((iabs(i3-i1).EQ.1).or.(iabs(i2-i1).EQ.1)) then
                                           do j3 =1,(vibmax-1)
                                             v3 = j3
                                           if ((v1+v2+v3)>vibmax) cycle
                                           x1 = indx5(i1,j1,i2,j2,i3,j3)
                                           do i11 =1,N
                                             do j11 =1,vibmax
                                               v11 = j11 - 1
                                             do i22=1,N
                                               if (i22.eq.i11) cycle
                                               do j22 =1,vibmax
                                                 v22 = j22
                                                 if ((v11+v22)>vibmax) cycle
                                                 x2 = indx2(i11,j11,i22,j22)
                                                    if ((iabs(i11-
i1)==1).or.(iabs(i11-i1)==N-1).or.(iabs(i11-i1)==2)) then
```

```
                                              if
((i11==i2).and.(i22==i3).and.(v22==v3)) then
                              OL13P2P(x1,x2) =
Jfunction(i1,i11,JDA,JDD,JAA,N)*FC(t1,v1,lamb)*FC(v2,v11,lamb)
                              OL22P3P(x2,x1) = OL13P2P(x1,x2)
                                              else if
((i11==i3).and.(i22==i2).and.(v22==v2)) then
                              OL13P2P(x1,x2) =
Jfunction(i1,i11,JDA,JDD,JAA,N)*FC(t1,v1,lamb)*FC(v3,v11,lamb)
                              OL22P3P(x2,x1) = OL13P2P(x1,x2)
                                              else
                                                OL13P2P(x1,x2) = 0.0
                                                OL22P3P(x2,x1) =
OL13P2P(x1,x2)
                                              end if
                                            end if
                                            end do
                                          end do
                                        end do
                                      end do
                                    end do
                                  end if
                                  end do
                                end do
                              end do
                            end do
                          end do
                                  !print *, 'here is OL13P2P'
                                  !do x1 = 1,B5
                                                  !write(*,43)
(OL13P2P(x1,x2), x2=1,B2)

                                  !end do
                                  !43         format (9f9.3)

                                  !print *, 'here is OL21P2P'
                                  !do x1 = 1,B2
                                                  !write(*,38)
(OL22P3P(x1,x2), x2=1,B5)

                                  !end do
                                  !38         format (18f9.3)
!----------------------------------------------OL1CT1P  and  OL21PCT------
--------------------------!
t1 = 0
do i1=1,N
  do j1=1,(vibmax+1)
    vp1 = j1 - 1
    do i2=1,N
      if (i2==i1) cycle
      if ((iabs(i2-i1).EQ.1).or.(iabs(i2-i1).EQ.N-1)) then
        do j2=1,(vibmax+1)
          vn2 = j2 - 1
        if ((vp1+vn2)>vibmax) cycle
```

```
            x1 = indx3(i1,j1,i2,j2)
          do i11=1,N
           do j11=1,(vibmax+1)
                  x2 = indx1(i11,j11)
                  v11 = j11 - 1
                  if ((i11==i1).and.(iabs((i11-i2))==1)) then
!e-moves!
                    OL1CT1P(x1,x2) =
Tefunction(i2,i11,Teintra,N)*FC(vp1,v11,lambEP)*FC(t1,vn2,lambN)
                    OL21PCT(x2,x1) = OL1CT1P(x1,x2)
                  else if ((i11==i1).and.(iabs((i11-i2))==N-1)) then
!e-BC!
                    OL1CT1P(x1,x2) =
Tefunction(i2,i11,Teintra,N)*FC(vp1,v11,lambEP)*FC(t1,vn2,lambN)
                    OL21PCT(x2,x1) = OL1CT1P(x1,x2)
                  else if ((iabs(i11-i1)==1).and.(i11==i2)) then
!h-moves!
                    OL1CT1P(x1,x2) =
Thfunction(i1,i11,Thintra,N)*FC(vn2,v11,lambEN)*FC(t1,vp1,lambP)
                    OL21PCT(x2,x1) = OL1CT1P(x1,x2)
                  else if ((iabs(i11-i1)==N-1).and.(i11==i2)) then
!h-BC!
                    OL1CT1P(x1,x2) =
Thfunction(i1,i11,Thintra,N)*FC(vn2,v11,lambEN)*FC(t1,vp1,lambP)
                    OL21PCT(x2,x1) = OL1CT1P(x1,x2)
                  else
                    OL1CT1P(x1,x2) = 0.0
                    OL21PCT(x2,x1) = OL1CT1P(x1,x2)
                  end if
                                end do
                              end do
                            end do
                          end if
                          end do
                        end do
                      end do
                       !print *, 'here is OL1CT1P'
                       !do x1 = 1,B3
                                       !write(*,94) (OL1CT1P(x1,x2),
x2=1,B1)

                       !end do
                       !94         format (9f9.3)

                       !print *, 'here is OL21PCT'
                       !do x1 = 1,B1
                                      ! write(*,27) (OL21PCT(x1,x2),
x2=1,B3)

                       !end do
                      ! 27         format (16f9.3)
!---------------------------------------------------------OL1CT2P and
OL22PCT-------------------------!
t1 = 0
```

```
do i1=1,N
  do j1=1,(vibmax+1)
    vp1 = j1 - 1
    do i2=1,N
      if (i2==i1) cycle
      if ((iabs(i2-i1).EQ.1).or.(iabs(i2-i1).EQ.N-1)) then
        do j2=1,(vibmax+1)
          vn2 = j2 - 1
        if ((vp1+vn2)>vibmax) cycle
            x1 = indx3(i1,j1,i2,j2)
          do i11=1,N
            do j11=1,vibmax
                v11 = j11 - 1
              do i22=1,N
                if (i22==i11) cycle
                do j22=1,vibmax
                  v22 = j22
                    if ((v11+v22)>vibmax) cycle
                    x2 = indx2(i11,j11,i22,j22)
                        if ((i11==i1).and.(iabs(i11-i2)==1).and.(i22==i2))
then                   !e-moves!
                          OL1CT2P(x1,x2) =
Tefunction(i2,i11,Teintra,N)*FC(vp1,v11,lambEP)*FC(v22,vn2,lambN)
                          OL22PCT(x2,x1) = OL1CT2P(x1,x2)
                        else if ((i11==i1).and.(iabs(i11-i2)==N-
1).and.(i22==i2)) then                  !e-BC!
                          OL1CT2P(x1,x2) =
Tefunction(i2,i11,Teintra,N)*FC(vp1,v11,lambEP)*FC(v22,vn2,lambN)
                          OL22PCT(x2,x1) = OL1CT2P(x1,x2)
                        else if ((iabs(i11-
i1)==1).and.(i11==i2).and.(i22==i1)) then              !h-moves!
                          OL1CT2P(x1,x2) =
Thfunction(i1,i11,Thintra,N)*FC(vn2,v11,lambEN)*FC(v22,vp1,lambP)
                          OL22PCT(x2,x1) = OL1CT2P(x1,x2)
                        else if ((iabs(i11-i1)==N-
1).and.(i11==i2).and.(i22==i1)) then              !h-BC!
                          OL1CT2P(x1,x2) =
Thfunction(i1,i11,Thintra,N)*FC(vn2,v11,lambEN)*FC(v22,vp1,lambP)
                          OL22PCT(x2,x1) = OL1CT2P(x1,x2)
                        else
                          OL1CT2P(x1,x2) = 0.0
                          OL22PCT(x2,x1) = OL1CT2P(x1,x2)
                        end if
                      end do
                    end do
                   end do
                 end do
              end do
            end if
             end do
           end do
         end do
         !print *, 'here is OL1CT2P'
         !do x1 = 1,B3
```

```
                       !write(*,83) (OL1CT2P(x1,x2), x2=1,B2)

         !end do
         !83         format (6f9.3)

         !print *, 'here is OL22PCT'
         !do x1 = 1,B2
                      ! write(*,46) (OL22PCT(x1,x2), x2=1,B3)

         !end do
         !46         format (12f9.3)
!---------------------------------------------------OL1CTV2P and OL22PCTV---
--------------------------!
t1 = 0
do i1 =1,N
  do j1 =1,vibmaxT
    vp1 = j1 - 1
  do i2=1,N
    if (i2.eq.i1) cycle
    if ((iabs(i2-i1).EQ.1).or.(iabs(i2-i1).EQ.N-1)) then
    do j2 =1,vibmaxT
      vn2 = j2 -1
      do i3=1,N
        if (i3.EQ.i1) cycle
        if (i3.EQ.i2) cycle
      do j3 =1,vibmaxT
        v3 = j3
      if ((vp1+vn2+v3)>vibmaxT) cycle
      x1 = indx4(i1,j1,i2,j2,i3,j3)
      do i11=1,N
        do j11=1,vibmax
            v11 = j11 - 1
          do i22=1,N
            if (i22==i11) cycle
            do j22=1,vibmax
              v22 = j22
                if ((v11+v22)>vibmax) cycle
                x2 = indx2(i11,j11,i22,j22)
                if ((i11==i1).and.(iabs(i1-
i2)==1).and.(i22==i3).and.(v22==v3)) then                   !e-moves!
                  OL1CTV2P(x1,x2) =
Tefunction(i2,i11,Teintra,N)*FC(vp1,v11,lambEP)*FC(t1,vn2,lambN)
                  OL22PCTV(x2,x1) = OL1CTV2P(x1,x2)
                else if ((i11==i1).and.(iabs(i1-i2)==N-
1).and.(i22==i3).and.(v22==v3)) then                   !e-BC!
                  OL1CTV2P(x1,x2) =
Tefunction(i2,i11,Teintra,N)*FC(vp1,v11,lambEP)*FC(t1,vn2,lambN)
                  OL22PCTV(x2,x1) = OL1CTV2P(x1,x2)
                else if ((iabs(i1-
i2)==1).and.(i11==i2).and.(i22==i3).and.(v22==v3)) then              !h-
moves!
                  OL1CTV2P(x1,x2) =
Thfunction(i1,i11,Thintra,N)*FC(vn2,v11,lambEN)*FC(t1,vp1,lambP)
                  OL22PCTV(x2,x1) = OL1CTV2P(x1,x2)
                else if ((iabs(i1-i2)==N-
1).and.(i11==i2).and.(i22==i3).and.(v22==v3)) then              !h-BC!
                  OL1CTV2P(x1,x2) =
Thfunction(i1,i11,Thintra,N)*FC(vn2,v11,lambEN)*FC(t1,vp1,lambP)
                  OL22PCTV(x2,x1) = OL1CTV2P(x1,x2)
                else
                  OL1CTV2P(x1,x2) = 0.0
                  OL22PCTV(x2,x1) = OL1CTV2P(x1,x2)
                end if
              end do
            end do
          end do
        end do
     end do
  end do
end do
end if
end do
end do
end do

!print *, 'here is OL1CTV2P'
!do x1 = 1,B4
                !write(*,43) (OL1CTV2P(x1,x2), x2=1,B2)

!end do
!43         format (6f9.3)

!print *, 'here is OL22PCTV'
!do x1 = 1,B2
                !write(*,95) (OL22PCTV(x1,x2), x2=1,B4)

!end do
!95         format (4f9.3)
!-----------------------------------------------------OL1CTV3P and
OL23PCTV-----------------------------------------!
t1 = 0
do i1 =1,N
  do j1 =1,vibmaxT
    vp1 = j1 - 1
  do i2=1,N
    if (i2.eq.i1) cycle
    if ((iabs(i2-i1).EQ.1).or.(iabs(i2-i1).EQ.N-1)) then
    do j2 =1,vibmaxT
      vn2 = j2 -1
      do i3=1,N
        if (i3.EQ.i1) cycle
        if (i3.EQ.i2) cycle
      do j3 =1,vibmaxT
        v3 = j3
      if ((vp1+vn2+v3)>vibmaxT) cycle
      x1 = indx4(i1,j1,i2,j2,i3,j3)
      do i11 =1,N
        do j11 =1,(vibmax-1)
```

```
          v11 = j11 - 1
        do i22=1,N
          if (i22.eq.i11) cycle
          do j22 =1,(vibmax-1)
            v22 = j22
            do i33=1,N
              if ((i33.LE.i22)) cycle
              if ((i33.eq.i11)) cycle
if ((iabs(i33-i11).EQ.1).or.(iabs(i22-i11).EQ.1)) then
            do j33 =1,(vibmax-1)
              v33 = j33
            if ((v11+v22+v33)>vibmax) cycle
            x2 = indx5(i11,j11,i22,j22,i33,j33)
           if ((i11==i1).and.(iabs((i2-i11))==1)) then
!electron moves!
            if ((i22==i2).and.(i33==i3).and.(v33==v3)) then
                  OL1CTV3P(x1,x2) =
Tefunction(i2,i11,Teintra,N)*FC(vp1,v11,lambEP)*FC(v22,vn2,lambN)
                  OL23PCTV(x2,x1) = OL1CTV3P(x1,x2)
            else if ((i22==i3).and.(i33==i2).and.(v22==v3)) then
                  OL1CTV3P(x1,x2) =
Tefunction(i2,i11,Teintra,N)*FC(vp1,v11,lambEP)*FC(v33,vn2,lambN)
                  OL23PCTV(x2,x1) = OL1CTV3P(x1,x2)
            end if
          else if ((i11==i1).and.(iabs((i2-i11))==N-1)) then
!electron BC!
           if ((i22==i2).and.(i33==i3).and.(v33==v3)) then
                 OL1CTV3P(x1,x2) =
Tefunction(i2,i11,Teintra,N)*FC(vp1,v11,lambEP)*FC(v22,vn2,lambN)
                 OL23PCTV(x2,x1) = OL1CTV3P(x1,x2)
           else if ((i22==i3).and.(i33==i2).and.(v22==v3)) then
                 OL1CTV3P(x1,x2) =
Tefunction(i2,i11,Teintra,N)*FC(vp1,v11,lambEP)*FC(v33,vn2,lambN)
                 OL23PCTV(x2,x1) = OL1CTV3P(x1,x2)
           end if
          else if ((iabs(i11-i1)==1).and.(i11==i2)) then
!hole moves!
            if ((i22==i1).and.(i33==i3).and.(v33==v3)) then
                  OL1CTV3P(x1,x2) =
Thfunction(i1,i11,Thintra,N)*FC(vn2,v11,lambEN)*FC(v22,vp1,lambP)
                  OL23PCTV(x2,x1) = OL1CTV3P(x1,x2)
            else if ((i33==i1).and.(i22==i3).and.(v22==v3)) then
                  OL1CTV3P(x1,x2) =
Thfunction(i1,i11,Thintra,N)*FC(vn2,v11,lambEN)*FC(v33,vp1,lambP)
                  OL23PCTV(x2,x1) = OL1CTV3P(x1,x2)
            end if
          else if ((iabs(i11-i1)==N-1).and.(i11==i2)) then
!hole BC!
            if ((i22==i1).and.(i33==i3).and.(v33==v3)) then
                  OL1CTV3P(x1,x2) =
Thfunction(i1,i11,Thintra,N)*FC(vn2,v11,lambEN)*FC(v22,vp1,lambP)
                  OL23PCTV(x2,x1) = OL1CTV3P(x1,x2)
            else if ((i33==i1).and.(i22==i3).and.(v22==v3)) then
```

```
                  OL1CTV3P(x1,x2) =
Thfunction(i1,i11,Thintra,N)*FC(vn2,v11,lambEN)*FC(v33,vp1,lambP)
                  OL23PCTV(x2,x1) = OL1CTV3P(x1,x2)
            end if
          else
            OL1CTV3P(x1,x2) = 0.0
            OL23PCTV(x2,x1) = OL1CTV3P(x1,x2)
          end if
          end do
        end if
        end do
      end do
    end do
 end do
end do
end do
end do
end do
end if
end do
end do
end do

!print *, 'here is OL1CTV3P'
!do x1 = 1,B4
                !write(*,92) (OL1CTV2P(x1,x2), x2=1,B5)

!end do
!92         format (3f9.3)

!print *, 'here is OL23PCTV'
!do x1 = 1,B5
                !write(*,83) (OL22PCTV(x1,x2), x2=1,B4)

!end do
!83         format (16f9.3)
!--------------------------------------------OL1CTVCT and OL2CTCTV------
----------------------------------------------!
t1 = 0
do i1 =1,N
  do j1 =1,vibmaxT
    vp1 = j1 - 1
  do i2=1,N
    if (i2.eq.i1) cycle
    if ((iabs(i2-i1).EQ.1).or.(iabs(i2-i1).EQ.N-1)) then
    do j2 =1,vibmaxT
      vn2 = j2 -1
      do i3=1,N
        if (i3.EQ.i1) cycle
        if (i3.EQ.i2) cycle
      do j3 =1,vibmaxT
        v3 = j3
      if ((vp1+vn2+v3)>vibmaxT) cycle
      x1 = indx4(i1,j1,i2,j2,i3,j3)
      do i11=1,N
        do j11=1,(vibmax+1)
          vp11 = j11 - 1
          do i22=1,N
            if (i22==i11) cycle
            if ((iabs(i22-i11).EQ.1).or.(iabs(i22-i11).EQ.N-1)) then
              do j22=1,(vibmax+1)
                vn22 = j22 - 1
              if ((vp11+vn22)>vibmax) cycle
                  x2 = indx3(i11,j11,i22,j22)
           if ((i11==i1).and.(vp11==vp1).and.(iabs((i2-
i22))==1).and.(i22==i3)) then              !electron moves!
             OL1CTVCT(x1,x2) =
Tefunction(i2,i22,Teintra,N)*FC(v3,vn22,lambN)*FC(t1,vn1,lambN)
             OL2CTCTV(x2,x1) = OL1CTVCT(x1,x2)
           else if ((i11==i1).and.(vp11==vp1).and.(iabs((i2-i22))==N-
1).and.(i22==i3)) then              !electron BC!
             OL1CTVCT(x1,x2) =
Tefunction(i2,i22,Teintra,N)*FC(v3,vn22,lambN)*FC(t1,vn1,lambN)
             OL2CTCTV(x2,x1) = OL1CTVCT(x1,x2)
          else if ((iabs(i11-
i1)==1).and.(i22==i2).and.(vn22==vn1).and.(i11==i3)) then
!hole moves!
            OL1CTVCT(x1,x2) =
Thfunction(i1,i11,Thintra,N)*FC(v3,vp11,lambP)*FC(t1,vp1,lambP)
            OL2CTCTV(x2,x1) = OL1CTVCT(x1,x2)
          else if ((iabs(i11-i1)==N-
1).and.(i22==i2).and.(vn22==vn1).and.(i11==i3)) then              !hole
BC!
            OL1CTVCT(x1,x2) =
Thfunction(i1,i11,Thintra,N)*FC(v3,vp11,lambP)*FC(t1,vp1,lambP)
            OL2CTCTV(x2,x1) = OL1CTVCT(x1,x2)
          else
            OL1CTVCT(x1,x2) = 0.0
            OL2CTCTV(x2,x1) = OL1CTVCT(x1,x2)
          end if
          end do
        end if
        end do
      end do
    end do
 end do
end do
end do
end if
end do
end do
end do
  print *, 'TEST3'
  !print *, 'here is OL1CTVCT'
  !do x1 = 1,B4
                !write(*,18) (OL1CTVCT(x1,x2), x2=1,B3)

  !end do
  !18         format (12f9.3)

  !print *, 'here is OL2CTCTV'
  !do x1 = 1,B3
                  !write(*,16) (OL2CTCTV(x1,x2), x2=1,B4)

  !end do
  !16         format (4f9.3)

!----------------------------Now, Form the total Hamiltonian--------------
-------------------!
                                  do i =1,(B+1)
                                    do j =1,(B+1)
                                     if (i==1) then
                                       if
((j>(B1+B2+1)).and.(j<=(B1+B2+B3+1))) then
                                         x2 = j - (B1+B2+1)
                                         H(i,j) = Hground(x2)
                                         H(j,i) = H(i,j)
                                       else if (j==1) then
                                         H(i,j) = wGround
                                       else
                                         H(i,j) = 0.0
                                         H(j,i) = 0.0
                                       end if
                                     end if
                                     end do
                                   end do

                                   do i =1,B
                                     do j =1,B
                                       if (i<=B1) then
                                         if (j<=B1) then
                                                           H(i+1,j+1) =
H1PE(i,j)
                                          else if ((j>B1).and.(j<=(B1+B2)))
then
                                                    x2 = j - B1
                                                         H(i+1,j+1) =
OL21P2P(i,x2)
                                          else if
((j>(B1+B2)).and.(j<=(B1+B2+B3))) then
                                                   x2 = j - (B1+B2)
                                                   H(i+1,j+1) =
OL21PCT(i,x2)
                                          else if
((j>(B1+B2+B3)).and.(j<=(B1+B2+B3+B4))) then
                                                   x2 = j - (B1+B2+B3)
                                                        H(i+1,j+1) = 0.0
                                          else if
((j>(B1+B2+B3+B4)).and.(j<=B)) then
                                                                x2 = j -
(B1+B2+B3+B4)
```

```
H(i+1,j+1) = 0.0

!print *, 'TEST3'
                                                      end if
                                                      !print *, 'TEST4'
                                                   else if ((i>B1).and.(i<=(B1+B2)))
then
                                                      x1 = i - B1
                                                      if (j<=B1) then
                                                                              H(i+1,j+1) =
OL12P1P(x1,j)
                                                                              !print *,
'TEST5'
                                                     else if ((j>B1).and.(j<=(B1+B2)))
then
                                                                   x2 = j - B1
                                                                         H(i+1,j+1) =
H2PE(x1,x2)
                                                                         !print *, 'TEST6'
                                                     else if
((j>(B1+B2)).and.(j<=(B1+B2+B3))) then
                                                                  x2 = j - (B1+B2)
                                                                  H(i+1,j+1) =
OL22PCT(x1,x2)
                                                                  !print *, 'TEST7'
                                                     else if
((j>(B1+B2+B3)).and.(j<=(B1+B2+B3+B4))) then
                                                                  x2 = j - (B1+B2+B3)
                                                                        H(i+1,j+1) =
OL22PCTV(x1,x2)
                                                                        !print *, 'TEST8'
                                                     else if
((j>(B1+B2+B3+B4)).and.(j<=B)) then
                                                                                    x2 = j -
(B1+B2+B3+B4)

H(i+1,j+1) = OL22P3P(x1,x2)

!H(i,j) = 0.0          !for test!

!print *, 'TEST9'
                                                     end if

                                                  else if
((i>(B1+B2)).and.(i<=(B1+B2+B3))) then
                                                      x1 = i - (B1+B2)
                                                      if (j<=B1) then
                                                                              H(i+1,j+1) =
OL1CT1P(x1,j)
                                                                              !print *,
'TEST10'
                                                     else if ((j>B1).and.(j<=(B1+B2)))
then
```

```
                                                    x2 = j - B1
                                                          H(i+1,j+1) =
OL1CT2P(x1,x2)
                                                          !print *, 'TEST11'
                                        else if
((j>(B1+B2)).and.(j<=(B1+B2+B3))) then
                                                   x2 = j - (B1+B2)
                                                   H(i+1,j+1) = HCTnn(x1,x2)
                                                   !print *, 'TEST12'
                                        else if
((j>(B1+B2+B3)).and.(j<=(B1+B2+B3+B4))) then
                                                   x2 = j - (B1+B2+B3)
                                                         H(i+1,j+1) =
OL2CTCTV(x1,x2)
                                                         !print *, 'TEST8'
                                        else if
((j>(B1+B2+B3+B4)).and.(j<=B)) then
                                                                x2 = j -
(B1+B2+B3+B4)

H(i+1,j+1) = 0.0

!print *, 'TEST13'
                                        end if
                                        !print *, 'TEST6'
                                      else if
((i>(B1+B2+B3)).and.(i<=(B1+B2+B3+B4))) then
                                         x1 = i - (B1+B2+B3)
                                         if (j<=B1) then
                                                            H(i+1,j+1) = 0.0
                                        else if ((j>B1).and.(j<=(B1+B2)))
then
                                                    x2 = j - B1
                                                          H(i+1,j+1) =
OL1CTV2P(x1,x2)
                                        else if
((j>(B1+B2)).and.(j<=(B1+B2+B3))) then
                                                   x2 = j - (B1+B2)
                                                   H(i+1,j+1) =
OL1CTVCT(x1,x2)
                                        else if
((j>(B1+B2+B3)).and.(j<=(B1+B2+B3+B4))) then
                                                   x2 = j - (B1+B2+B3)
                                                         H(i+1,j+1) =
HCTnnv(x1,x2)
                                        else if
((j>(B1+B2+B3+B4)).and.(j<=B)) then
                                                              x2 = j -
(B1+B2+B3+B4)
                                                              H(i+1,j+1) =
OL1CTV3P(x1,x2)
                                        end if
                                      else if
((i>(B1+B2+B3+B4)).and.(i<=B)) then
```

```
                                        x1 = i - (B1+B2+B3+B4)
                                        if (j<=B1) then
                                                         H(i+1,j+1) = 0.0
                                       else if ((j>B1).and.(j<=(B1+B2)))
then
                                                  x2 = j - B1
                                                        H(i+1,j+1) =
OL13P2P(x1,x2)
                                                        !H(i,j) = 0.0
!for test!
                                       else if
((j>(B1+B2)).and.(j<=(B1+B2+B3))) then
                                                 x2 = j - (B1+B2)
                                                 H(i+1,j+1) = 0.0
                                       else if
((j>(B1+B2+B3)).and.(j<=(B1+B2+B3+B4))) then
                                                 x2 = j - (B1+B2+B3)
                                                       H(i+1,j+1) =
OL23PCTV(x1,x2)
                                       else if
((j>(B1+B2+B3+B4)).and.(j<=B)) then
                                                             x2 = j -
(B1+B2+B3+B4)
                                                             H(i+1,j+1) =
H3PE(x1,x2)
                                       end if
                                      end if
                                   end do
                                   end do




                              !print *, 'here is the total H'
                                        !do x1 = 1,(B+1)
                                                        !write(*,39)
(H(x1,x2), x2=1,(B+1))
                                        !end do
                                   !39         format (13f5.1)
!-------------------------------Check each part of Hamiltonian------------
-------------------!
!print *, 'kount1=',kount1,''
!print *, 'kount2=',kount2,''
!print *, 'kount3=',kount3,''
!print *, 'kount4=',kount4,''
!print *, 'kount5=',kount5,''
print *, 'Sum of kounts=',kount1+kount2+kount3+kount4+kount5,''
print *, 'Dimension of Hamiltonian=',B+1,''
!-----------------------------Check the Hamiltonian-----------------------
------------------!
!print *, 'here is the total H'
!            do x1 = 1,B
!                             write(*,1) (H(x1,x2), x2=1,B)
!            end do
!           1         format (12f5.1)
!-------------------------------Is it correct Hamiltonian?----------------
--------------------!
!print *, 'Is it correct Hamiltonian? if yes, type Y, and if no, type N'
!read *, Answer
!                      if (Answer /= 'Y')  then
!                                  continue
!                      else
!                                  stop
!                      end if
!------------------------------Copy the original Hamiltonian--------------
------------------------!
                   do x1=1,(B+1)
                                   do x2=1,(B+1)
                                                   HS(x1,x2) = H(x1,x2)
                                                   !HSLL(x1,x2) =
H(x1,x2)
                                                   !HSLS(x1,x2) =
H(x1,x2)
                                                   !HSSL(x1,x2) =
H(x1,x2)
                                                   !HSSS(x1,x2) =
H(x1,x2)
                                                   !HLLOff(x1,x2) =
H(x1,x2)
                                                   !HSLOff(x1,x2) =
H(x1,x2)
                                                   !HLSOff(x1,x2) =
H(x1,x2)
                                                   !HSSOff(x1,x2) =
H(x1,x2)
                                   end do
                   end do
!--------------------------------Let's add Disorder----------------------
--------------------!
!Print *, "Diagonal Disorder:)"

!call LLDiagonal()
!call SLDiagonal()
!call LSDiagonal()
!call SSDiagonal()


!print *, "just finished diagonal disorder:))"
!print *, "Off-diagonal disorder--Ugly Face"

!call LLOffDiagonal()
!call SLOffDiagonal()
!call LSOffDiagonal()
!call SSOffDiagonal()

!print *, "just finished Off-diagonal disorder:))"
```

```
!----------------------------------------------------------------------------
-------------------!
!------------------------------NOW, Diagonalize the Hamiltonian------------
------------------!
Write(*,*) 'DSYEV program results'
!-------------------------------------------------------------!
LWORK = -1
allocate(WORK(3*(B)-1))
Call DSYEV('Vector', 'Upper', (B+1), H, (B+1), Eign, WORK, LWORK, INFO)
LWORK=WORK(1)

deallocate(WORK)
allocate(WORK(LWORK))
Call DSYEV('Vector', 'Upper', (B+1), H, (B+1), Eign, WORK, LWORK, INFO)
!----------------------------------------------------------------------------
-!
!-----------------------------Check for Convergence------------------------
--------------!
            if (INFO.gt.0) then
                            write(*,*) 'The algorithm failed to compute
the eigenvalue'
                            stop
            end if
!---------------------------Print the eigenvalues and eigenvectors--------
-------------------!
!CALL PRINT_MATRIX('eigenvalues', 1, B, Eign, 1)
!CALL PRINT_MATRIX('Eigenvectores--stored columnwise', B, B, H, LDA)

!-------------------------------------So, the results are------------------
----------------------!
!-----------------------index 1 and 2 goes over x and y direction 1:y and
2:x----------------------!

do x4 = 1,13
  print *, Eign(x4)
end do
!-----------------------------------------------CT Character--------------
---------------------------!
      do x2 = 1,B
        COF1PE (x2) = 0.0
        COF2PE(x2) = 0.0
        COFCTnn(x2) = 0.0
        COFCTnnv(x2) = 0.0
        COF3PE(x2) = 0.0

        do x4 = 1, B1
          COF1PE(x2) = COF1PE(x2) + H(x4,x2)*H(x4,x2)
        end do

        do x4 = 1, B2
          v4 = x4 + B1
          COF2PE(x2) = COF2PE(x2) + H(v4,x2)*H(v4,x2)
        end do
        do x4 = 1, B3
          v4 = x4 + (B1+B2)
          COFCTnn(x2) = COFCTnn(x2) + H(v4,x2)*H(v4,x2)
        end do

        do x4 = 1, B4
          v4 = x4 + (B1+B2+B3)
          COFCTnnv(x2) = COFCTnnv(x2) + H(v4,x2)*H(v4,x2)
        end do

        do x4 = 1, B5
          v4 = x4 + (B1+B2+B3+B4)
          COF3PE(x2) = COF3PE(x2) + H(v4,x2)*H(v4,x2)
        end do

      end do
!-----------------------------------------the probability of D+A-; among
all the CT states-----------------------------------------!
                            do X2 = 1, B
                              COFDpAm (x2) = 0.0
                                                    !Go over CTnn!
                    do i1=1,N
                      do j1=1,(vibmax+1)
                        vp1 = j1 - 1
                        do i2=1,N
                          if (i2==i1) cycle
                          if ((iabs(i2-i1).EQ.1).or.(iabs(i2-i1).EQ.N-1))
then
                            do j2=1,(vibmax+1)
                              vn2 = j2 - 1
                            if ((vp1+vn2)>vibmax) cycle
                                x1 = indx3(i1,j1,i2,j2)
                                x4 = (B1+B2) + x1

                                if (MOD(i2,2).eq.0) then
                                      COFDpAm(x2) = COFDpAm(x2) +
H(x4,x2)*H(x4,x2)
                                end if
                              end do
                            end if
                              end do
                            end do
                          end do
                                                ! Go over CTnnv!
                                                  do i1 =1,N
                                                    do j1 =1,vibmaxT
                                                      vp1 = j1 - 1
                                                    do i2=1,N
                                                      if (i2.eq.i1) cycle
                                                      if ((iabs(i2-
i1).EQ.1).or.(iabs(i2-i1).EQ.N-1)) then
                                                      do j2 =1,vibmaxT
                                                        vn2 = j2 -1
                                                        do i3=1,N
```

```
                                                          if (i3.EQ.i1)
cycle
                                                          if (i3.EQ.i2)
cycle
                                                        do j3 =1,vibmaxT
                                                          v3 = j3
                                                        if
((vp1+vn2+v3)>vibmaxT) cycle
                                                        x1 =
indx4(i1,j1,i2,j2,i3,j3)
                                                        x4 = (B1+B2+B3) + x1

                                                        if (MOD(i2,2).eq.0)
then
                                                              COFDpAm(x2) =
COFDpAm(x2) + H(x4,x2)*H(x4,x2)
                                                        end if
                                                      end do
                                                    end do
                                                  end do
                                                end if
                                              end do
                                            end do
                                          end do


                      end do
!---------------------------------------------------------------------------
------------------------------!
print *, 'now for OSillator strength'
              do x2 =2,(B+1)
                x3 = x2 - 1

                !Go over 1PE states!
                        do i1 = 1,N
                          do j1 =1,(vibmax+1)
                                  x1 = indx1(i1,j1)
                                  x5 = x1
                                  x1 = x1 + 1

                                  v1 = j1 - 1
                                  t1 = 0
                                  if (MOD(i1,2).eq.0) then
                                    x4 = 2
  !  HX(x5,x3) =
Dipoles(x4,i1,MuA,MuD,Angle,Alfadegree)*((H(1,1)*H(x1,x2)*FC(t1,v1,lamb))+
(H(x1,1)*H(1,x2)*FC(t1,v1,lamb)) &
  !  +(H(x1,1)*H(x1,x2)))
    HX(x5,x3) = 0.0
                                    x4 = 1
    HY(x5,x3) =
Dipoles(x4,i1,MuA,MuD,Angle,Alfadegree)*((H(1,1)*H(x1,x2)*FC(t1,v1,lamb))+
(H(x1,1)*H(1,x2)*FC(t1,v1,lamb)) &
    +(H(x1,1)*H(x1,x2)))
```

```
                                   !print *, 'FCA and v1
<',FC(t1,v1,lamb),'/',v1,'>'
                                   else
                                     x4 = 2
  !  HX(x5,x3) =
Dipoles(x4,i1,MuA,MuD,Angle,Alfadegree)*((H(1,1)*H(x1,x2)*FC(t1,v1,lamb))+
(H(x1,1)*H(1,x2)*FC(t1,v1,lamb)) &
  !  +(H(x1,1)*H(x1,x2)))
    HX(x5,x3) = 0.0
                                     x4 = 1
    HY(x5,x3) =
Dipoles(x4,i1,MuA,MuD,Angle,Alfadegree)*((H(1,1)*H(x1,x2)*FC(t1,v1,lamb))+
(H(x1,1)*H(1,x2)*FC(t1,v1,lamb)) &
    +(H(x1,1)*H(x1,x2)))
                                   !print *, 'FCD and v1
<',FC(t1,v1,lamb),'/',v1,'>'
                                   end if
                           end do
                        end do

                !Go over CTnn states!
                             do i1=1,N
                               do j1=1,(vibmax+1)
                                 vp1 = j1 - 1
                                 do i2=1,N
                                   if (i2==i1) cycle
                                   if ((iabs(i2-i1).EQ.1).or.(iabs(i2-
i1).EQ.N-1)) then
                                     do j2=1,(vibmax+1)
                                       vn2 = j2 - 1
                                     if ((vp1+vn2)>vibmax) cycle
                                         x1 = indx3(i1,j1,i2,j2)
                                         x5 = x1 + B1
                                         !x1 = x1 + B1
                                         x1 = (x1 + 1 + B1 + B2)
                                           if (MOD(i2,2).eq.0) then
                             HX(x5,x3) = MuCT*H(x1,1)*H(x1,x2)
                             !HY(x5,x3) = MuCT*H(x1,1)*H(x1,x2)
                                          else
                             HX(x5,x3) = 0.0
                             HY(x5,x3) = 0.0
                                          end if
                           end do
                         end if
                       end do
                     end do
                   end do

              end do
!................To calculate Oscillator
Strength.......................................................!
                     do x2=1,B

                           SummX (x2) = 0.0
```

```
                          SummY (x2) = 0.0
                          do x1=1,B
                                            SummX(x2) = SummX(x2) +
HX(x1,x2)
                                            SummY(x2) = SummY(x2) +
HY(x1,x2)
                            end do
                          end do
!---------------------------------------!
                   do x2=1,B
                                  SummX(x2) = SummX(x2)*SummX(x2)
                                  SummY(x2) = SummY(x2)*SummY(x2)
                   end do
!----------------------------------------!
print *, 'the values of oscillator strength are as follows: '
                    do x2=1,B

                                 FX(x2) = SummX(x2)
                                 FY(x2) = SummY(x2)
                                 !print *, 'os and f
<',SummX(x2),'/',SummY(x2),'>'
                                 !Freq(x2) = Eign(x2)*wcm*1.d0 +
monomer_E*1.0

                                 !Freq(x2) = Eign(x2)*wcm*1.d0
                                 !F(x2) = Freq(x2)*F(x2)

                                 x3 = x2 + 1
                                 Freq(x2) = (Eign(x3)-Eign(1))*wcm*1.d0
!for wavenumber!
                                 !print *, 'os and f
<',SummX(x2),'/',SummY(x2),'/',Freq(x2),'>'
                                 !Freq(x2) = Eign(x2)*wcm*1.d0/8065.0 +
monomer_E*1.0

                                 FX(x2) = Freq(x2)*FX(x2)            !here
I multiply with frequency!
                                 FY(x2) = Freq(x2)*FY(x2)
                                     !print *, Freq(x2)
                             !            print *, 'freq and f
<',Freq(x2),'/',F(x2),'>'
                                 !F(x2) = Freq(x2)*F(x2)
                                 !write(*,8) F(x2)
                    end do
!      8                         format (1f5.1)

!---------------------------------------------------------------------------
-----------------!
!------------------------------------Form the Line Shape Matrix,
Absorption---------------------!
!Fmax = MAXVAL(FY)
!do x4 = 1,B
    !print *, 'Wavelength and w and f
<',(10000000.0/Freq(x4)),'/',(x4*1.0),'/',(FY(x4)/Fmax),'>'
```

```
!end do

                      gamLE = gamLE*wcm
                      gamHE = gamHE*wcm
                    do x1=1,Z
                      w = Wmin + ((x1-1)*dw)
                      !w = w*wcm*1.d0 + monomer_E*1.0
                      !w = w*wcm*1.d0
                                  AbX(x1) = 0.0
                                  AbY(x1) = 0.0
                                  do x2=1,B
                                           !LS(x1,x2) = DEXP(-((w-
Eign(x2))**2)/(gam**2))
                                           if (Freq(x2).LE.Wcut) then
                                           LSX(x1,x2) =
(1.0d0/gamLE)*DEXP(-((w-Freq(x2))**2)/(gamLE**2))
                                           LSY(x1,x2) =
(1.0d0/gamLE)*DEXP(-((w-Freq(x2))**2)/(gamLE**2))
                                           else
                                           LSX(x1,x2) =
(1.0d0/gamHE)*DEXP(-((w-Freq(x2))**2)/(gamHE**2))
                                           LSY(x1,x2) =
(1.0d0/gamHE)*DEXP(-((w-Freq(x2))**2)/(gamHE**2))
                                           end if
                                           !print *, 'freq and w and f
<',Freq(x2),'/',w,'/',F(x2),'>'
                                           !LS(x1,x2) = (gam**2)/(((w-
Eign(x2))**2)+(gam**2))
                                           !F(x2) = F(x2)*abs((w-
Eign(x2)))
                                           !Ab(x1) = Ab(x1) +
(F(x2)*(Eign(x2))*LS(x1,x2))
                                           !Ab(x1) = Ab(x1) +
(F(x2)*(Freq(x2) - monomer_E*1.0)*LS(x1,x2))
                                           AbX(x1) = AbX(x1) +
(FX(x2)*LSX(x1,x2))
                                           AbY(x1) = AbY(x1) +
(FY(x2)*LSY(x1,x2))
                                           !print *, 'freq and w and f
<',Freq(x2),'/',w,'/',F(x2),'/',Ab(x1),'>'
                                  end do
                                  AbX(x1) = (1.0/N)*AbX(x1)
                                  AbY(x1) = (1.0/N)*AbY(x1)
                                  Abtot(x1) = AbX(x1) + AbY(x1)
                                  !print *, 'w and abs <',w,'/',Ab(x1),'>'
                                !  print *, Ab(x1)
                    end do
                    !renormalize spectrum!
                    AbmaxX = MAXVAL(AbX)
                    AbmaxY = MAXVAL(AbY)
                    Abmaxtot = MAXVAL(Abtot)
                    do x1=1,Z
                      AbX(x1) = AbX(x1)/AbmaxX
                      AbY(x1) = AbY(x1)/AbmaxY
                    Abtot(x1) = Abtot(x1)/Abmaxtot
                  end do
                   !print *, 'check!'
                   !do x1= 1,B
                   !print *, Eign(x1)
                   !end do
!-----------------------------------------Show the absorption for each
frequency-------------------!
print *, 'Here is the spectrum'
open(UNIT=4,FILE="COPVKSVMU1.txt",STATUS="OLD",ACTION="READWRITE")
                            do x1=1,Z
                              w = Wmin + ((x1-1)*(dw))
                              weV = w/8065.0
                              !weV = w*1400.0/8065.0
                              !wavenumber = wcm*1.0*w + monomer_E*1.0
                              !w = (w-1)*wcm*1.d0 + monomer_E*1.0
                              !w = w*wcm*1.d0
                              !w = w*monomer_E*1.0
                              !print *, w
                              !wavenumber = w*wcm + monomer_E
                              wavelength = 10000000.0/w
                                     !write(UNIT=4, FMT="(4(F15.7,2X))")
Ab(x1), w
                                     !write(UNIT=4, FMT="(2(f15.10,2X))")
w, Ab(x1)
                                     !write(UNIT=4, FMT="(4(F15.10,2X))")
wavenumber, Ab(x1)
                                     !write(UNIT=4, FMT=105) weV, AbX(x1),
AbY(x1), Abtot(x1)             !here for eV!
                                      write(UNIT=4, FMT=105) weV, AbX(x1),
AbY(x1), Abtot(x1)             !here for wavelength!
                              !       write(UNIT=4, FMT=105) w, AbX(x1),
AbY(x1), Abtot(x1)              !here for cm-1!
                                     105 format(4e20.12)
                                       !print *, 'w and abs
<',w,'/',Ab(x1),'>'

                            end do
                close(UNIT=4)

print *, 'kount1=',kount1,''
print *, 'kount2=',kount2,''
print *, 'kount3=',kount3,''
print *, 'kount4=',kount4,''
print *, 'kount5=',kount5,''
print *, 'Dimension of Hamiltonian=',B+1,''
print *, 'This is Eignvalue matrix'
!-----------------------------------------------------CT Character----
------------------------------------------------------!
print *, 'Here is the Vector'
open(UNIT=6,FILE="LOOK5.txt",STATUS="OLD",ACTION="READWRITE")
                            do x1=1,B
                              wavelength = (10000000.0/Freq(x1))
```

```
    write(UNIT=6, FMT=106) Wavelength, (COF1PE(x1)+COF2PE(x1)+COF3PE(x1)),
(COFCTnn(x1)+COFCTnnv(x1)), COFDpAm(x1)
                                    106 format(4e20.12)
                                      !print *, 'w and abs
<',w,'/',Ab(x1),'>'

                            end do
                close(UNIT=6)
!--------------------------------------------------------------------------
-------------------------------------------------------------------!
!open(UNIT=5,FILE="EGN1.txt",STATUS="OLD",ACTION="READWRITE")
                      ! do i1=1,B
                                           !  write(UNIT=5,
FMT="(2(F15.7,2X))") Eign(i1), F(i1)
                                             !write(*,322) Eign(i1),
F(i1)
                       !end do
                  !322                       format (6f10.1)
                !  close(UNIT=5)
!----------------------------------------------Dynamics-------------------
--------------------------------------------!
                      !set the initial state:
!--------------------------------------------------------------------------
--------------------------------------------------------------!

end program Copolymer
!--------------------------------------------------------------------------
--------------------!
!-----------------------------------LL Diagonal Disorder------------------
--------------------------!
!--------------------------------------Disorder Subroutine----------------
--------------------------!
Subroutine disorder_table()
  use common_variables
  implicit none
  !------------------------------------!
  !integer                                              :: Vx, Vy
  !Double Precision                                     :: rand
  !real*8, external                                     :: dlarnd
  !integer                                              :: config, i, j,
configmax, N
  !integer, parameter                                   :: idist = 3
  !Double Precision, dimension(:,:,:), allocatable      ::
disorder_elements
  !integer                                              ::
iseed(4)=(/47,3093,1041,77/)
  !----------------------!
call random_seed()
                      do config=1,realization
                              do Vx=1,N
                                do Vy=1,N
                                  call random_number(rand1)
                                  call random_number(rand2)
                                  randR =  dsqrt(-2*log(rand1))
                                  theta = 2.0*PI*rand2
                                  rand1N = randR*cos(theta)
                                  rand2N = randR*sin(theta)
                                  rand1Nnew = (rand1N*sigma) + randmean
                                  disorder_elements(config,Vx,Vy) =
rand1Nnew
                                end do
                              end do
                            end do
                              !-------------------------!
                        !to calculate the distribution!
                            whole = 0
                            wholeEl = 0.0d0
                        do config=1,realization
                                do Vx=1,N
                                  do Vy=1,N
                                      whole = whole + 1
                                        wholeEl = wholeEl +
disorder_elements(config,Vx,Vy)
                                  end do
                                  end do
                                  end do
                                  !-----------------------------------!
                          wholemean = wholeEl/whole    !mean!
                                  wholedif = 0.0d0
                                  do config=1,realization
                                          do Vx=1,N
                                            do Vy=1,N
                                                wholedif = wholedif +
((disorder_elements(config,Vx,Vy)-wholemean)**2)
                                              end do
                                              end do
                                              end do
                                              !-------------------------!
                                              !standarddeviation!
                                              standD = (1.0/(whole -
1))*wholedif

                                              standD = sqrt(standD)
                                              !---------------------------
------!

                                              !Okay, we have the uniform
distribution!

open(UNIT=5,FILE="random3.txt",STATUS="NEW",ACTION="READWRITE")
                            do config=1,realization
                                    do Vx=1,N
                                      do Vy=1,N
       !wholeP(config,Vx,Vy) = (1.0/sqrt(2.0*PI*(standD**2)))*EXP(-
1.0*((disorder_elements(config,Vx,Vy)**2)/(2.0*standD**2)))
       wholeP(config,Vx,Vy) = (1.0/sqrt(2.0*PI*(sigma**2)))*EXP(-
1.0*((disorder_elements(config,Vx,Vy)**2)/(2.0*sigma**2)))
                     write(UNIT=5, FMT="(4(F15.7,2X))")
disorder_elements(config,Vx,Vy), wholeP(config,Vx,Vy)
                                                        end do
```

```
                                                                    end do
                                                                  end do
                                                          close(UNIT=5)
                                                          print *, wholemean, standD
end Subroutine disorder_table
!--------------------------------------------------------------------------
-------------------!
!-----------------------------Auxilary routine for PRINT_MATRIX
Subroutine----------------------!
Subroutine PRINT_MATRIX(DESC, M, N, A, LDA)

  character :: DESC
  integer   :: M, N, LDA
  Double Precision :: A(LDA, *)
  integer          :: i, j

  Write(*,*)
  Write(*,*) DESC

        do i=1,M
                      Write(*,9998) (A(i,j), j=1,N)
        end do

        9998      format(11(:,1X, F6.2))
        return
end Subroutine PRINT_MATRIX
!------------------------------------Coulomb Repulsion--------------------
---------------------------------------!
Function Repulsion(m,n,distance)
  implicit none
  integer                   :: m, n
  Double Precision          :: distance, Repulsion, Rcoulomb

            Rcoulomb = iabs(m-n)*distance

            Repulsion = 8.3/Rcoulomb
            print *, 'Repulsion for <',m,'/',n,'> is ',repulsion,''

end Function Repulsion
!---------------------------------------------Dopant----------------------
---------------------------------------!
Function Dopant(m,n,totaln,distance,danion)
  implicit none
  integer                   :: m, n, totaln, jm, jn
  Double Precision          :: distance, Dopant, Rcoulombm, danion,
Rcoulombn
  Double Precision          :: dpm, dpn

          !notice, I have assumed that we have even number of
chromophores!
                if (m.LE.(INT(totaln/2)).and.n.LE.(INT(totaln/2))) then
                  dpm = ((INT(totaln/2)-m)*1.0*distance)+(distance/2.0)
                  dpn = ((INT(totaln/2)-n)*1.0*distance)+(distance/2.0)
                  Rcoulombm = sqrt((dpm*dpm)+(danion*danion))
                  Rcoulombn = sqrt((dpn*dpn)+(danion*danion))
                  Dopant = (-1.0*(8.3/Rcoulombm)) + (-1.0*(8.3/Rcoulombn))
                else if (m.GT.(INT(totaln/2)).and.n.LE.(INT(totaln/2)))
then
                  jm = m-INT(totaln/2)-1
                  dpm = (jm*1.0*distance)+(distance/2.0)
                  dpn = ((INT(totaln/2)-n)*1.0*distance)+(distance/2.0)
                  Rcoulombm = sqrt((dpm*dpm)+(danion*danion))
                  Rcoulombn = sqrt((dpn*dpn)+(danion*danion))
                  Dopant = (-1.0*(8.3/Rcoulombm)) + (-1.0*(8.3/Rcoulombn))
                else if (m.LE.(INT(totaln/2)).and.n.GT.(INT(totaln/2)))
then
                  jn = n-INT(totaln/2)-1
                  dpm = ((INT(totaln/2)-m)*1.0*distance)+(distance/2.0)
                  dpn = (jn*1.0*distance)+(distance/2.0)
                  Rcoulombm = sqrt((dpm*dpm)+(danion*danion))
                  Rcoulombn = sqrt((dpn*dpn)+(danion*danion))
                  Dopant = (-1.0*(8.3/Rcoulombm)) + (-1.0*(8.3/Rcoulombn))
                else
                  jm = m-INT(totaln/2)-1
                  jn = n-INT(totaln/2)-1
                  dpm = (jm*1.0*distance)+(distance/2.0)
                  dpn = (jn*1.0*distance)+(distance/2.0)
                  Rcoulombm = sqrt((dpm*dpm)+(danion*danion))
                  Rcoulombn = sqrt((dpn*dpn)+(danion*danion))
                  Dopant = (-1.0*(8.3/Rcoulombm)) + (-1.0*(8.3/Rcoulombn))
                end if
            !print *, 'Dopant for <',m,'/',n,'> is ',Dopant,''

end Function Dopant
!-------------------------------------------------------CT Energy Function-
----------------------------------------!
Function EnergyCT(m,n,wdnap,wdpan,Ntot,Sfactor,VMU)
  implicit none
  integer                  :: m, n, t, Ntot
  Double Precision         :: EnergyCT, wdnap, wdpan
  Double Precision         :: Sfactor, VMU
  !Double Precision         :: nA, nD1, nD2

  if ((MOD(m,2).eq.0).and.(iabs(m-n).EQ.1)) then
    EnergyCT = (Sfactor*wdnap) + VMU
    !print *, 'ECT <',m,'/',n,'> is ',EnergyCT,''
  else if ((MOD(m,2).eq.0).and.((m-n).EQ.Ntot-1)) then
    EnergyCT = (Sfactor*wdnap) + VMU
  else if ((MOD(m+1,2).eq.0).and.(iabs(m-n).EQ.1)) then
    EnergyCT = (Sfactor*wdpan) + VMU
    !print *, 'ECT <',m,'/',n,'> is ',EnergyCT,''
  else if ((MOD(m+1,2).eq.0).and.((m-n).EQ.1-Ntot)) then
    EnergyCT = (Sfactor*wdpan) + VMU
  end if

end Function EnergyCT
!-------------------------------------------------------Te Function--------
--------------------------------!
```

```
Function Tefunction(m,n,Teintra,Ntot)
  implicit none
  integer                   :: m, n, Ntot
  Double Precision          :: Tefunction, Teintra
  !Double Precision          :: nA, nD1, nD2


  if ((MOD(m,2).eq.0).and.(iabs(n-m).EQ.1)) then
    Tefunction = Teintra
    !print *, 'Tef is <',m,'/',n,'> is ',Tefunction,''
  else if ((MOD(m+1,2).eq.0).and.(iabs(n-m).EQ.1)) then
    Tefunction = Teintra
    !print *, 'Tef is <',m,'/',n,'> is ',Tefunction,''
  else if ((MOD(m+1,2).eq.0).and.((m-n).EQ.1-Ntot)) then
    Tefunction = Teintra
    !print *, 'Tef is <',m,'/',n,'> is ',Tefunction,''
  else if ((MOD(m,2).eq.0).and.((m-n).EQ.Ntot-1)) then
    Tefunction = Teintra
    !print *, 'Tef is <',m,'/',n,'> is ',Tefunction,''
  end if

end Function Tefunction
!-------------------------------------------------------Th Function--------
-----------------------------!
Function Thfunction(m,n,Thintra,Ntot)
  implicit none
  integer                   :: m, n, Ntot
  Double Precision          :: Thfunction, Thintra
  !Double Precision          :: nA, nD1, nD2

  if ((MOD(m,2).eq.0).and.(iabs(n-m).EQ.1)) then
    Thfunction = Thintra
    !print *, 'Thf is <',m,'/',n,'> is ',Thfunction,''
  else if ((MOD(m+1,2).eq.0).and.(iabs(n-m).EQ.1)) then
    Thfunction = Thintra
    !print *, 'Thf is <',m,'/',n,'> is ',Thfunction,''
  else if ((MOD(m+1,2).eq.0).and.((m-n).EQ.1-Ntot)) then
    Thfunction = Thintra
    !print *, 'Thf is <',m,'/',n,'> is ',Thfunction,''
  else if ((MOD(m,2).eq.0).and.((m-n).EQ.Ntot-1)) then
    Thfunction = Thintra
    !print *, 'Thf is <',m,'/',n,'> is ',Thfunction,''
  end if

end Function Thfunction
!-----------------------------------------------Jfunction------------------
-----------------------------------!
Function Jfunction(m,n,JDA,JDD,JAA,Ntot)
  implicit none
  integer                   :: m, n, Ntot
  Double Precision          :: Jfunction, JDA, JDD, JAA
  !Double Precision          :: nA, nD1, nD2

  !print *, ' JDA ',JDA,''
  if ((MOD(m,2).eq.0).and.(iabs(n-m).EQ.1)) then
    Jfunction = JDA
  else if ((MOD(m+1,2).eq.0).and.(iabs(n-m).EQ.1)) then
    Jfunction = JDA
  else if ((MOD(m,2).eq.0).and.((n-m).EQ.1-Ntot)) then
    Jfunction = JDA
  else if ((MOD(m+1,2).eq.0).and.((n-m).EQ.Ntot-1)) then
    Jfunction = JDA
  else if ((MOD(m,2).eq.0).and.(iabs(n-m).EQ.2)) then
    Jfunction = JAA
  else if ((MOD(m+1,2).eq.0).and.(iabs(n-m).EQ.2)) then
    Jfunction = JDD
  !else if ((MOD(m,2).eq.0).and.((n-m).EQ.Ntot-2)) then
    !Jfunction = JAA
  !else if ((MOD(m+1,2).eq.0).and.((n-m).EQ.Ntot-2)) then
    !Jfunction = JDD
  end if

end Function Jfunction
!-----------------------------------------Dipoles in different directions--
----------------------------------------!
Function Dipoles(m,n,MuA,MuD,Angle,Alfadegree)
  implicit none
  integer                    :: m,n
  Double Precision           :: Dipoles, MuA, MuD
  Double Precision           :: Angle, Alfadegree
  Double Precision           :: DegtoRad, MuxA, MuyA, MuxD, MuyD
  Real, parameter            :: PI = 3.1415927


  DegtoRad = Angle*PI/Alfadegree       !Angle in radian!
  MuxA = MuA*sin(DegtoRad)
  MuyA = MuA*cos(DegtoRad)
  MuyD = MuD                       !Notice that the angle is expressed
with respect to y-axis along which the MuD is poiting!
  MuxD = 0.0
  !print *, ' MuA ',MuA,''
  !print *, ' cosAng ',cos(DegtoRad),''
  !print *, ' SinAng ',sin(DegtoRad),''

  if ((m.EQ.1).and.(MOD(n,2).eq.0)) then                 !in y-direction!
    Dipoles = MuyA
  else if ((m.EQ.1).and.(MOD(n+1,2).eq.0)) then
    Dipoles = MuyD
  else if ((m.EQ.2).and.(MOD(n,2).eq.0)) then            !9n x-direction!
    Dipoles = MuxA
  else if ((m.EQ.2).and.(MOD(n+1,2).eq.0)) then
    Dipoles = MuxD
  end if

end Function Dipoles
!--------------------------------------Frank-Condon Factor, general
formula---------------------------------------!
```

```
Function FC(m,n,lam)
    implicit none
    integer                    :: i, j, k, m, n        !m: #quanta in
ground state   n: #quanta in excited state!
    Double Precision           :: S, FC, C1, D1, Summ, C2, D2, C3, lam, DEN
    Double Precision, External   :: Factorial

S = lam*lam   !lam is HR factor!

C1 = dsqrt(Factorial(m))
D1 = dsqrt(Factorial(n))

                                  !j = MIN(m,n)
                                  j = n
                                  Summ = 0.0
                                  do i=1,(j+1)
                                    k = i - 1
                                    if ((m-n+k).lt.0) cycle
                                          C2 = Factorial(k)
                                          D2 = Factorial((n-k))
                                          C3 = Factorial((m-n+k))
                                          DEN = C2*D2*C3
                                               Summ = Summ + ((((-1)**(m-
n+k))*(S**((m-n+(2*k))/2.0)))/DEN)
                                              end do

                  FC = C1*D1*dexp(-1.0*(S/2.0))*Summ
                  !print *, 'FC factor for <',m,'/',n,'> is ',FC,''
                  !print *, 'lamda ',lam,''
end Function FC
!---------------------------------------------------------------------------
-----------------------------------------!
!-------------------------------------Factorial-----------------------------
----------------------------------------!
Function Factorial(x)
  implicit none
  integer                      :: i, x
  Double Precision             :: Factorial

  Factorial = 1.0
  do i=1,x
    Factorial = Factorial*(1.0*i)
  end do
  !print *, 'Factorial of ',x,' is ',Factorial,''

end Function Factorial
!---------------------------------------------------------------------------
------------------------------------!
```