Title

A Bond-Order Time Series Reaction Event Classifier and a Semi-Automated Tinker Polarizable Force Field Initialization Method

Author

Hutchings, Marshall Eugene

Publication Date

2022

Peer reviewed|Thesis/dissertation

A Bond-Order Time Series Reaction Event Classifier and a Semi-Automated Tinker Polarizable Force Field Initialization Method

By

MARSHALL EUGENE HUTCHINGS

DISSERTATION

Submitted in partial satisfaction of the requirements for the degree of

DOCTOR OF PHILOSOPHY

in

Chemistry

in the

OFFICE OF GRADUATE STUDIES

of the

UNIVERSITY OF CALIFORNIA

DAVIS

Approved:

Lee-Ping Wang, Chair

Matthew Augustine

Davide Donadio

Committee in Charge

2022

A Bond-Order Time Series Reaction Event Classifier and a Semi-Automated Tinker Polarizable Force Field Initialization Method

ABSTRACT

Exploratory chemistry is an important branch of computational chemistry where precursor molecules are simulated in reaction-prone conditions without pre-supposed hypotheses. These molecular dynamics simulations can yield many unexpected reaction pathways, confirming existing mechanisms or suggesting new ones. Decreasing cost of hardware and increasing savviness of software make it enticing to explore reaction spaces in this un-guided manner. However, reactions are still rare events buried in data where the molecules spend most of their time not clearing reaction barriers. Therefore, advanced data processing techniques that can parse the simulations for reaction events with high accuracy are valuable in a field with ever increasing data generation.

In this work, a method is described in Chapter 2 which addresses the problem of extracting valuable reaction location data from exploratory simulations with high accuracy and low cost. The method uses first derivatives on bond order time series obtained from *Ab Initio* molecular dynamics data to predict temporal reaction locations with high accuracy and speed. The two tunable parameters (low pass filtering cutoff and threshold placement on the derivative) are parameterized against a "gold standard" approach that represents the ideal data processing scenario given infinite time and computing resources. This reaction event classifier is showcased on two single-molecule reactive test systems. The first, simpler test case being a heptanylium alkyl carbocation, $C_7H_{15}{}^+$. The other more complicated system is $Fe_3(CO)_9$, which is a highly unsaturated iron carbonyl cluster. The effectiveness of the reaction event classifier is analyzed for both systems using heat maps and other custom plots and metrics.

Deep Eutectic Solvents (DESs) are a relatively new type of mixture which exhibits similar properties to ionic liquids, but with lower cost and environmental impact. Until recently,

most force fields describing DESs were not polarizable and therefore lacked certain behaviors created by their extremely partially-charged environment. This second project began with the goal of creating a polarizable force field for DESs, but eventually morphed into creating a program to automate the Tinker polarizable force field initialization process with Python. The test molecule for refining the initialization process is urea. Urea is a hydrogen bond-donor which when paired with choline chloride in the correct whole-number ratio create a DES. A semi-automated Python method which creates polarizable Tinker force fields with simple setup is explained.

ACKNOWLEDGEMENTS

There are many people who aided me on this journey. I am so grateful for my PhD advisor Lee-Ping Wang. I knew I wanted to go to graduate school studying chemistry, but did not have clear direction beyond that. Lee-Ping's research presentation was so exciting and interesting that I knew what to pursue. It was under your guidance that I really learned to program and leveled up my problem solving. I appreciate your positivity, patience, expertise, and ever abundance of great ideas.

I also want to thank the other members of my dissertation committee: Matthew Augustine and Davide Donadio. Thank you both for reading and reviewing this document, and for being on my QE committee as well.

I want to also thank all the members (both past and present) of the Wang group. Yudong Qiu, thank you for being an immense resource, always friendly and eager to help. Nanhao Chen, you are always so helpful, good at troubleshooting, and fun to talk NBA with. Lisa Gong-Oh, you are so friendly and helped us all hang out much more than we otherwise would have. Hyesu Jang, thank you for being so nice to be around and helping optimize the leftover cookie pickups. Nathan Yoshino, it has been great to talking to you about board games and pedagogy and obtaining leftover cookies. Zhecheng He, you are someone I could really count on as a TA/Head TA and that means a lot to me. Heejune Park, you are fun to talk to about programming and carnivorous plants. Jesi Lee, thank you for the joy and curiosity you bring to our group. Also, thank you Kaixin Wu for working with me on force fields and for creating that DES database.

The content of Chapter 2 is a reprint of the published work of the same name with only small modifications. I want to thank the other authors: Johnson Liu, Yudong Qiu, Chenchen Song, and Lee-Ping Wang for their contributions to "Bond-Order Time Series Analysis for Detecting Reaction Events in *Ab Initio* Molecular Dynamics Simulations". This article can be found here: https://pubs.acs.org/doi/10.1021/acs.jctc.9b01039.

On the utility side of things, I would like to thank the creator(s) of the useful little

online tool at https://www.bruot.org/ris2bib/. This tool converts .ris (Endnote) bibliography entries to .bib (LaTeX) entries which was extremely useful and saved time when citing the many journals that did not natively support LaTeX. A similar tool found at https://www.bibtex.com/c/pmid-to-bibtex-converter/ for converting PubMed bibliography entries to .bib was also useful.

I also want to acknowledge the TMP Chem (https://www.youtube.com/user/tmpchem) and Chris Cramer (https://www.youtube.com/channel/UCQWne9DleuGu0q9TmA70jWQ) computational chemistry YouTube channels. Both are exceptional sources for learning/reviewing computational chemistry topics and were helpful in my journey.

I definitely need to thank the "Physical Kids": Sommer, Shannon, Dan, Zachary, and Zach who began Physical Chemistry graduate study at UC Davis at the same time as me. It was great working on homework together with you, joking around, hanging out, and going on this graduate school journey together. I am excited for when we can all get together again!

I am also grateful for Matthew Asplund for letting me work in his lab during my undergraduate studies. The project was interesting and the techniques were useful and important in my development, thank you!

Thank you to Joe and Trish and the many other members on Stacey's side of the family. I appreciate everything you have done for me and my family and how you are all eager to help us.

I would like to thank my family for their constant support. My parents Ken and Sharlene and siblings Leanna, Nolan, and Rory are so invested in me and my activities and I appreciate their encouragement in this endeavor. Additional thanks to Nolan and Rory for reviewing this document and suggesting edits. I am grateful for the joy that my kids Natalia, Katharina, Kaia, and Camden bring to my life and their patience during this process. I hope each of you develop a love of study and learning and go on to have successful careers. Maybe you will study lots of math and science! I am excited for you all to keep learning and growing.

I am grateful most of all for the love and support of my wife Stacey. You have always been supportive of me and I am glad you are by my side. You are the most important person of all to me. Thank you for your patience and partnership during this graduate school time, and I am excited to see where life takes us next.

Contents

1 Introduction

The PhD work contained in this document is centered on two main projects that are fairly disparate. The first project is described in Chapter 2 and deals with processing quantum mechanics (Section 1 .1) data obtained via exploratory chemistry (Section 1 .2). The second project is described in Chapter 3 and deals with developing molecular mechanics methods (Section 1 .3) to study deep eutectic solvents (Section 1 .4).

1 .1 Quantum Mechanics Methods

Humanity's journey towards quantum mechanics (QM) started a long time ago with scientists and thinkers speculating about the smallest indivisible units of matter. Over time, atoms were discovered and chemical study began. But it was the discovery of the electron by J.J. Thompson with his cathode ray tube experiment that was the monumental step towards what is known as quantum chemistry today. He discovered not only that the ray consisted of negatively charged particles, but also that they were smaller than atoms.[2] In addition to those observations, he obtained a charge to mass ratio of those "corpuscles" the world would come to know as electrons. Thompson then developed the "plum pudding" model of the electrons where the electrons are dispensed throughout a diffuse positive charge like raisins amidst the namesake British dessert. The idea of diffuse charge central to the "plum pudding" model was then disproved by Ernest Rutherford's famous gold foil experiment (also known as the Geiger–Marsden experiment). The experiment conducted by Rutherford's group involved directing a beam of α particles at an atomically thin gold foil with a detector behind it. The majority of the α particles passed directly through the foil, experiencing essentially no deflection. But the scientists moved their detector and observed some particles deflecting at large angles, even back towards the emitter. The only way those energetic α particles could deflect in such a manner was if the positive charges of the gold atoms were incredibly dense.[3] With this new knowledge, Rutherford posed a different model of the atom,

one where the electrons orbited around a small, dense nucleus of positive charge.

The idea of an atomic nucleus was carried forward as the concept of the atom improved with Niels Bohr's model (presented together with Rutherford). The Bohr model of the atom suggested that the electrons traveled around the nucleus in certain allowed orbits. He was able to explain the hydrogen emission spectrum with this model. Earlier, Johannes Rydberg came up with an empirical formula that, for unknown reasons at that time, fit the emission spectra of hydrogen:[4,5]

$$\frac{1}{\lambda} = R_H\Big(\frac{1}{n_1}^2 - \frac{1}{n_2}^2\Big), for\ n_2 > n_1 \tag{1}$$

where λ is the wavelength associated with the transition, R_H is the the Rydberg constant, and the n values are what Bohr realized are whole numbers representing allowed orbits. The Rydberg Equation (Equation 1) for hydrogen was manifest in the Bohr model and with that model, Bohr was able to calculate and verify the previously empirical Rydberg constant.

Another important milestone in the history of QM was Louis de Broglie's hypothesis of that electrons exhibit both wave and particle-like behavior. This was later extended from just electrons to all matter. The de Broglie equation, relating wavelength and momentum, was born in part from the ideas of Max Planck and Albert Einstein.[6]

$$E_{photon} = h\nu \tag{2}$$

When analyzing the photoelectric effect, Einstein proposed that light existed as packets of discrete energy (later designated photons). Equation 2 expresses the energy of a photon in terms of h, Planck's Constant, and ν, the frequency. It was the previous discoveries of photons from the photoelectric effect that helped guide de Broglie's theory on the wave-particle duality of electrons. The addition of physically motivated boundary conditions to the matter and light waves gave rise to discrete energy levels, and discreteness is otherwise known as quantization. Quantization is where QM diverges from classical mechanics.

The culmination of these experiments, hypotheses, and contributions over the years was

when Erwin Schrödinger posed the Schrödinger equation:

$$\hat{H}\Psi = E\Psi \tag{3}$$

where Ψ is a wavefunction which contains all the information about a system, $\hat{H}$ is the Hamiltonian operator, and E is an energy eigenvalue. Schrödinger realized that the system had to be treated as a wave and devised an operator to measure energy. The time-independent Schrödinger Equation for a 1D system is:

$$\Big(\frac{-\hbar^2}{2m}\frac{d^2}{dx^2} + V(x)\Big)\psi(x) = E\psi(x) \tag{4}$$

In Equation 4, the Hamiltonian operator has a kinetic energy component containing a second derivative and also potential energy represented by $V(x)$. The 1D Schrödinger Equation is a convenient starting point when studying QM. Equation 4 is typically first applied to the (mostly) hypothetical system of a particle in a 1D box with walls of infinite potential and a potential of zero inside the box.[7] From this system, one can see the wave-like character of the particle emerging once the sinusoidal, quantized solutions emerge. From the foundation of the 1D system, the dimensionality can be increased and the Hamiltonian operator and wavefunctions both increase in complexity. However, problems solving the Hamiltonian are encountered once additional electrons are added to the system. Equation 5 is the 3D Hamiltonian operator for a many electron atom (already assuming the Born-Oppenheimer approximation of fixed nuclear positions):

$$\hat{H} = \sum_i^N \frac{-\hbar^2}{2m}\Big(\frac{\partial^2}{\partial x_i^2} + \frac{\partial^2}{\partial y_i^2} + \frac{\partial^2}{\partial z_i^2}\Big) + \sum_i^N \frac{-Ze^2}{4\pi\epsilon_0 r_i} + \sum_i^N \sum_{j>i}^N \frac{e^2}{4\pi\epsilon_0 r_{ij}} \tag{5}$$

where N is the number of electrons, Z is the atomic number r_i is the electron-nuclear distance, and r_{ij} is the electron-electron distance. Because the variables cannot be separated in the electron-electron repulsion term at the end of Equation 5, it causes the Schrödinger

equation to not have an exact solution for systems with more than one electron.

1 .1.1 Hartree Fock

One way to approach the dilemma caused by the electron-electron repulsion term in Equation 5 is with the Hartree Fock (HF) method. HF begins by creating molecular orbital (MO) wavefunctions out of a linear combination of primitive one-electron atomic orbitals:

$$\Phi(\mathbf{r})_i = \sum_{n=j}^{N} c_{ji}\phi(\mathbf{r})_j \tag{6}$$

where N is the number of electrons, $\Phi(\mathbf{r})_i$ is the ith molecular orbital, c_{ji} is the jth coefficient associated with $\phi(\mathbf{r})_j$ the one-electron orbital. Building wavefunctions using atomic basis functions is convenient because variance in electronic character can be expressed by increasing or decreasing the contributions of individual atomic orbitals. The next step in HF is the application of the variational principle to the MOs from Equation 6. A secular determinant emerges from applying the variational principle to the trial MO wavefunction and minimizing the energy:

$$\begin{vmatrix} H_{11} - ES_{11} & H_{12} - ES_{12} & \dots & H_{1N} - ES_{1N} \\ H_{21} - ES_{21} & H_{22} - ES_{22} & \dots & H_{2N} - ES_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ H_{N1} - ES_{N1} & H_{N2} - ES_{N2} & \dots & H_{NN} - ES_{NN} \end{vmatrix} = 0 \tag{7}$$

where H is the Hamiltonian integral, E is the energy, and S is the overlap integral. Because HF uses a time-independent Hamiltonian, the expectation value of the energy of any trial wavefunction will always be an upper bound to the true ground state energy. This allows HF to use a basis set of orbitals to define each MO and minimize the energy with respect to the wavefunction coefficients. In HF methods, the Slater determinant is a convenient means of expressing the many electron wavefunction because it naturally preserves anti-symmetry

of the electrons with respect to exchange:

$$\Psi(\mathbf{x_1}, \mathbf{x_2}, ...\mathbf{x_N}) = \frac{1}{\sqrt{N!}} \begin{vmatrix} \chi_1\mathbf{x_1} & \chi_2\mathbf{x_1} & \cdots & \chi_N\mathbf{x_1} \\ \chi_1\mathbf{x_2} & \chi_2\mathbf{x_2} & \cdots & \chi_N\mathbf{x_2} \\ \vdots & \vdots & \ddots & \vdots \\ \chi_1\mathbf{x_N} & \chi_2\mathbf{x_N} & \cdots & \chi_N\mathbf{x_N} \end{vmatrix} \tag{8}$$

where χ_i is the ith spin orbital and $\mathbf{x_i}$ is the coordinates of electron i.

Even with the usage of a basis set and Slater determinants, there is still the issue encountered in Equation 5 where the electron-electron repulsion term cannot be solved exactly. HF accounts for electron-electron repulsion by first beginning with an assumption of non-interacting electrons in order to create a simpler Hamiltonian. Removing electron-electron interactions allows the Hamiltonian to be written as:

$$\hat{H}_{elec} = \sum_{n=1}^{N} \hat{h}(i) \tag{9}$$

where $\hat{H}_{elec}$ is the electronic Hamiltonian which has become a sum of one-electron Hamiltonians, N is the number of electrons and $\hat{h}(i)$ is a one-electron Hamiltonian defined using atomic units here:

$$\hat{h}(i) = -\frac{1}{2}\nabla_i^2 - \sum_{A=1}^{M} \frac{Z_A}{r_{iA}} \tag{10}$$

where in Equation 10, the first term is the kinetic energy and the second term is the electron-nuclear potential where M is the number of nuclei.

A consequence of Equation 10 is that each individual electron i would have a spin orbital (spatial and spin components) that is an eigenfunction of $\hat{h}(i)$:

$$\hat{h}(i)\chi(\mathbf{x_i}) = \epsilon\chi(\mathbf{x_i}) \tag{11}$$

where $\chi(\mathbf{x_i})$ is the spin orbital and ϵ is the orbital energy. Because each one-electron Hamil-

tonian has an energy eigenvalue, the wavefunction that is a eigenfunction of $\hat{H}_{elec}$ from Equation 9 can be written as:

$$\Psi(\{\mathbf{x_i}\}) = \prod_{i=1}^{N} \chi_i(\mathbf{x_i}) \tag{12}$$

where $\Psi(\{\mathbf{x_i}\})$ is simply a product of N electron spin orbitals referred to as a Hartree product. However, as written in Equation 12, the Hartree product is treating the electrons as distinguishable when they should be treated as indistinguishable. This is addressed by expressing the wavefunction as a Slater determinant like in Equation 8.

With non-interacting wavefunctions established, the next step is to try and reclaim the accuracy lost from $\hat{H}_{elec}$ by removing electron interaction. The Fock approximation uses the non-interacting Hamiltonian as a foundation and includes a correction factor. The one-electron form of the Fock operator is:

$$\hat{F}(i) = \hat{h}(i) + \hat{v}^{HF}(i) \tag{13}$$

where $\hat{F}$ is the Fock operator, $\hat{h}$ is the one-electron "core" Hamiltonian from Equations 10-11, and $\hat{v}^{HF}$ is the mean field operator, each operating on the ith electron. The mean field operator is named such because for a given electron it is a potential created by averaging the contributions of the other $N-1$ electrons. Because $\hat{v}^{HF}$ depends on each other electron (and what their spin orbitals are), $\hat{F}$ cannot simply be solved for one electron. It must be solved for all electrons simultaneously. The $\hat{v}^{HF}$ term in the Fock operator can be expanded as a sum on two-electron operators:

$$\hat{F}(i) = \hat{h}^{core}(i) + \sum_{j}^{N} [2\hat{J}_j(i) - \hat{K}_j(i)] \tag{14}$$

The HF method maintains the separable portion of the Hamiltonian (the first two terms of Equation 5) as $\hat{h}^{core}$ and introduces the $\hat{J}_j$ Coulomb operator and the $\hat{K}_j$ exchange operator. Together $\hat{J}_j$ and $\hat{K}_j$ create the mean field term. In HF the correlation energy is defined as

the difference between the HF energy and the (often unattainable) exact solution.

HF employs the variational theorem in a self-consistent field (SCF) approach using a basis set of spin orbitals and can be represented entirely in a matrix form.[8,9] The wavefunction is represented by a summation of the basis functions each multiplied by a coefficient:

$$\psi_i = \sum_{\mu=1}^{k} c_{\mu i}\chi_\mu \tag{15}$$

where $c_{\mu i}$ is the ith coefficient and χ_μ is the μth basis function out of k total. The energy in terms of the basis is found by solving:

$$\mathbf{FC} = \mathbf{SC}\boldsymbol{\epsilon} \tag{16}$$

With $\mathbf{F}$ as the Fock matrix, $\mathbf{C}$ as the coefficient matrix, $\mathbf{S}$ as the overlap matrix, and $\boldsymbol{\epsilon}$ as the energy. Equation 15 would be the ith column of $\mathbf{C}$ and $\boldsymbol{\epsilon}$ is a diagonal matrix containing the energy eigenvalues. The SCF approach is necessary because $\mathbf{F}$ is needed to determine $\mathbf{C}$ and vice versa. A starting guess is obtained for the density matrix which is used to obtain $\mathbf{F}$ which can be used to obtain $\mathbf{C}$ which is then used to obtain a new density matrix. The iterations continue until a convergence threshold is met and the process is self-consistent.

However, because HF omits the correlation energy, it is limited in its accuracy. After the advent of HF, methods emerged using the HF foundation but employing various techniques to account for HF's missing correlation energy. Determining correlation energy allows these approaches (known as post-HF methods) to achieve much higher energy accuracy than HF. MPn ($n = 2$ is common) adds correction factors up to order n onto the HF energy using perturbation theory.[10,11] Configuration Interaction (CI) is another post-HF method, but CI is a variational method which uses a linear combination of Slater determinants to describe the wavefunction.[12] CI is more accurate than MPn methods and the limit of an complete basis set together with full CI is the exact solution.[13–15]

1 .1.2 Density Functional Theory

A drawback of HF and post-HF methods is how rapidly the complexity scales with increasing system size. HF requires $3N$ spatial and N spin coordinates for an N electron system, so large molecules are computationally costly to evaluate. Density Functional Theory (DFT) offers an alternative solution for determining the wavefunction and energy that only ever (in principle) requires three spatial coordinates.[16] However, hybrid DFT methods (which incorporate parts of HF) do scale the same as HF, but are still useful because of accuracy gains by including electron correlation. Inclusion of electron correlation is another advantage of DFT over HF. The first Hohenberg-Kohn theorem (of which DFT is born) states that the ground state wavefunction of a multi-electron system is uniquely determined by its electron density.[17] This electron density is a function of only three spatial coordinates. The second Hohenberg-Kohn theorem builds upon the first and states that there exists an energy functional (i.e. a mapping from a function to a scalar) that is minimized by the ground state electron density.[18] This second theorem is shown in Equation 17:

$$E[\rho] = F[\rho] + \int \rho(\mathbf{r}) V_{ext}(\mathbf{r}) d\mathbf{r} \tag{17}$$

where $E[\rho]$ is the energy, $F[\rho]$ is the proposed functional, $V_{ext}(\mathbf{r})$ is the external potential (in chemistry, the nuclear potential), and ρ is the electron density. $E[\rho]$ is equal to E_0 when ρ is equal to ρ_0. In other words, $E[\rho]$ can be minimized to approach the true value of the energy. The functional in Equation 17 acts as a sort of correction term for the absent correlation components of the total energy. However, although Hohenberg-Kohn showed that $F[\rho]$ existed and knew some aspects about it, $F[\rho]$ itself was unknown.

The $F[\rho]$ hurdle was then overcome by Kohn and Sham when they found a way to approximate the electron kinetic energy and turn DFT into an actual method. Starting with a proposed system of non-interacting electrons, Kohn and Sham were able to simplify the Hamiltonian to a sum of one-electron operators and express the density using an orbital

basis set.[19] The Kohn-Sham (KS) potential resulting from the KS Hamiltonian shows what the energy components look like after starting from non-interacting electrons:

$$E[\rho] = T_{KS}[\rho] + \int d\mathbf{r} V_{ext}[\mathbf{r}]\rho(\mathbf{r}) + E_{Hartree}[\rho] + E_{xc}[\rho] \tag{18}$$

here $T_{KS}[\rho]$ is the KS non-interacting kinetic energy, the integral term is the external potential, $E_{Hartree}[\rho]$ is the Hartree (Coulomb) energy, and $E_{xc}[\rho]$ is the exchange-correlation energy. However, $T_{KS}[\rho]$ is not purely a function of ρ and depends on the KS orbitals ψ_i as seen in Equation 19:

$$T_{KS}[\rho] = \sum_{n=1}^{N} \int d\mathbf{r} \psi_i^*(\mathbf{r}) \Big(-\frac{\hbar^2}{2m} \nabla^2 \Big) \psi_i(\mathbf{r}) \tag{19}$$

The $E_{xc}[\rho]$ in Equation 18 term also importantly contains correction factors to account for the missing interaction portions of both kinetic energy and potential energy. Interestingly, if orbitals were introduced to $E_{xc}[\rho]$ and $T_{KS}[\rho]$ and if $E_{xc}[\rho]$ was set equal to the exact Hartree exchange, then DFT would reproduce HF. Hybrid functionals such as B3LYP mix in a portion of HF exchange into their DFT exchange to increase their accuracy.[20–23] The electron density is expressed as:

$$\rho(\mathbf{r}) = \sum_{i}^{N} |\psi_i(\mathbf{r})|^2 \tag{20}$$

where $\psi_i(\mathbf{r})$ is a KS orbital. The orbital basis set allows for DFT energy to be solved similar to HF by driving density towards self-consistency.

An important part of using DFT is the choice of functional. One straightforward type of functional was proposed by Kohn and Sham themselves and has become the local density approximation (LDA) class of functionals.[19,24] LDA uses an E_{xc} term that depends only on ρ as a function of $\mathbf{r}$, the "local density". A uniform electron gas is often the foundation of creating the E_{xc} term for LDA. The exchange portion can be exactly obtained for a uniform electron gas, but the correlation portion requires more work. The total energy for various densities of electron gases was obtained by Ceperley and Alder which led to correlation

energy being obtained by subtracting out the exchange energy from the total energy.[22,25] Another functional type is generalized gradient approximation (GGA). GGA functionals include the density and the first derivative and predicts bonds that are not overly stiff like those predicted by LDA.[26] Examples of GGA include PW91, PBE, and BLYP.[21,27–29] Another class of DFT functional are the hybrids. These are not pure DFT functionals, but have a built in contribution of exact exchange from HF. Examples of hybrids include B3LYP and TPSSH.[20,23,30] If building a hierarchy of (ascending/increasing) chemical accuracy for DFT functionals, the LDA class is the bottom tier, then GGA, and the variations of hybrid functionals form the top tier.[31]

Whether using HF or DFT to simulate molecular systems, the first step is to create molecular coordinates. The molecular coordinates contain the nuclear positions which are used to calculate the energy. QM methods like HF and DFT use the electrons (or electron density) to compute the energy according to the current nuclear positions. Molecular mechanics (MM) methods (Section 1 .3) do not account for electrons and instead calculate energy based solely on nuclear positions. MM methods are created by summing calibrated functions of nuclear positions that approach experimental and QM accuracy through parameter tuning. However, both QM and MM simulate molecular dynamics (MD) in a similar fashion, even with drastically different approaches to how energy is obtained. In order to progress an MD simulation through time, nuclear positions and velocities are needed for each atom at each time step. When combined with the time step, the velocities determine how far (and in which direction) the nuclei move. The positions and velocities for each time step are obtained from the current energy of the system after it has been converted to a force. Force can be obtained from the energy of the system by taking the negative derivative of the potential:

$$F = -\nabla V(r) \tag{21}$$

where F is the force and ∇V is the potential. Position and velocity can be obtained from the energy by setting the force in Equation 21 equal to the force from Newton's second law

of motion:

$$F = ma \tag{22}$$

where F is the force, m is the mass, and a is the acceleration. Setting Equations 21 and 22 equal allows the potential energy to directly inform the acceleration, and therefore, the nuclear positions and velocities. There are various integrators that each use a unique approach to update positions and velocites such as Verlet, Velocity Verlet, Predictor-Corrector, and Runge-Kutta.[32,33] The general pattern of integrators is to use position to find the force/potential, the force/potential determines velocity, and the velocity determines a new position to restart the cycle.

1 .2 Exploratory Chemistry

With the increasing availability and cheaper cost of computer hardware and software, computational chemists are no longer required to only run hypothesis-driven simulations. Because computing resources have become abundant, it is possible to engage in exploratory chemistry. Traditionally, chemical analysis begins with a question and then trials and experiments to answer it. Exploratory chemistry is set up by initializing a collection of molecules under reactive conditions, without predisposed hypotheses, and observing the outcome. New, unintuitive reactions can be uncovered or existing reactions can be shown to occur with new pathways. Hypothesis-driven computational chemistry was necessary when resources were more scarce. Simulations had to be carefully selected because the cost of potentially "wasting" resources was high. However, hypotheses/ideas in the form of chemical intuition is still very important in exploratory chemistry when choosing precursor molecules and experimental setup. Otherwise, computing resources could still be wasted if initial setup was not properly thought through. Exploratory chemistry does not require generating exceedingly specific reaction path hypotheses, and can instead map out vast reaction networks including unintuitive reaction pathways. These reaction networks emerging from unguided

experiments are a popular means of generating new chemical questions and answers.

A recent example of exploratory chemistry in action is the *ab Initio* Nanoreactor.[34] The Nanoreactor is a simulation environment that takes collections of molecules or atoms and induces reaction events. The reaction events are encouraged periodically over the course of molecuar dynamics (MD) simulations with a virtual, spherical piston. The piston serves as a means of increasing the energy and inducing collisions.

Reactions are rare events and can be sparse (or even impossible to observe) in the timeframes of molecular simulation. By artificially accelerating collisions and overcoming energy barriers, the Nanoreactor causes a myriad of reactions occur in a short time frame. The MD trajectory is then parsed for reaction events and energy pathways are created between the reactant-product pairs that emerged from the simulation. The proposed reaction pathways can then be vetted for viability by calculating the heights of energy barriers. This process yields unintuitive reactant-product pairs as well as unexpected reaction pathways which can spark new hypotheses.

Catalysis is an area with great exploratory chemistry interest and ongoing efforts. Catalysts provide value to industry by lowering reaction barriers to increase reaction rates, but do not have a universal theoretical approach for computational discovery.[35] Many efforts are being made to mine existing catalyst data and explore the vast candidate space. Ulissi et al. created hydrocarbon surface reaction networks by using DFT to model the likely rate-limiting step candidates and treated the less important steps in the network with surrogate models based on known catalyst trends.[36] That technique allows for rapid and unbiased reaction pathway searching by diverting resources towards more chemically significant steps out of the possible thousands. Another tool for exploring catalyst reaction networks is the Reaction Mechanism Generator (RMG).[37] RMG is an open source method that uses graphs and trees to connect molecules to each other and build reaction networks. The recent 3.0 version update of RMG now includes capability to map out heterogeneous catalyst models.[38,39]

Another innovation for autonomous reaction network exploration is Chemoton (currently

version 2.0).[40,41] Chemoton uses multiple low-dimensional potential energy surfaces to quickly parse elementary reactions and graph a reaction network. Global reaction route mapping (GRRM) is another exploratory chemistry approach which uses QM to automatically explore isomers, synthetic routes, and dissociation channels.[42,43] ZStruct uses a growing string method to form a reaction path by adding nodes/structures until encountering a reactant and product pair separated by a transition state.[44,45] In addition to those mentioned above, there are many other exploratory chemistry techniques in use and being actively developed.

1 .2.1 Data Processing

As exploratory chemistry methods refine over time and computational costs continue to drop, unguided simulations are becoming more popular. However, one significant hurdle when simulating exploratory chemistry is the processing the vast amounts of data that are generated. The uncertain outcomes and variety of candidate starting molecular ensembles lead to many long simulations. It is expected that reactions are occurring throughout these simulations, and so a major issue is knowing how to process all that data. One dilemma is how to even define a reaction event. Scientists have to find clever ways to impart their chemical intuition into automating the data processing steps. Systematic definitions of reacted and unreacted molecules are required.

There are many ongoing efforts in exploratory chemistry involving enhanced rare event sampling and/or reaction-dense data processing.[46] One recent example is the CARNOT software. CARNOT simultaneously simulates combustion reactions and can separate out interesting fragments and search for plausible pathways without bias.[47] Another example is the AutoMeKin (AMK) package. AMK exists in many variants, and among other things, it can automatically screen for transition state candidates using MD to generate guesses.[48,49] Rice et al. took on the challenge of identifying reaction events buried in MD simulation data with a combination of bond length and vibrational metrics.[50] The work in chapter 2 joins

these efforts and attempts to address the challenge of defining and detecting reaction events within simulations.

1 .3 Molecular Mechanics Methods

With infinite time and resources, the optimal choice for computational chemistry studies would be to use the best QM software available. However, when actually optimizing method choice for accuracy and cost, compromises must be made. Molecular Mechanics (MM) force fields are a low cost alternative to QM that for many use cases does not sacrifice too much accuracy. Not only accuracy is sacrificed, but MM methods cannot simulate chemical reactions (specifically, changes in atom connectivity). However, certain force fields such as ReaxFF have introduced reactivity into MM. ReaxFF uses atomic distances to calculate bond orders and uses bond orders to introduce connection-dependant terms into their force fields to allow implicit electronic bonding interactions.[51–53] Despite these and other advances, MM methods fundamentally lack the detail of QM, but are useful because of their significant cost and scaling advantages. The ability of MM to deal with large, bio-molecule scale simulations means MM play a large role in the computational chemistry landscape. Once systems grow to the size of many atoms and many more electrons MM is often used instead of QM. There exist some hybrid MM/QM approaches that use MM for the majority of the system and QM for the specific small region of interest, but that is not the focus of the projects herein.[54]

Where QM differs from MM is how the potential energy is obtained. QM obtains energy by solving an electronic problem with HF, post-HF methods, or DFT whereas MM directly obtains it from nuclear coordinates. In MM force fields, molecules are approximated with ball and stick models and energy is computed more quickly and less costly by approximating interactions with simple expressions. By starting from simple expressions, a force field builder can refine the approximations by adding more terms or improving the equations representing terms. MM force fields components can be separated into two major categories: bonding and non-bonded.

1 .3.1 A Simple Force Field

A simple force field can be built by including a number of bonding and non-bonded components such as the ones listed in Equation 23.

$$E_{total} = E_{bond} + E_{angle} + E_{torsion} + E_{electrostatic} + E_{vdw} \tag{23}$$

The bond term of a simple force field can be created by capturing two-body bond-stretching interactions with a simple Hooke's law expression as seen in Equation 24:

$$E_{bond} = \frac{1}{2} k_b (r - r_0) \tag{24}$$

where k_b is a spring constant unique to the element pair, r_0 is the equilibrium bond length, and r is the measured bond length. One way to approximate the three-body angle-bending interactions is with a harmonic potential as seen in Equation 25:

$$E_{angle} = \frac{1}{2} k_a (\theta - \theta_0)^2 \tag{25}$$

Where k_a is a force constant, θ_0 is the equilibrium angle, and θ is the measured angle. Four-body torsions (also known as dihedrals) can be approximated with a cosine series expansion as shown in Equation 26:[55]

$$E_{torsion}^{ABCD}(\omega) = \sum_{n=1} V_n \cos(n\omega) \tag{26}$$

where for connected atoms ABCD, V_n is a force constant, n is the multiplicity, and ω is the torsion angle.Torsions measure the strain/interaction between four atoms, three bonds apart using an angle formed by looking down the B-C atom internuclear axis and measuring the angle the A-B and C-D bonds make with respect to each other.

Each term in a force field has adjustable parameters which need to be selected and/or tuned for each of the elements involved. Five-body and higher order terms are not always in-

cluded because of diminishing returns regarding energy accuracy. Higher order terms increase the cost typically without a profound impact on the accuracy, though some CHARMM force fields have found value in protein structure simulations by including five-body terms.[56–58] Despite not including higher order terms, force fields can be made more accurate through comparison to trusted data. Scientists tune force field parameters to fit QM and experimental data such as energy, density, vibrational frequency, and many more metrics.[59] Well tuned force fields can accurately reflect the desired behavior of molecules.

The first of the non-bonded terms of a simple force field mimics van der Waals (VDW) interactions. VDW interactions can be described in a force field with the Lennard Jones potential which is shown in Equation 27:

$$E_{vdw} = 4\epsilon\Big[\Big(\frac{\sigma}{r}\Big)^{12} - \Big(\frac{\sigma}{r}\Big)^{6}\Big] \tag{27}$$

where r is the distance between atoms, ϵ is the well depth, and σ is a separation term which controls the distance between $E = 0$ and the exponential spike. The Lennard Jones potential very cleanly and simply mimics the non-bonded potential well of two atoms approaching each other. Short range repulsion emerges when r approaches zero and the energy goes to zero as r approaches infinity. The second simple non-bonded term involving electrostatic interactions can be approximated by with Coulomb's Law shown in Equation 28:

$$E_{electrostatic} = \frac{q_i q_j}{4\pi\epsilon_0 r_{ij}} \tag{28}$$

where the q terms represent charges in space and r_{ij} represents the distance separating them. As with the bonded terms, both force field components and their accompanying parameters can be further refined to better reproduce the desired chemical attributes of both QM and experimental studies. However, even simple force fields can capture relevant chemical behaviors.

Some force fields include terms to account for molecular polarizability. Depending on

the intended application, many force fields exclude these terms. However, simulation of certain molecular behaviors can be improved by including polarizability terms. Warshel et al. showed this by comparing cation-valinomycin binding energies in polariazable and non-polarizable force fields.[60] They found that while polarizable force fields better replicated experimental binding energies of monovalent cations, nonpolarizable force fields were not too far off. However, when they analyzed binding energies of divalent cations, polarizable force fields were extremely close to experiments and nonpolarizable were multiple kcal/mol incorrect for each ion. Even the relative binding energies in the nonpolarizable case were incorrect. On the other hand, Kamenik et al. compared protein folding in polarizable and non-polarizable force fields and found that their choice of polarizable force field actually under-performed their non-polarizable choices.[61] However, they were only comparing one polarizable force field to three non-polarizable force fields, and that polarizable force field has since had that specific issue corrected in a later update.[62] It is expected that introducing polarizability to force fields improves their non-bonding interactions and therefore their overall accuracy, but it cannot be assumed. It is worth implementing polarizable force fields and measuring them against experimental data and non-polarizable force fields to see if the additional computational cost delivers measurably better results.

1 .4 Deep Eutectic Solvents

Deep Eutectic Solvents (DESs) are a relatively new field with exciting potential.[63] A eutectic mixture is created upon mixing two molecules in the correct whole-number ratio such that the resulting mixture has a lower melting point than either individual component. In other words, two solids can be mixed to create a liquid mixture. Eutectic mixtures are achieved by using Lewis or Brønsted-Lowry acids and bases to create hydrogen bonding intermolecular interactions.[64] DESs are often compared to Ionic Liquids (ILs) since they are both partial charge-rich solvents. Formation of DESs is different from ILs which are simply anion-cation pairs attracted by Columb potential.[65] However, DESs and ILs occupy a similar usage space

even though are fundamentally different.

The prevalence of strong partial charges helps give DESs character similar to that of ILs. ILs have low vapor pressure, an ion-rich conductive environment, and high chemical and thermal stability.[66–68] They have incredible variety due to the fact that their properties can be tuned with ion selection or mixing with other ionic liquids. ILs are differentiated from molten salts by the fact that they have been designed to be liquid at temperatures less than 100°C.[69] These lower melting points are achieved by pairing large cations with large anions which greatly reduces the Coulomb interaction. ILs play a role in solvating industrial chemical processes such as manufacturing of batteries, superconductors, and solar cells. However, some classes of ILs (like those that contain halogens) are not ideal solvents because of the risk they pose to the environment.[65] DESs are a promising alternative to ILs because they are inexpensive to create and less toxic to the environment.[70,71]

DESs can be created by mixing a quaternary ionic compound with a metal salt of a hydrogen bond donor.[72] Choline chloride is a popular quaternary ionic compound for building DESs. DESs have vast variety and tunability because of the numerous candidate hydrogen bond donors. Simple changes to hydrogen bond donor size and chemical character can change the viscosity, density and numerous other attributes to favor desired use cases.

DESs are important to study computationally, not only because of their intrinsic value as industrial solvents, but because they are tunable. Greater understanding of their properties and behavior can help elucidate undiscovered DES combinations with better features. DESs are used in organic and inorganic extraction, formation of gels, metal processing and plating, synthesis, and more.[64,65,73] There are active studies both experimentally and computationally into how to utilize and create new DESs. Chapter 3 showcases a method for semi-automated initialization of polarizable force fields with DES ingredient urea.

2 Bond-Order Time Series Analysis for Detecting Reaction Events in *Ab Initio* Molecular Dynamics Simulations[1]

2 .1 Abstract

Ab initio molecular dynamics is able to predict novel reaction mechanisms by directly observing the individual reaction events that occur in simulation trajectories. In this article, we describe an approach for detecting reaction events from simulation trajectories using a physically motivated model based on time series analysis of *ab initio* bond orders. We found that applying a threshold to the bond order was insufficient for accurate detection, whereas peak finding on the first time derivative resulted in significantly improved accuracy. The model is trained on a reference set of reaction events representing the ideal result given unlimited computing resources. Our study includes two model systems: a heptanylium carbocation that undergoes hydride shifts, and an unsaturated iron carbonyl cluster that features CO ligand migration and bridging behavior. The results indicate a high level of promise for this analysis approach to be used in mechanistic analysis of reactive AIMD simulations more generally.

2 .2 Introduction

A central goal of theoretical chemistry is to provide sufficient insight into reactivity at the molecular scale to inform the design of experiments including reaction routes, reaction conditions, and catalysis.[74–76] Computational studies of reaction mechanisms often start by hypothesizing a reaction pathway from chemical intuition, followed by calculating the minimum energy path and associated critical points (reactant, product, and transition state structures) with local optimization methods.[77,78] The reaction rate associated with a pathway may be estimated from the activation energy using kinetic models, enabling a semi-quantitative

comparison with experiment.[79,80] The main drawback of this strategy is that only existing hypotheses can be tested, and such hypotheses traditionally originate from chemical intuition; in other words, the systematic generation of mechanistic hypotheses is an important challenge for theoretical chemistry. Another aspect of this challenge is that many reaction mechanisms proceed through multiple elementary steps and short-lived intermediates that are difficult to experimentally characterize.

Recently, computational methods have been developed that automate the searching procedure by systematically applying basic rules to break and form chemical bonds in a combinatorial fashion.[37,44,81–86] These methods, which are based on assuming general rules of reactivity rather than specific mechanistic hypotheses, can greatly increase the automation in mechanistic studies and have proven successful in applications.[87–89] However, there are still limitations to such approaches because they require assuming the basic rules of reactivity, which are not fully understood; moreover, the relative positioning of reactants in multi-molecular or roaming reaction pathways continues to be a challenge for rules-based approaches.

In the past few years, *ab initio* molecular dynamics (AIMD) has emerged as a useful tool for the discovery of reaction mechanisms. In fact, classical molecular mechanics (MM) simulations have long been used to discover pathways of protein folding and conformational change;[90–94] these involve changes in the protein backbone and side chain conformations as well as intermolecular interactions, which do not require a quantum mechanical description. Consequently, the PES can be approximated using inexpensive force fields, allowing MM simulations to routinely reach microsecond time scales and beyond. On the other hand, predictive sampling of a reactive system usually requires a quantum mechanical calculation of the electronic wavefunction at every time step, which costs at least four orders of magnitude more than evaluating a MM force field and usually scales less favorably with system size. More recently, modern advances in electronic structure methods and accelerated hardware implementations have resulted in speed-ups of 2-3 orders of magnitude for

Hartree-Fock and density functional theory (DFT) calculations,[95–106] placing AIMD simulations on the threshold of discovering reaction mechanisms that occur on nanosecond or longer timescales.[34,107–111]

Because reaction rates are exponentially decreasing functions of the activation barrier, it is still highly challenging to map the chemically interesting reaction pathways in an unbiased AIMD simulation. Recently, we and others have introduced specialized AIMD simulation methods for accelerating the discovery of reaction pathways. The Pietrucci group introduced topological-based permutation invariant "SPRINT" coordinates helped to address the isomer degeneracy problem in metadynamics.[112] The Pfaendtner group demonstrated how to reduce computational cost and the need to manually specify reaction coordinates by using parallel bias metadynamics using SPRINT coordinates as collective variables.[113] The *ab initio* nanoreactor causes a large number of reactions to occur in a relatively short simulation by periodically forcing the molecules in the simulation to undergo high-velocity collisions.[34] Because the nanoreactor requires no specification of reaction coordinate, it is able to discover new pathways for interesting reactions such as the prebiotic synthesis of glycine and sugars.[34,114,115] As these simulations do not involve specifying reaction coordinates or desired products, an automatic approach is needed to identify the potentially interesting reaction events.

The recent emergence of AIMD simulations containing large numbers of reaction events requires new theoretical tools for deriving useful knowledge from them. One of the principal tasks is to identify the discrete transition between chemical structures from the continuous variables of the simulation trajectory. We recently introduced a procedure for detecting and extracting reaction events based on analysis of interatomic distances, followed by a series of optimization calculations to locate the minimum energy path associated with the observed reaction.[116] In a related work, Döntgen and coworkers developed an analysis approach for reactive MD trajectories simulated using the ReaxFF force field, where the ReaxFF bond order was used to detect reaction events and calculate reaction rates directly from the observed

events.[117] Both studies noted that some ambiguity remains in reaction event detection, as a number of empirical parameters (including covalent radii, ReaxFF parameters, and lag times) were used to determine the threshold for what constituted a genuine reaction event in the simulation. As these exploratory-type simulations are destined to become increasingly important in simulation studies of reaction mechanisms, a greater amount of rigor and precision is clearly needed in the identification of the reaction events. Motivated by this need, we would like to address the following questions: How can we properly define a reaction event in an AIMD simulation? How can we systematically improve on reaction event detection methods?

In this paper, we address these questions by introducing a new reaction event detection method based on time series analysis of the AIMD trajectory. To develop this method, a suitable set of reference reaction events is created by local energy minimization of each structure on a reactive trajectory. The sequence of optimized structures is clustered into a discrete number of chemical states, and the transitions in the sequence of states are used as a reference dataset for the time series analysis. Our reaction detection approach is based on the *ab initio* bond order index defined by Mayer.[118] Our results show that the time series analysis based on bond order indices is able to reproduce the reference data set accurately using few parameters. An iron carbonyl cluster ($Fe_3(CO)_9$) previously studied theoretically by Schaefer and coworkers[119] and a heptanylium cation ($C_7H_{15}{}^+$) are used as a testing ground for this method; our results indicate the AIMD simulation method is able to discover a significant number of new local minima connected by low energy barriers at a far lower cost than optimizing entire trajectories. Our methods and results provide a foundation for more chemically relevant understanding of reactive AIMD trajectories.

2 .3 Theory and Methods

Here we briefly summarize the main considerations in the development of our reaction detection model before describing individual aspects in more detail. This paper focuses

on reactions that involve rearrangements of bonding within a single molecule, though we think making generalizations to reactivity involving several molecules should be conceptually straightforward. Because our reaction events involve making and/or breaking chemical bonds, we intuitively expect the atom pair-wise bond orders (BO) will increase or decrease when bonds are formed, broken, or undergo changes in electronic character. Thus, our model will use the BO time series between all atom pairs as input data and detect reaction events from changes in the time series. We use the *ab initio* bond order defined by Mayer as:

$$M_{ab}[i] = 2 \sum_{\mu \in a} \sum_{\nu \in b} \left[(\mathbf{P}^{\alpha}\mathbf{S})_{\mu\nu} (\mathbf{P}^{\alpha}\mathbf{S})_{\nu\mu} + (\mathbf{P}^{\beta}\mathbf{S})_{\mu\nu} (\mathbf{P}^{\beta}\mathbf{S})_{\nu\mu} \right] [i] \tag{29}$$

where $\mathbf{P}^{\alpha,\beta}$ is the one-particle density matrices for alpha and beta spin, $\mathbf{S}$ is the overlap matrix, μ, ν are indices for atomic basis functions, the sums are restricted to functions centered on atom indices a, b, and $[i]$ indicates values at frame i in the simulation trajectory. Thus, the bond order is defined as a discrete series spaced in time by the simulation time step δ.

In the context of our work, "detection" refers to estimating or predicting the approximate location of a reaction event. This definition requires introducing a set of reference reaction events that represents the desired result given unlimited computing resources. A reference reaction event is defined when the *ab initio* molecular dynamics trajectory crosses between two catchments (energy basins) in configuration space that contain chemically different local minima. Energy basins are separated by manifolds of local maxima (dividing surfaces), and we assume the minima are located in the interior of the catchment and away from dividing surfaces, such that chemically different energy-minimized species will differ significantly in their structures and BO matrices. Therefore, if we could carry out energy minimization of every trajectory frame, the chemically distinct species and reaction events could be precisely located by comparing the BO matrices of energy-minimized structures (Figure 1).

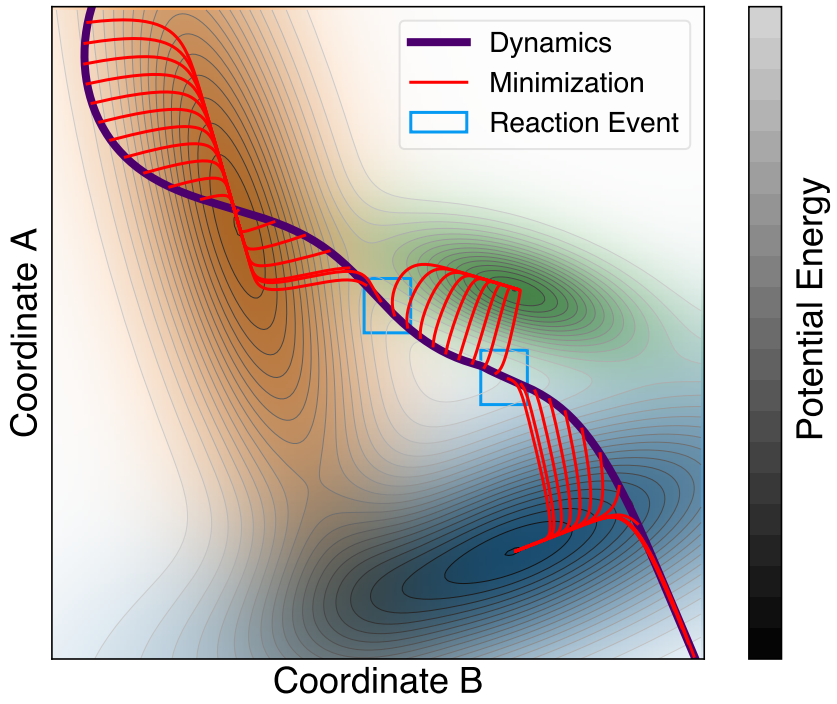

Figure 1: Example MD trajectory (violet) with the optimization pathways for the discrete time steps shown in red. Light blue boxes highlight where the trajectory crosses subdomains in configuration space. These crossings are defined as reaction events.

The reference method is too computationally costly for routine applications because energy minimization of every trajectory frame is significantly more expensive than the AIMD simulation itself. Here, we have computed the reference reaction events to train the model parameters for our two systems, the iron carbonyl cluster $Fe_3(CO)_9$ and heptanylium cation $C_7H_{15}^+$; these systems have major differences in terms of their composition, bonding and coordination. By applying our method to both systems and comparing the results, we characterize the parameter sensitivity of the reaction detection model and provide some guidelines for when it is necessary to compute the reference reaction events for a system of interest.

2 .3.1 Computational Details

To generate a set of reference reaction events and bond order time series for both systems we used unbiased, temperature-accelerated *ab initio* molecular dynamics simulations.[120,121] For both systems we used a Velocity Verlet integrator with a timestep of 1 fs and a Langevin thermostat with an equilibrium temperature of 1000 K and a damping time of 1 ps^{-1}. We simulated the iron carbonyl cluster using the BP86 density functional approximation together with a double-ζ plus polarization (DZP) all-electron basis for all atoms including iron, following Ref.[119] The molecular dynamics simulation was propagated for a duration of 8,373 steps before terminating with a SCF convergence error. The heptanylium simulation used the B3LYP density functional and a 6-31G* basis set, and the simulation was propagated for 10,000 steps. To create the reference reaction sets, every AIMD frame was used as the input coordinates for energy minimization at the same level of theory.[122] All of the simulations in this study were carried out using the TeraChem quantum chemistry software package.[95–97]

2 .4 Details of the model systems

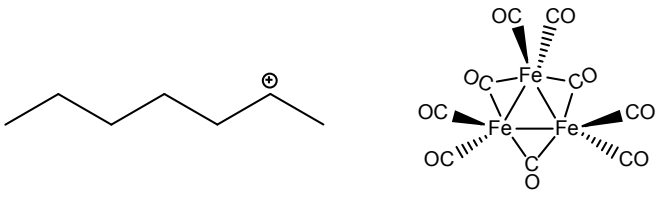

Figure 2: Starting structures of the two systems; heptanylium cation $C_7H_{15}{}^+$ (left) and iron carbonyl cluster $Fe_3(CO)_9$(right).

We chose two model systems to characterize the accuracy of our reaction detection model in this study. The first system with simpler and more straightforward reactivity is the heptanylium alkyl carbocation, $C_7H_{15}{}^+$. We expected the reaction events for the heptanylium system to manifest as hydride and methyl shifts. This system was chosen so that generating a

tertiary carbocation was unlikely under our simulation conditions, as this would halt further reactions because of their relative stability.

The iron carbonyl cluster we chose to study is $Fe_3(CO)_9$ which is the smallest in a series of four clusters with increasing carbonyl count studied by Schaefer and coworkers.[119] Because $Fe_3(CO)_9$ is unsaturated, this system presents interesting possibilities for CO ligand migration and bridging multiple Fe atoms. These two systems were chosen to be chemically distinct in order to illustrate the performance of the model when used in diverse applications.

2 .4.1 Reference reaction events

We assume that the potential energy surface is divided into catchments or energy basins denoted as $\mathcal{S}_k$ in the regions of the potential energy surface accessed by the AIMD simulation, where the index k represents all such basins that are sampled by one simulation. These are bounded regions on the potential energy surface where each point in the region is mapped by energy minimization to a local minimum somewhere in the interior as $\mathbf{y}_k = \mathit{Optimize}\,(\mathbf{x} \in \mathcal{S}_k)$. Moreover, because we are interested in detecting reactivity, catchments that correspond to chemically identical species and share any boundaries are grouped together. Our task consists of finding the catchments that are visited by the AIMD trajectory frames and identifying when the trajectory crosses over their dividing surfaces (i.e. reaction events).

We expect that two local minima in different energy basins $(\mathbf{y}_k, \mathbf{y}_l)$ with major differences in chemical bonding should be distinguishable by comparing their BO matrices. Thus, constructing the reference reaction events from an AIMD trajectory follows this procedure:

1. Calculate a series of optimized structures by local energy minimization of every frame in the simulation trajectory, i.e. $\mathbf{y}[i] = \mathit{Optimize}(\mathbf{x}[i])$.

2. Cluster the series of optimized structures using a chosen distance metric and clustering algorithm. This produces a set of clusters $\{C_k, 1 \leq k \leq N_C\}$ where each trajectory frame belongs to only one cluster and N_C is the number of clusters. Each cluster k

corresponds to a distinct catchment $\mathcal{S}_k$ and a representative optimized structure $\mathbf{y}_k$. The cardinality of the cluster is represented as $|C_k|$.

3. Assign each optimized structure to a cluster to produce a series of cluster numbers $\{K[i], 1 \leq i \leq N_{\text{steps}}\}$.

4. The time coordinates of reference reaction events are where the cluster number of the optimized structure differs between two consecutive frames as:

$$\mathcal{E}_{\text{ref}} = \{i \mid K[i] \neq K[i+1]\} \tag{30}$$

For two energy-minimized structures, we compute the bond-order distance metric (BODM) as the L_2 norm of the difference in bond order matrices:

$$d[i,j] = \sqrt{\sum_{a<b}^{N_{\text{atom}}} \left(\widetilde{M}_{ab}[i] - \widetilde{M}_{ab}[j]\right)^2} \tag{31}$$

where the tilde over $\widetilde{M}$ indicates that the BO matrix of the energy-minimized structure is used. This idea is similar to, and indeed inspired by, the featurization of biomolecular simulation trajectories such as contact maps, dihedral angles, and metrics such as RMSD which are used in the construction of kinetic models.[123,124] Our choice of using BO matrices is an important distinguishing factor from earlier work, and justified because the BODM directly measures changes in chemical bonding and should exclude other conformational changes. (*Remark:* Two structures that differ only by the permutation of atomic indices may also have significant BODMs, e.g. isotope exchange. This does not significantly affect our main conclusions, but including permutation invariant reactions as predictions may add additional computational cost in post-processing.)

To cluster the optimized structures into discrete chemical species, we used a hierarchical clustering algorithm with an average-linkage criterion as implemented in the SciPy pack-

age,[125,126] where the input data is a matrix of the BODM values above. At the start of the algorithm, each structure is assigned to a separate cluster. The pair of clusters with the smallest distance is merged into a single cluster, and the new clusters are renumbered in consecutive ascending order. The distance between the newly merged cluster C_k and all other clusters C_l is defined as:

$$D_{kl} \equiv \frac{1}{|C_k||C_l|} \sum_{i \in C_k,\, j \in C_l} d[i,j] \tag{32}$$

The merging procedure is repeated until the smallest pairwise distance is larger than a threshold parameter, resulting in the final set of clusters $\{C_k\}$. We chose a clustering threshold parameter of 1.0 because it represents a difference of approximately one bond between clusters, and because the number and contents of clusters was consistent with our chemical intuition and visual examination of the optimized structures. A dendrogram showing the successive merging of clusters as a function of the threshold parameter for each system is given in Supporting Figures S1 and S2.

A sequence of cluster indices is obtained for the trajectory of optimized frames. For each frame where the cluster number differs between the current and next frame, a reference reaction event is defined. Each reference reaction event is a data structure containing the current frame number, as well as the optimized structures and BO matrices of the current and next frame. The BO matrices allow us to query which bonds were formed or broken in the reaction event, which will become important in §2 .4.3.

2 .4.2 Reaction detection by time series analysis

Here we describe efficient and approximate models for estimating the reaction events via direct analysis of the AIMD BO trajectory data. The purpose of these models is to reduce the number of computationally costly energy minimizations needed to find the reaction events in the simulation. In our current context where the entire system consists of a single

molecule, the model predicts which time coordinates (i.e. frame numbers) are likely to be near true reaction events, thereby restricting the energy minimizations to within small time windows of these predicted frames. A high-quality model should be sensitive enough to correctly detect most or all of the reaction events, while ruling out "unreactive" parts of the simulation trajectory to reduce computational cost.

For a particular atom pair with indices a and b, the bond order time series $\{M_{ab}\} = \{M_{ab}[i]; 1 \leq i \leq N_{\text{steps}}\}$ is a discrete sampling of the bond order as a function of time. Because variations in the bond orders are slow compared to the time step, we assume aliasing effects from discrete sampling are negligible. $\{M_{ab}\}$ contains both long-lasting changes that represent genuine reaction events and changes in chemical bonding, as well as higher-frequency fluctuations that we are less interested in. Thus, we process $\{M_{ab}\}$ with a low-pass filter to remove the fast fluctuations and retain the chemically important features of the time series:

$$\{\overline{M}_{ab}(\sigma)\} = L\Big(\{M_{ab}\},\ \sigma\Big) \tag{33}$$

Here, L is the function that performs the low-pass filtering (we used a sixth-order Butterworth filter), the line over $\overline{M}$ indicates that the time series has been smoothed, and σ represents the cutoff frequency parameter. σ can be optimized in order to produce the best agreement between the detected reaction events and the reference set. One of our goals in this paper is to show that the performance of this method is not highly sensitive to the choice of σ for different applications.

2 .4.2.1 Thresholding on time series values One intuitive approach to predicting reaction events is to detect when the smoothed time series crosses over a threshold that separates bonded from non-bonded regimes. This approach is similar to our previous study[116] where connectivity between atom pairs was defined by comparing interatomic distances to a threshold derived from covalent radii. One advantage of using bond orders is that the

highly sensitive element-wise radius parameters are no longer needed. Here we will show that applying a threshold to the bond order is insufficient for detecting reactions, which motivates the time derivative approach in § 2 .4.2.2.

Equation 34 is the set of predicted reaction events for atom pair (a, b) where $\overline{M}_{ab}$ crosses a threshold μ:

$$\mathcal{E}_{0;ab}(\sigma, \mu) = \left\{ i \;\middle|\; \overline{M}_{ab}(\sigma)[i] > \mu > \overline{M}_{ab}(\sigma)[i+1] \vee \overline{M}_{ab}(\sigma)[i] < \mu < \overline{M}_{ab}(\sigma)[i+1] \right\} \quad (34)$$

The set of predicted reaction events for the entire system is found by taking the union over all atom pairs:

$$\mathcal{E}_0 \;=\; \bigcup_{b>a=1}^{N_{\text{atoms}}} \mathcal{E}_{0;ab} \quad (35)$$

However, we found that the reaction events identified in Equation 35 were incomplete, as many reaction events could not be accurately predicted using a single threshold in the iron carbonyl simulation.

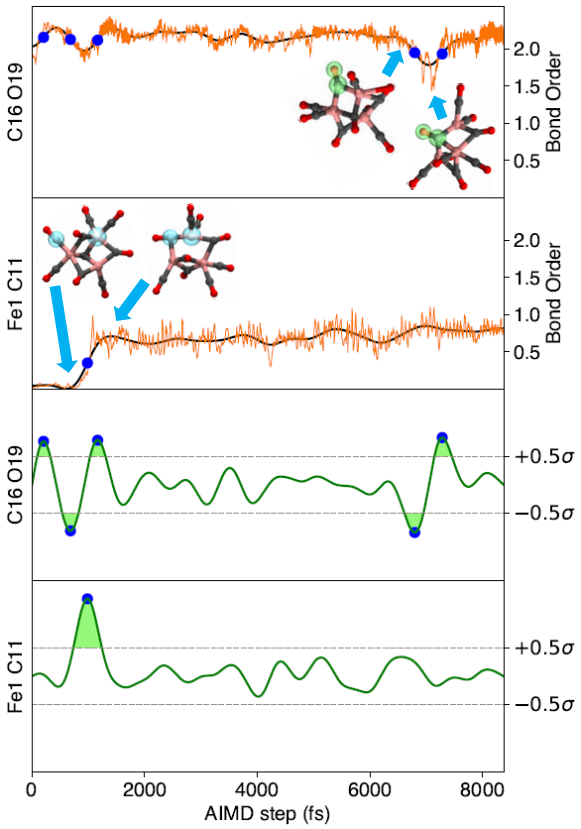

Figure 3: Top two panels: Raw (M_{ab}, orange) and 40 cm^{-1} low-pass filtered ($\overline{M}_{ab}$, black) time series of two selected bond orders. Bottom two panels: First time derivative of filtered series ($\overline{M}'_{ab}$, green curve) with threshold $\mu = \pm 0.5\sigma$ (dashed lines). $\text{Intvl}_{+;ab}$ and $\text{Intvl}_{-;ab}$ shown with light green shading. Detected reaction events $\mathcal{E}_{1;ab}$ from the bottom two panels are shown as blue dots across all panels. Molecule color scheme: iron: pink, carbon: gray, and oxygen: red.

Figure 3 shows why applying a single threshold to the smoothed bond order time series to detect reaction events can be challenging. In the upper panel showing the Fe-C bond order there is a distinct increase from 0.0 to 0.9 near $t = 1000$ fs, indicating that a threshold parameter of $0.1 - 0.8$ would work well for this atom pair. However, the upper C-O panel contains fluctuations in the bond order near $t = 7000$ fs that are indicative of changes in the carbonyl ligand coordination to Fe, where the value of the bond order is consistently in the $1.9 - 2.3$ range. In order to detect any reaction events in the C-O time series, the threshold would need to be much higher, around 2.0. If we were forced to use different thresholds

for different kinds of bonds, then it would detract from the usefulness of the *ab initio* bond order as a simple criterion for detecting reactivity.

Another drawback of applying a threshold directly to $\{\overline{M}_{ab}(\sigma)\}$ is the risk of detecting large numbers of false positives and false negatives. If the thresholds were chosen close to the mean value of an oscillating BO time series, repeated crossings over the threshold could cause many false positives. Although the number of oscillations may be reduced by increasing the smoothing, it does not address the fundamental problem that the threshold parameter is close to the mean value of the oscillation. In both upper panels of Figure 3, there exist ranges of the threshold parameter that would contain many crossings due to oscillations around an apparent mean. The risk of excessive false positives due to repeated threshold crossings and false negatives due to missed crossings indicates that if we applied a threshold to the bond order to detect reaction events, the results would be highly sensitive to parameter choice, which negatively affects the utility of the method. In what follows, we show that applying a similar thresholding approach to the bond order *time derivative* is a simple way to address many of these issues.

2 .4.2.2 Peak finding on first time derivative Once the raw time series has been filtered to remove high-frequency components (Equation 33), the first time derivative of the smoothed time series is taken:

$$\overline{M}'_{ab}(\sigma)[i] \equiv \frac{d}{dt}\Big(\overline{M}_{ab}(\sigma)[i]\Big) \approx \frac{M_{ab}(\sigma)[i+1] - M_{ab}(\sigma)[i-1]}{2\delta} \tag{36}$$

The prime on $\overline{M}'_{ab}$ indicates the first time derivative, approximated via central difference on the discrete values of $\overline{M}_{ab}$. Next, a threshold (μ) is applied to $\overline{M}'_{ab}(\sigma)$:

$$\text{Intvl}_{+;ab} \equiv \Big\{(u,v) \;\Big|\; \overline{M}'_{ab}(\sigma)[t] > \mu \;\forall\; t \in (u,v) \wedge \overline{M}'_{ab}(\sigma)[u] \leq \mu \wedge \overline{M}'_{ab}(\sigma)[v] \leq \mu\Big\} \tag{37}$$

$$\text{Intvl}_{-;ab} \equiv \Big\{(u,v) \;\Big|\; \overline{M}'_{ab}(\sigma)[t] < -\mu \;\forall\; t \in (u,v) \wedge \overline{M}'_{ab}(\sigma)[u] \geq -\mu \wedge \overline{M}'_{ab}(\sigma)[v] \geq -\mu\Big\} \tag{38}$$

Here, $\text{Intvl}_{+;ab}$ and $\text{Intvl}_{-;ab}$ are sets of continuous time intervals for which $\{\overline{M}'_{ab}(\sigma)\}$ is above $+\mu$ and below $-\mu$ respectively. We then collect the time-coordinates of the positive maxima of $\{\overline{M}'_{ab}(\sigma)\}$ above $+\mu$ and the negative minima below $-\mu$ as:

$$\begin{aligned}\mathcal{E}_{1;ab}(\sigma,\mu) \equiv & \left\{ t \;\middle|\; \underset{t\in(u,v)}{\arg\max}\; M'_{ab}(\sigma)[t] \;\forall\; (u,v) \in \text{Intvl}_{+;ab} \right\} \\ & \bigcup \left\{ t \;\middle|\; \underset{t\in(u,v)}{\arg\min}\; M'_{ab}(\sigma)[t] \;\forall\; (u,v) \in \text{Intvl}_{-;ab} \right\}\end{aligned} \tag{39}$$

As a result, we obtain $\mathcal{E}_{1;ab}(\sigma,\mu)$ as the final set of reaction events derived from the BO time derivative for atom pair ab. The smoothing, derivative, and thresholding steps are illustrated in Figure 3. The time-coordinate of every blue dot (identified in the bottom two panels) represents the set of detected reaction events $\mathcal{E}_{1;ab}(\sigma,\mu)$ for the given atom pair ab. Figure 3 shows the advantage of using BO time derivatives instead of applying a threshold directly on the BO values, because the derivative approach can detect reaction events from both the Fe-C and C-O time series whereas the same direct threshold cannot be used for both atom pairs. The fundamental assumption of this approach is that there are no reaction events that change the BO time series very slowly, as that would not be detected by the threshold. We expect this assumption to be generally valid due to the relatively short distance ranges over which chemical bonds are broken and formed, and the atomistic forces along the reaction pathway would prevent bond orders from changing very slowly.

Comparisons to the reference set of reaction events can be made in two ways: either on an "bond-wise" basis, or on an "unified" basis where we take the union over all atom pairs. Thus, the unified set of reaction events is taken by collecting all reaction events for all atom pairs in the system as:

$$\mathcal{E}_1 \;=\; \bigcup_{b>a=1}^{N_{\text{atoms}}} \mathcal{E}_{1;ab} \tag{40}$$

2 .4.3 Receiver operating characteristic objective function

If our model were perfectly accurate, then for every reaction event predicted, the pair of structures preceding and following the event will minimize to chemically different structures, enabling us to carry out further studies such as reaction pathway optimizations. Because the predictor has imperfect accuracy, the predicted event is generally not identical to the actual event, and the pair of frames corresponding to the current and next trajectory frame will minimize to chemically identical structures. Thus, we should quantify the accuracy of our predictions using some measure of distance to the actual reaction events, or equivalently, by the amount of computational cost it requires to find the actual reaction events starting from the predicted ones. Because we have computed $\mathcal{E}_{\text{ref}}$ in § 2 .4.1, our goal is to optimize the parameters and characterize the accuracy of $\mathcal{E}_1$, thus enabling its application with more confidence in future applications where we do not have $\mathcal{E}_{\text{ref}}$.

In the ideal case, the set of detected reaction events and actual events are equal, and the complete set of reactant and product structures could be found by two energy minimizations for each detected reaction event, with a computational cost of $2 \cdot |\mathcal{E}_{\text{ref}}| \ll N_{\text{steps}}$ times the cost of a single energy minimization. On the other hand, if the predicted reaction event is located close in time to the actual event, then it could be found by energy minimizing more structures in a time window of increasing size around the detected event. As the time window around each element of $\mathcal{E}_1$ is increased (both forwards and backwards in time), an increasing number of true reaction events will be found, and the computational cost is increased as well. In the limiting case, the window size is equal to the entire trajectory length, and all of the reaction events in $\mathcal{E}_{\text{ref}}$ are found at a cost equal to computing $\mathcal{E}_{\text{ref}}$ itself. Thus, the detection method is deemed to be useful if it detects a greater fraction of reaction events in $\mathcal{E}_{\text{ref}}$ than the fraction of the trajectory that is optimized with a chosen value of the window size. By increasing the window size over a range $(0, N_{\text{steps}})$, we can interpolate between these two limits and construct a receiving operator characteristic (ROC) objective function.

The ROC is a commonly used statistical approach for evaluating the diagnostic ability

of a binary classifier, created by plotting the true positive rate (TPR) vs. the false positive rate (FPR) as a sensitivity threshold is varied.[127,128] In our definition of the ROC, we use a time window of variable size $n \cdot \delta$ representing the number of trajectory frames being optimized in the neighborhood of each detected reaction event in $\mathcal{E}_1$. The smallest possible set of optimized frames $\mathcal{X}^{(0)}$ corresponds to a window size of zero:

$$\mathcal{X}^{(0)} = \mathcal{E}_1 \tag{41}$$

where the superscript on $\mathcal{X}$ is the window size.

For any window size w, the set of frames being minimized $\mathcal{X}^{(w)}$ may be defined as:

$$\mathcal{X}^{(w)} \;=\; \left\{ i+n \;\middle|\; i \in \mathcal{X}^{(0)}, -w \leq n \leq w \right\} \cap \mathcal{T} \tag{42}$$

$\mathcal{X}^{(w)}$ is then used to determine the true positive rate ($\mathrm{TPR}^{(w)}$) and false positive rate ($\mathrm{FPR}^{(w)}$). $\mathrm{TPR}^{(w)}$ is the amount of reference reaction events in $\mathcal{E}_{\mathrm{ref}}$ included in the optimized frames $\mathcal{X}^{(w)}$ divided by the total number of reference reaction events $|\mathcal{E}_{\mathrm{ref}}|$. $\mathrm{FPR}^{(w)}$ is calculated as the fraction of trajectory frames not containing reaction events included in $\mathcal{X}^{(w)}$. These functions are defined as:

$$\mathrm{TPR}^{(w)} = \frac{\left|\mathcal{X}^{(w)} \bigcap \mathcal{E}_{\mathrm{ref}}\right|}{|\mathcal{E}_{\mathrm{ref}}|}; \quad \mathrm{FPR}^{(w)} = \frac{\left|\mathcal{X}^{(w)} - \mathcal{E}_{\mathrm{ref}}\right|}{|\mathcal{T} - \mathcal{E}_{\mathrm{ref}}|} \tag{43}$$

The parametric curve $(\mathrm{FPR}^{(w)}, \mathrm{TPR}^{(w)})$ is traced out as w increases, and the ROC objective function $\phi(\sigma, \mu)$ is calculated as the area under the parametric curve as shown in Figure 4.

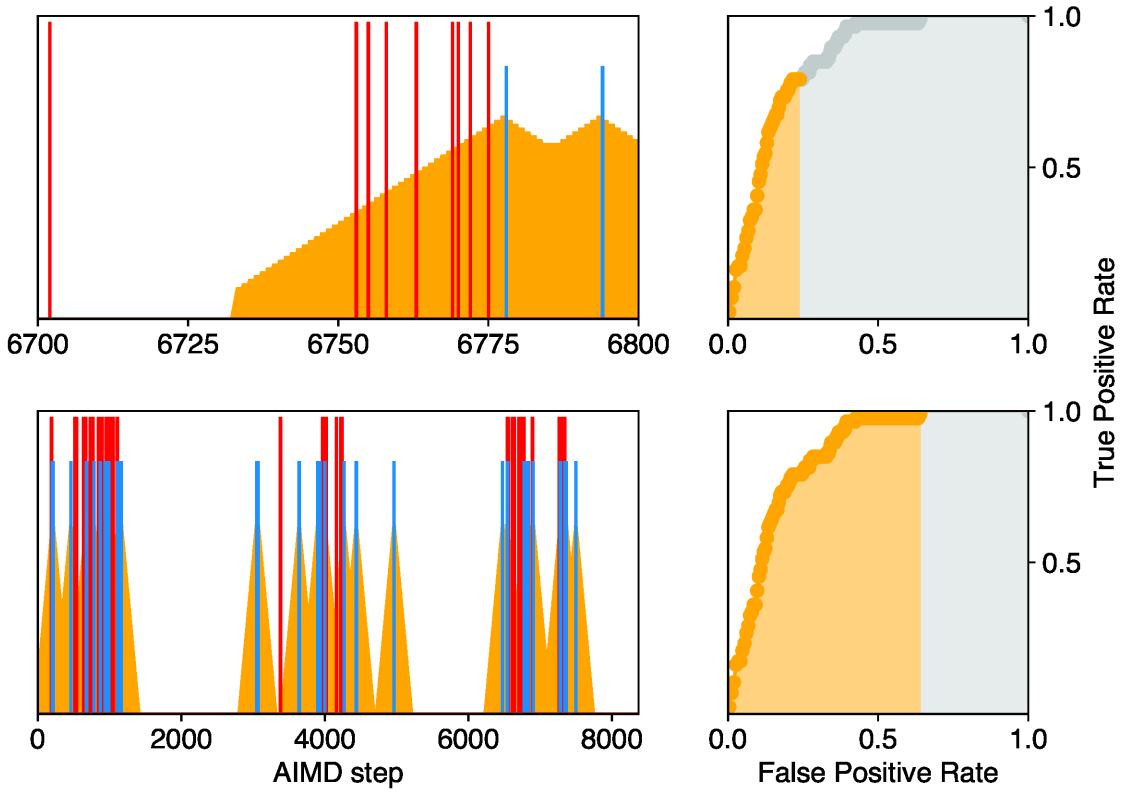

Figure 4: Left panels: Reference reaction event locations (red), BOTS reaction event locations (blue), broadening time windows (orange wedges) around BOTS reaction events. Top panels: Magnification of a 100-frame sequence of the trajectory showing the expanding time window. Bottom panels: Entire trajectory with sufficiently large window size for maximal true positive rate. Right panels: Plot of $\mathrm{TPR}^{(w)}$ vs. $\mathrm{FPR}^{(w)}$ (Equation 43) where orange region indicates the current value of w. The area under the whole curve is the ROC objective function.

The ROC score has an upper bound of 1.0 corresponding to perfect accuracy, i.e. all of the true reaction events are found using a window size of zero, whereas scores of 0.5 or lower indicate the method has no predictive power beyond a random number generator.

2 .4.4 Bond-wise criterion for reaction detection

The procedure defined above uses a unified set of detected reaction events across all atom pairs (Equation 40) to predict the total set of reaction events in the entire system. This

approach was found to be problematic because it ignores the local character of reaction events, i.e. a single reaction event involves changes in bond order for a particular subset of atom pairs. If the predicted reaction event could be mapped to an actual event where different bonds are broken or formed, then it would not be possible to identify the reactive sites within the system; this would become an important deficiency of the method for systems that contain multiple molecules. To resolve this issue, we defined a "bond-wise" criterion that ensures the detected and actual reaction events can only be matched if the changes in the pairwise bond orders are similar, which is adopted in the work.

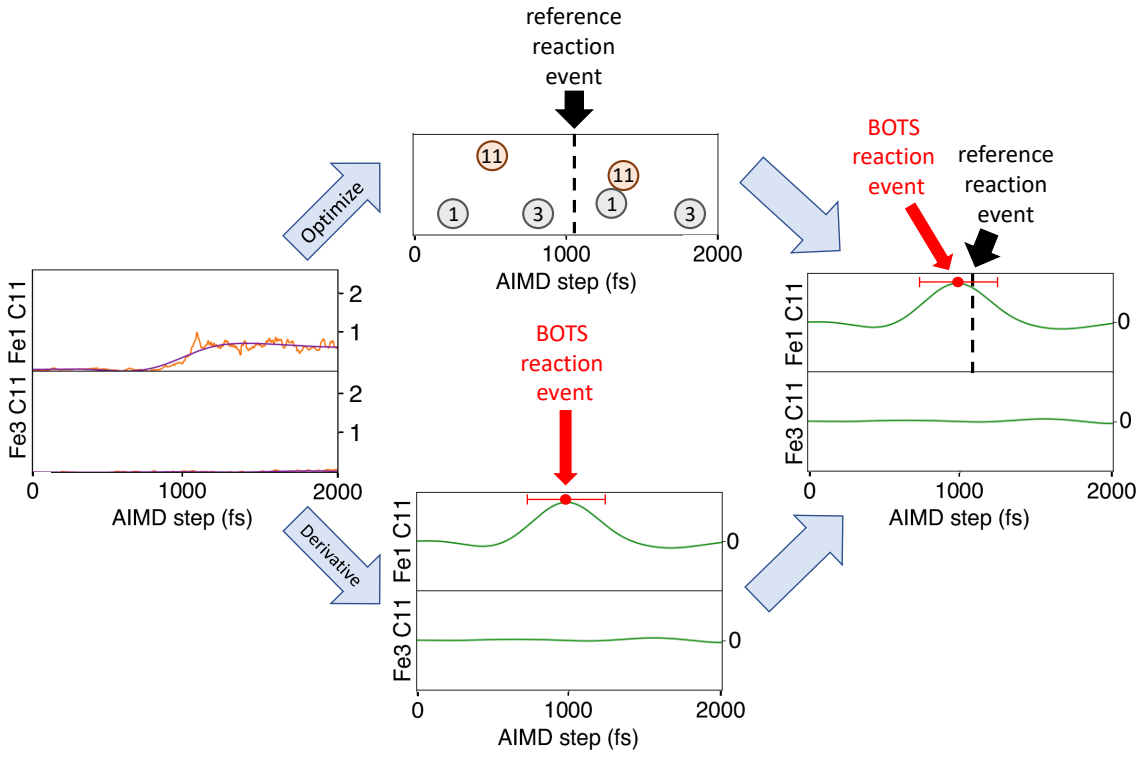

Figure 5: Left: Raw and smoothed bond order time series. Top: Optimized positions of atoms. Reference reaction event near 1000 fs experienced a large change in its optimized BO matrix from atom pair (1, 11). Bottom: BOTS method predicting a reaction event by identifying extrema in the first time derivative of the BO time series beyond a threshold. Right: The BOTS-predicted reaction event in (1, 11) is compared to the reference reaction event for that atom pair.

In the bond-wise ROC, illustrated in Figure 5, predicted reaction events in $\mathcal{E}_{1;ab}$ can only be matched to reference events that involve significant changes in the BO of atom pair

ab. This procedure involves defining the pairwise BO difference between clusters of energy-minimized structures. We first define an averaged BO matrix over the energy-minimized structures within the cluster as:

$$\widehat{M}_{ab}[i] \equiv \frac{1}{|C_{K[i]}|} \sum_{j \in C_{K[i]}} \widetilde{M}_{ab}[j] \tag{44}$$

where K is a cluster index, i, j are frame indices and a, b are atom indices. This enables the definition of absolute pairwise BO difference between clusters as:

$$\Delta_{ab}[i] \equiv \mathrm{abs}(\widehat{M}_{ab}[i+1] - \widehat{M}_{ab}[i]) \tag{45}$$

The reference reaction events involving atom pair (a, b), given by $\mathcal{E}_{\mathrm{ref},ab}$, is defined as:

$$\mathcal{E}_{\mathrm{ref},ab} = \{i \ | \Delta_{ab}[i] \geq 0.5 \max(\mathbf{\Delta}[i])\} \cap \mathcal{E}_{\mathrm{ref}} \tag{46}$$

where $\mathbf{\Delta}[i]$ is the BO difference matrix between clusters $K[i], K[i+1]$ and the maximum is taken over all pairs of atoms. Thus, each individual event in $\mathcal{E}_{\mathrm{ref}}$ may be included in one or more bond-wise sets $\mathcal{E}_{\mathrm{ref},ab}$.

The trajectory frames being optimized within a time window w of $\mathcal{E}_{1;ab}$ is denoted using $\mathcal{X}_{ab}^{(w)}$ and defined in a similar manner to Eqs.41-42 with $\mathcal{E}_{1;ab}$ replacing $\mathcal{E}_1$. The true positive rate with the added bond-wise criterion is then defined by taking the union over all successfully found reaction events in the numerator:

$$\mathrm{TPR}'^{(w)} = \frac{\left| \bigcup_{b>a=1}^{N_{\mathrm{atoms}}} \left(\mathcal{X}_{ab}^{(w)} \bigcap \mathcal{E}_{\mathrm{ref};ab} \right) \right|}{|\mathcal{E}_{\mathrm{ref}}|} \tag{47}$$

The corresponding false positive rate represents the ratio of all energy-minimized frames not corresponding to reaction events in the numerator, and the same denominator as in Equation 43. Due to the extra condition imposed by the bond-wise criterion, the numerator

may slightly exceed the denominator when $\left|\mathcal{X}^{(w)}\right|$ approaches the trajectory length; the FPR is set equal to 1.0 when this occurs. Additionally, the predicted reaction events from a random number generator no longer result in a ROC of 0.5 due to the additional conditions imposed on matching a reference reaction event to a predicted one.

$$\mathrm{FPR}'^{(w)} = \min\left(\frac{\left|\mathcal{X}^{(w)} - \bigcup_{b>a=1}^{N_{\mathrm{atoms}}}\left(\mathcal{X}_{ab}^{(w)} \bigcap \mathcal{E}_{\mathrm{ref};ab}\right)\right|}{|\mathcal{T} - \mathcal{E}_{\mathrm{ref}}|}, 1\right) \tag{48}$$

Similar to before, the bond-wise ROC $\phi'(\sigma, \mu)$ is calculated as the area under the parametric curve $(\mathrm{FPR}'^{(w)}, \mathrm{TPR}'^{(w)})$. We will drop the primes in the next section, as our results will use the bond-wise criterion exclusively.

2 .5 Results and Discussion

In this section, we characterize the performance and parameter sensitivity of our reaction detection models. The primary means of measuring performance is the ROC $\phi(\sigma, \mu)$ discussed above, and the parameter sensitivity is characterized by observing how the ROC varies with respect to its two parameters: the cutoff frequency in the low pass filter σ (given in cm^{-1}, and the threshold on the time derivative μ given in units of multiples of σ. Because the parameter space is two-dimensional, the global optimum and parameter sensitivity can be obtained by plotting $\phi(\sigma, \mu)$ as a heat map.

2 .5.1 Heptanylium cation

The reaction events observed in the AIMD trajectory for heptanylium cation ($C_7H_{15}{}^+$) mostly involve hydride shifts where H^- is transferred from a non-terminal CH_2 group to the neighboring trivalent carbon with a formal positive charge. The energy-minimized local minima, shown in the top row of Figure 7, include carbocation species with a formally positive trivalent carbon (clusters 3-6) as well as carbonium species with a pentavalent carbon

(clusters 1-2). The heat map for heptanylium in Figure 6 shows that the ROC objective function $\phi(\sigma, \mu)$ has values above 0.95 in a broad region of parameter space, indicating a high degree of accuracy in detecting reaction events that is not highly sensitive to parameter choice. The objective function value indicates that most or all of the predicted events with only small time differences from the reference events. The global optimum is given as $\phi(\sigma = 140 \text{ cm}^{-1}, \mu = 1.0) = 0.97$.

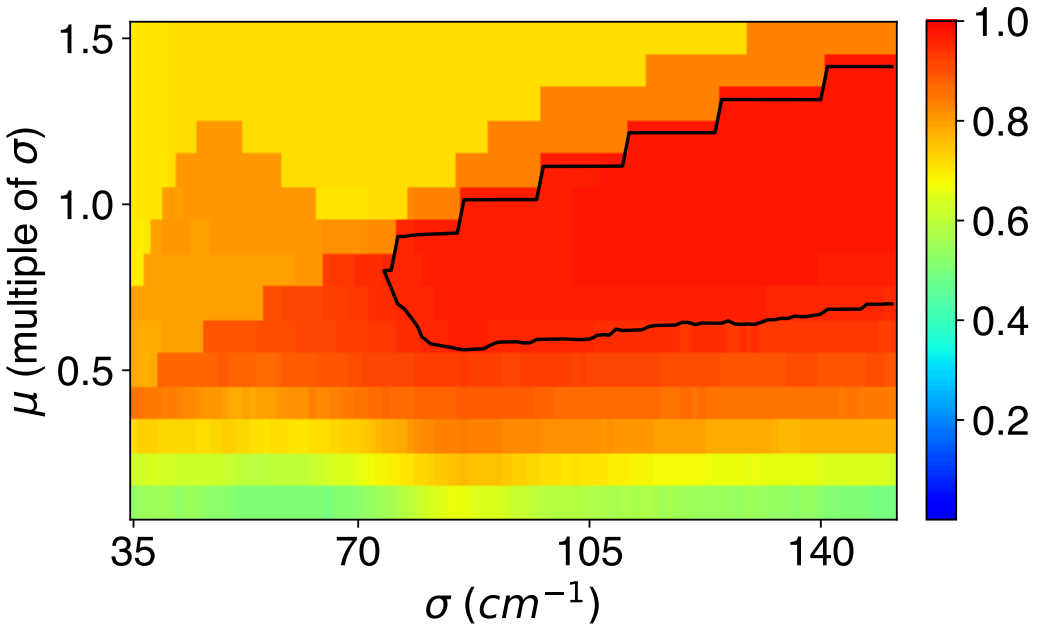

Figure 6: Heat map of bond-wise objective function scores for $C_7H_{15}^+$ using different combinations of σ and μ. Black contour indicates scores above 0.95.

Figure 7 examines the level of agreement between predicted and reference reaction events using the optimal parameter combination of $\sigma = 140 cm^{-1}$ and $\mu = 1.0\sigma$ identified from the heat map. There are 17 predicted and 13 reference reaction events respectively, and the maximum time difference between any reference event and the nearest predicted event that satifies the bond-wise criterion was 42 frames. The set of energy-minimized trajectory frames using a window size of 42 ($\mathcal{X}^{(42)}$) covers 7.9% of the whole trajectory, which is another indicator of the accuracy of the reaction detection model. Figure 7 also shows the starting and ending cluster numbers for each reference reaction event. The colors of arrows indicate

the time difference between the reference and predicted reaction events. From this data, we observed that reference reaction events have a tendency to occur in multiplets due to re-crossing of dividing surfaces. Some reference reaction events occur in closely spaced opposite pairs, such as cluster number 6 which is visited once from cluster number 5.

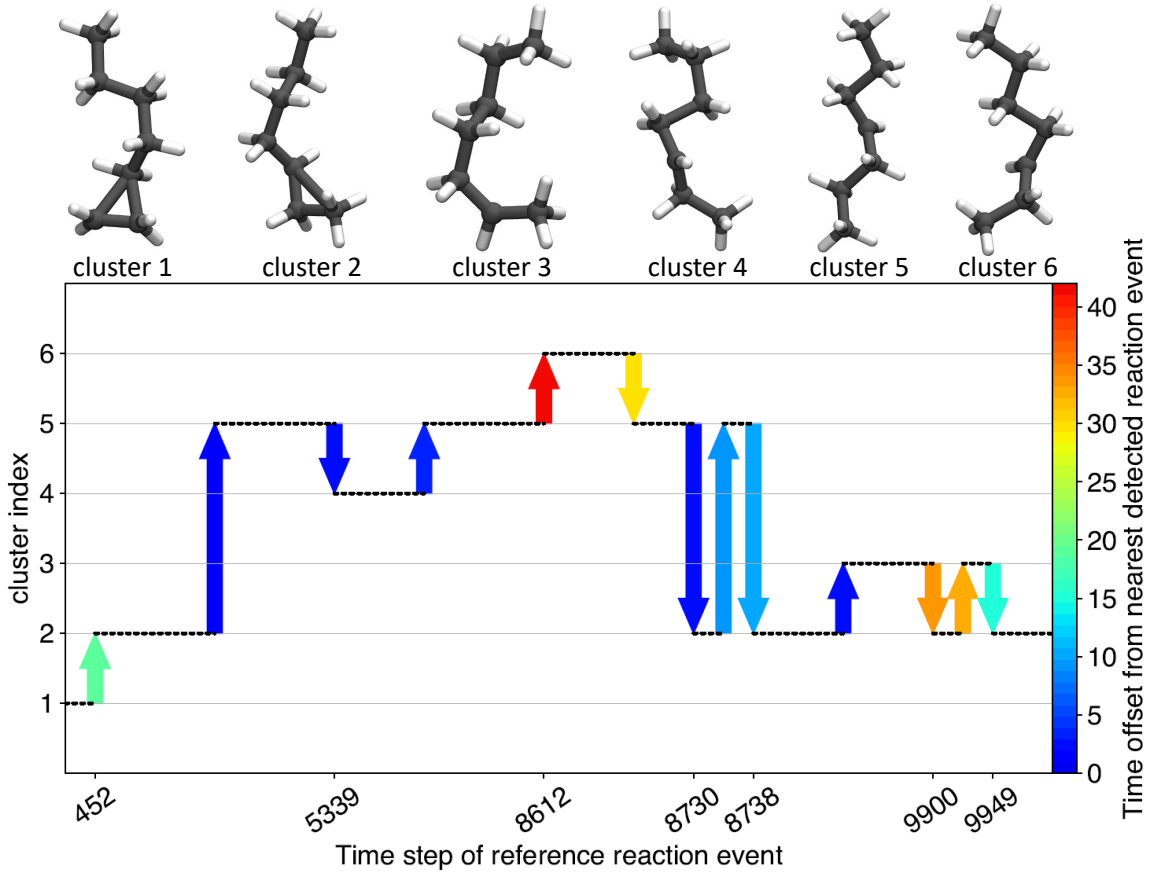

Figure 7: Summary of reaction events and detection model in heptanylium cation ($C_7H_{15}^+$) simulation trajectory. Top: Chemically distinct clusters found after minimizing all trajectory frames followed by clustering. Bottom: Comparison of reaction events for reference and BOTS predictions. Horizontal coordinates of arrows indicate the time step (x axis not to scale), vertical coordinates indicate starting and ending cluster, and color indicates temporal proximity to nearest BOTS prediction. Parameter combination of $\sigma = 140cm^{-1}$ and $\mu = 1.0\sigma$ was chosen from the optimal parameter range shown in Figure 6.

We also investigated basis set dependence of BO time series, and the results are shown in Figure S11 and Table S1. Figure S11 reveals that for the segment of the time series between 8500-8800 fs the BOs do not differ significantly between basis sets. Sharp changes in BO seen

for atom pairs C10-H13, C10-H15, C12-H13, and C12-H15, which led to successful reaction event predictions in the original 6-31G* basis set, are also apparent in the 3-21G and TZVP bases. This is important because our detection method uses rate of change of the BO to predict reactions, so minor differences in overall BO magnitude between bases should not matter as long as the sharp BO changes are consistent. Table S1 reveals small RMS errors of about 0.05 with a few larger deviations. The largest RMS errors (bold in Table S1) occur time series with significant high frequency components such as C3-C8 and C10-C12. Because our method involves filtering out high frequency components, we think the time-dependent BO variations in these atom pairs is expected to be consistent across bases. Thus, we are confident in the transferability of parameters between different basis sets in similar systems.

2 .5.2 Iron carbonyl cluster

The AIMD trajectory for the iron carbonyl cluster begins with an optimized structure reported by Schaefer and coworkers,[119] denoted as "9a" in their publication. This system is characterized by nearly constant, almost fluid migration of carbonyl ligands throughout the duration of the simulations, whereas the Fe atoms move more slowly due to their increased mass. The carbonyls migrate by breaking and forming coordinations with individual irons and breaking and forming bridging relationships across multiple irons.

The reactivity in this system is more difficult to characterize compared to the heptanylium system for several reasons. One reason is that the number of chemically distinct clusters and reaction events was simply higher in this trajectory. Perhaps more importantly, the chemical bonding in this system is less discrete compared to the previous case, because the Fe-C and Fe-Fe bond orders of the energy minimized structures are more broadly distributed between 0 and 1. This is also evident in the dendrogram of Figure S2, which shows that the number of clusters and reference reaction events has a significant dependence on the clustering threshold. Thus, this system approaches the limits of our basic assumptions that the potential energy surface consists of discrete and well-separated chemical species.

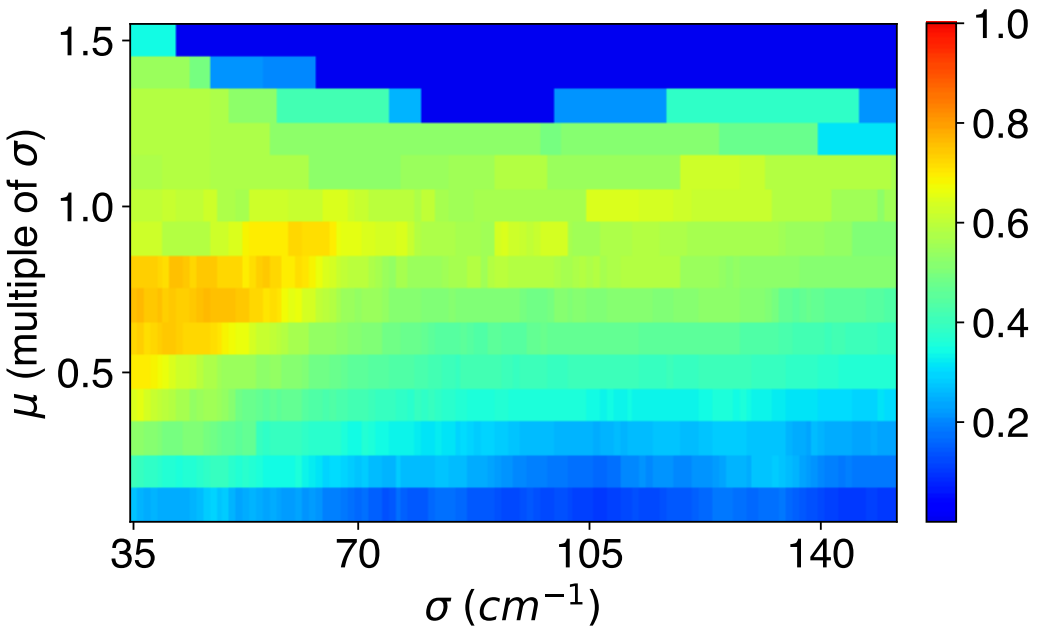

Figure 8: Heat map of ROC objective scores scores for $Fe_3(CO)_9$ as a function of σ and μ.

The heat map for the $Fe_3(CO)_9$ system is shown in Figure 8. Compared to the $C_7H_{15}{}^+$system, the objective function scores are generally lower and there is no parameter combination that gives a score above 0.9, but there still exists a region of parameter space that gives the optimal result as indicated by the orange area. These ideal parameter combinations occur lower σ values and lower μ values than in Figure 6, which we think are due to the slower overall dynamics of the system, owing to the increased mass of Fe and perhaps the relatively flat potential energy surface along reaction coordinates. The difference in optimal parameters between the heptanylium and the iron carbonyl simulation trajectories indicates that this method is not completely system independent. However, it does appear possible to choose parameter sets based on the elemental composition of the system without needing to determine parameters for each individual AIMD trajectory. We expect that parameter combinations should be reasonably transferable between chemically similar systems that have similar variations in the time scales and magnitudes of BO variations. We think that for organic reactions consisting of only C and H atoms, the parameters we identified for the hep-

tanylium cation may be directly used without modification. This is based on our observation that most organic reactions involve well-defined and rapid transitions between metastable potential energy minima. However, we did not carefully assess the transferability of these parameters in this paper, and it is possible these parameters are applicable to even wider organic reactions containing heteroatoms, or on the other hand they may only be appropriate for alkyl carbocation rearrangements. On the other hand, the iron carbonyl system is an atypical and highly challenging case in the sense that the chemical structures are somewhat less well-defined and the reactivity is more fluxional. We expect the parameters in this paper to be transferable for other iron carbonyl complexes, but more work is needed to assess the transferability to other transition metal-containing systems. We plan to include more careful studies of parameter transferability in a future paper.

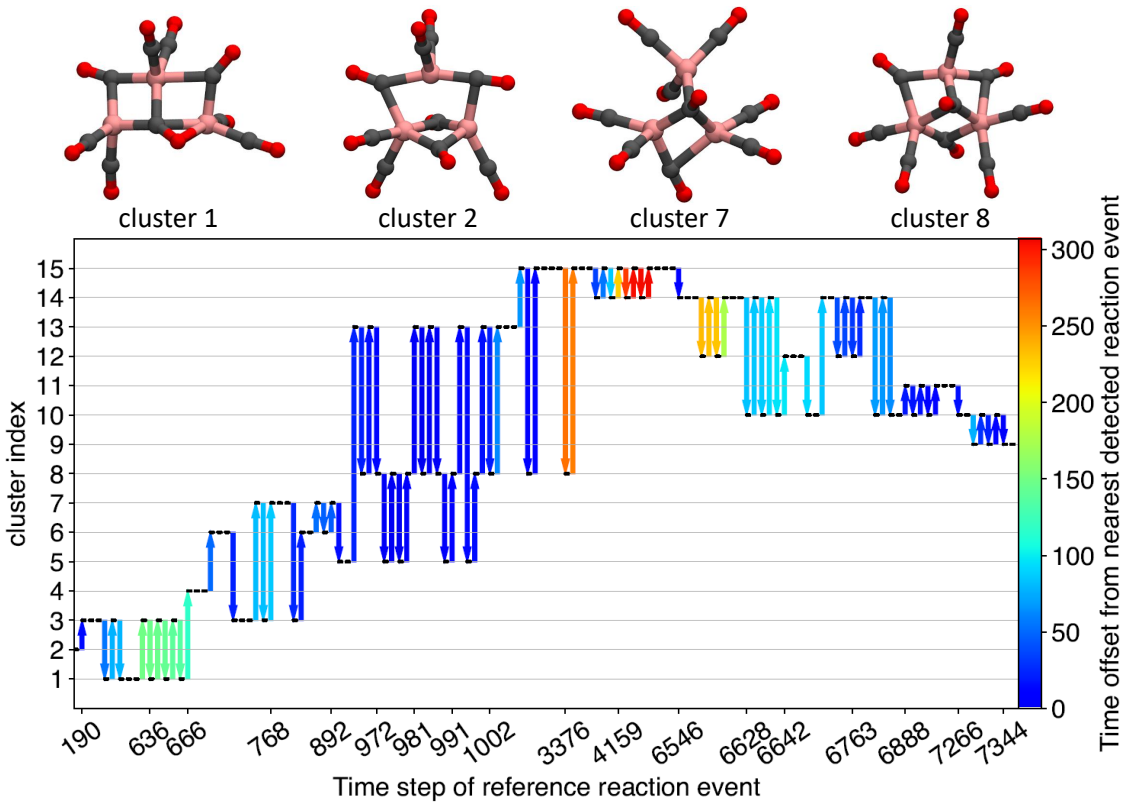

Figure 9: Summary of reaction events and detection model in $Fe_3(CO)_9$ simulation trajectory. Top: Selection of chemically distinct clusters found after minimizing all trajectory frames followed by clustering. All clusters in Supporting Figure S8. Bottom: Comparison of reaction events for reference and BOTS predictions. Horizontal coordinates of arrows indicate the time step (x axis not to scale), vertical coordinates indicate starting and ending cluster, and color indicates temporal proximity to nearest BOTS prediction. Parameter combination of $\sigma = 40cm^{-1}$ and $\mu = 0.7\sigma$ was chosen from the optimal parameter range shown in Figure 8. Molecule color scheme: iron: pink, carbon: gray, and oxygen: red.

A representative parameter set obtained from the optimal range in Figure 8 for $Fe_3(CO)_9$ is given by $\sigma = 40cm^{-1}$ and $\mu = 0.7\sigma$. Using those parameters, Figure 9 shows the temporal proximity of reference reaction events to the nearest BOTS predictions that satisfy the bond-wise criterion. The data shows that reference reaction events have a strong tendency to be grouped together as the dividing surface is crossed multiple times within a short simulation time. There is also a large variation in the "difficulty" of detecting certain reaction events vs. other ones, as indicated by the temporal distance between the reference reaction event and

the closest detected event. If a window size of 150 frames is used, 87% of reference reaction events can be found, which would require energy-minimizing 50% of the trajectory frames. To find the remaining 13% of reference reaction events in this trajectory, the time window needs to be 310 frames, which covers 68% of the trajectory. Closer inspection of the most difficult reference reaction events reveals that they occur in closely spaced opposite pairs, where the cluster index jumps to a new value for ∼10 frames then back again. Thus, we think that for challenging systems such as these, it may not be necessary to find 100% of the reaction events in order to get a comprehensive picture of the reactivity of the system. In applications where computational cost is a critical concern, the reference reaction events may be found more quickly (if not as thoroughly) using other methods such as skipping frames when extending the window, that may be more relevant as post-processing approaches than objective functions. In this context, the objective function score should not be seen as a literal measure of computational cost savings, but rather as a measure of the accuracy of reaction event detection.

2 .6 Conclusion

This paper describes how the time series analysis of bond orders is able to produce accurate predictions of the spatial and temporal locations of reaction events in reactive *ab initio* molecular dynamics trajectories. Reaction events in simple systems like hydrocarbons can be predicted with great accuracy; more complex and fluxional systems like iron carbonyl clusters contain reaction events that may still be identified, though not as easily. The accuracy of reaction event prediction can translate into more efficient computations, as it reduces the portions of the simulation trajectory that need to be examined in greater detail using methods such as geometry optimization. Our reaction detection method contains two adjustable parameters that are not fully system independent, but the optimized parameters of a system are expected to be broadly useful for simulations of chemically similar systems.

A natural extension of this research would be to identify reaction events in multi-

molecular simulations in a more rigorous manner. The challenges to be addressed include how to identify the subset of atoms in the overall system that are involved in a given reaction event, which could also be informed by analysis of the bond order matrix. Because the bond order matrix contains rich information about the chemical structure of the system, it might also be a useful collective variable for future metadynamics or other enhanced-sampling simulations to rapidly explore the chemical space. We anticipate that the bond order matrix will play an increasingly important role in reaction discovery as these methods continue to be developed and applied to chemical problems.

2 .7 Acknowledgements

We would like to acknowledge the ACS-PRF award 58158-DNI6 and the NSF ChemEnergy REU site, award CHE-1560479. C.S. is grateful for an E. K. Potter Stanford Graduate Fellowship and support through NSF ACI-1450179.

2 .8 Supporting Information

The Supporting Figures show dendrograms, atom pair-wise visualizations of the objective function, distance-ranked reference set reactions, and BO verses energy-minimized BO time series for both the heptanylium and $Fe_3(CO)_9$ systems. In addition, there is a figure comparing the "unified" and "bond-wise" objective functions for the $Fe_3(CO)_9$ system. There is also a figure containing the cluster images and energies for the $Fe_3(CO)_9$ system including those not depicted in Figure 9. Finally, there is a figure and an accompanying table that examine basis set sensitivity.

3 Polarizable Force Fields for Choline Chloride Urea Deep Eutectic Solvents

3 .1 Introduction

Deep eutectic solvents (DESs) are a relatively recent discovery and have been identified to have promising applications.[63] DESs are similar to ionic liquids, but where ionic liquids are cation-anion pairs, DESs are instead composed of hydrogen-bonding organic molecules in special eutectic mixtures. Eutectic mixtures are whole-number ratio combinations of molecule pairs that when mixed together, the resulting combination has a lower melting point than either component individually. The DESs of interest in this study contain choline chloride paired with one of a number of hydrogen bond donor molecules in the appropriate eutectic ratio. The hydrogen bond donor candidates include: urea, glycerol, phenol, ehtylene glycol, levulinic acid, oxalic acid, and malonic acid.[129] The attributes of choline chloride-based DESs depend on the identity of the included hydrogen bond donor (each of which requires a certain ratio). Understanding the bulk properties of DESs is of interest because it can aid in refining or finding new DES combinations as well as improving or expanding their application. The goal of this project is to create a semi-automated method for initializing AMOEBA polarizable force fields of DESs. The longer-term goal is to use that tool compare the bulk properties of an AMOEBA choline-chloride urea DES force field to experimental data, non-polarizable force fields, and other polarizable force fields.

At the beginning of this project, there were a handful of nonpolarizable DES force fields, but no polarizable DES force fields. We set out to create polarizable DESs within the AMOEBA force field and compare them with existing nonpolarizable DESs. Over time, setbacks caused the original goal to be modified towards automation of AMOEBA polarizable force field initialization. During the course of this project, other research groups have published studies of DES in other polarizable force field engines. In 2021, Jeong et al. created a polarizable DES force field for reline (choline chloride and urea) using symmetry

adapted perturbation theory (SAPT).[72] They adjusted base SAPT parameters to be better suited for the extensive hydrogen bonding of reline based on first principles MD calculations. The resulting reline force field Jeong et al. created showed good agreement with QM data and they noticed significant improvement on how their force field handed charge correlation structure factor compared to non-polarizable force fields. Also in 2021, Goloviznina et al. created polariable force fields for DESs, but built them inside their CL&Pol engine for ILs.[130,131] They used a Tang-Toennies function to smooth and dampen interactions between dense charges and induced dipoles. Those charge interactions had a tendency to derail MD trajectories because of their strong pull. Despite this recent emergence of polarizable DES force fields, there is still room for discovery and development. Polarizable force fields for DESs still have not yet been created using the AMOEBA force field. Also, none of the new polarizable force fields for DESs have had their parameters systematically optimized to reproduce experimental and QM data simultaneously. A tool that is capable of this style of force field parameter optimization is ForceBalance (FB).[132] FB is an optimization engine that is capable of fitting diverse targets simultaneously using a single objective function. FB already has a history of successful application and Section 3 .3 explains in more detail how FB has been used to significantly improve how water models and graphene force fields fit experimental data.[133–135] This project lays groundwork towards formation of AMOEBA force fields for DESs with heightened accuracy due to FB optimizing force field parameters.

As of 2011, the AMOEBA force field has the following functional form:[136]

$$
\begin{aligned}
U_{AMOEBA} &= \sum_{bond} k_b(b-b_0)^2[1-2.55(b-b_0)+3.793125(b-b_0)^2] \\
&+ \sum_{angle} k_\theta(\theta-\theta_0)^2[1-0.014(\theta-\theta_0)+5.6x10^{-5}(\theta-\theta_0)^2-7.0x10^{-7}(\theta-\theta_0)^3 \\
&+ 2.2x10^{-}8(\theta-\theta_0)^4] + \sum_{torsion}\sum_{n}\Big\{\frac{V_n}{2}[1+cos(n\phi-\lambda)]\Big\} + \sum_{PI-torsion} U_{PI-tor,i} \\
&+ \sum_{torsion-torsion} U_{tor-tor,i} + \frac{1}{2}\sum_{multipole} U_{MPol,i} + \sum_{polarization} U_{pol,i} \\
&+ \frac{1}{2}\sum_{Van-der-Waals} \epsilon_{ij}\Big(\frac{1+\delta}{\rho_{ij}+\delta)}\Big)^{n-m}\Big(\frac{1+\gamma}{\rho_{ij}^m+\gamma}-2\Big)
\end{aligned}
\tag{49}
$$

where the energies of the bonded-interaction parameters are similar to those in Equations 24-26, but with added correction factors. The other term with an explicit equation is the van der Waals term which is known as the buffered 14-7 potential.[137–139] Within the buffered 14-7 potential, the δ and λ terms are buffering constants, n and m control the exponents and are frequently equal to 14 and 7 respectively, and ρ_{ij} is a distance ratio.[139] Note that $n = 12$ and $m = 6$ recovers the 12-6 Lennard Jones potential (Equation 27).[139] When creating buffered 14-7 potential, other combinations of n and m were explored, but $n = 14$ and $m = 7$ yielded the best results.[139] AMOEBA is somewhat unique in its usage of the buffered 14-7 potential, since most other force fields use the Lennard Jones potential (Equation 27) or the Buckingham potential. The buffered 14-7 potential was selected as the VDW term for AMOEBA because Thomas Halgren showed that it better represented high quality noble gas data compared to the Lennard Jones potential.[138–140]

A lot of work is being done both using and refining the AMOEBA force field.[141] Some of the applications of the AMOEBA force field include: nucleic acids, proteins, and water.[141–143] The goal of this project is to contribute to the ongoing usages of AMOEBA by creating polarizable force fields for DES components.

This chapter reviews a semi-automated code for initializing polarizable AMOEBA force

fields. Also, the utilization of the force field optimizer ForceBalance is discussed. The force field initialization process and the force field optimization process were fraught with challenges. These challenges are addressed as well as the various workarounds that were discovered as a guide to others intending to build upon this work.

3 .2 Initializing AMOEBA Polarizable Force Fields

3 .2.1 Initialization Tutorial

The Ponder group have designed a number of tutorials to initialize different kinds of AMOEBA polarizable force fields using Tinker. The main tutorial referenced in this work uses Tinker and other software including: Gaussian, OpenBabel, and Stone's GDMA program.[136] As outlined, the the tutorial involves multiple steps optimizing structures and feeding program output (files and screen readouts) to other programs. While possible to navigate the various tutorial steps by hand, the process is prone to user error and becomes tedious when used for many candidate molecules.

3 .2.2 Semi-Automated Initialization Method

Creating input files and harvesting relevant output file data by hand leaves a lot of room for human error. A Python code that (mostly) automates the initialization procedure is outlined here. The semi-automated code streamlines the initialization procedure and removes most sources of human error. The code is not fully automated, because there are certain steps in which the programs involved are prone to error and are not easily bypassed without human input. This is most evident between steps 8 and steps 10 of the code where Tinker has a hard time determining chemically equivalent atoms (via symmetry). The goal of the Python code is to start with an input molecule coordinate file in the XYZ format and produce as output an AMOEBA force field file.

For the sake of explanation molecule.xyz is used as an example input. The code also requires an existing AMOEBA force field file (existing.prm) because the first step of ini-

tialization uses Tinker's poledit method requires one for some unknown reason. The input AMOEBA force field does not appear to be a template, and the requirement was added sometime after the referenced tutorial was created in 2011 for the Supporting Information of "Polarizable Atomic Multipole-Based Molecular Mechanics for Organic Molecules".[136]

One shortcoming of Tinker's native force field initialization process is that charged molecules cannot be used as input. It is possible to generate a force field using a charged molecule, but the force field will not exhibit physical behavior. The issue appears to lie in how Tinker parameterizes the multipole parameters when there is a strong charge on any of the atoms within the target molecule. In the early stages the force fields were initialized by hand, by creating a starting geometry file with Avogadro and following the steps in the tutorial.[136] This approach was not only tedious, but also prone to human error if ever the wrong file was selected across the numerous steps. Many files with the same extensions are collected and must be carefully named to avoid mishaps. Other potential sources of human error were the numerous steps involving copying parameters from one output file (or screen output) into another file and other similar keyword additions. There was a large number of accumulated files by the end and plenty of opportunities to accidentally miss something important.

To streamline the process, I created a semi-automated Python code that can take an input XYZ file and return a Tinker polarizable force field file. The code document can be viewed in the Supporting Information section for this chapter in Listing B 3. After the user intervention following Part 1 of the code, the user can run Part 2 to obtain the initialized force field file. The code initializes a force field with the following steps:

1. **PART 1:** Required programs for force field initialization are: Tinker, Guassian, Anthony Stone's Gaussian Distributed Multipole Analysis (GDMA), and OpenBabel.[144–148]

   NOTE: In this work, the following versions were used: Tinker 8.7.1, Gaussian 16, GDMA 2.3 patch 1, OpenBabel 2.3.1 , together with Python 3.7.[149]

2. Create molecule.xyz, a required input file for the initialization code.

   NOTE: Make sure to optimize the structure before proceeding to make the subsequent optimization steps more straightforward. In this project, molecule.xyz was created by building the molecule in Avogadro (version 1.1.1) and using the simple, built-in, UFF force field with 4-step steepest descent to minimize the molecule's energy before exporting the coordinates.[150,151]

3. Run initialization code providing inputs for coordinates, charge, spin, molecule name, part (1 or 2 of the code), starting atom type index, and starting atom class index.

4. In addition to the above, provide a Tinker force field file (or a symbolic link to one).

   NOTE: This step deviates from the reference polarizable force field initialization tutorial and seems to have been added to later versions of Tinker.[136] The purpose of this step in unclear, as it is a required input for Tinker, but does not appear to affect the resulting force field. One hypothesis was that this input force field was used as a template for the new force field, but after some verification it appears to be required for some other reason.

5. Run a Gaussian geometry optimization with the MP2/6-311G(1d,1p) level of theory, using the molecule.xyz coordinates as the starting point.

6. Run a Gaussian formchk method using the checkpoint file from previous optimization step to create a formatted checkpoint file.

   NOTE: In this step the binary checkpoint file is converted to a formatted checkpoint text file which is readable with many text editors and importantly, GDMA and OpenBabel.

7. Run GDMA to generate .gdmaout output required by Tinker.

NOTE: GDMA stands for Gaussian Distributed Multipole Analysis which is a means of assigning atomic multipoles from Gaussian wavefunctions.[146,152] The multipole moments are often centered on atoms, but can be defined in other ways. Earlier DMA methods partitioned the density in basis space, and while it was considered better to partition in physical space, methods which attempted that were costly. Methods which partitioned the density in basis space alone are flawed because the resulting DMA was highly basis set dependant. What Stone did to combat that was maintain the partitioning of the density in basis space for compact functions and numerical quadrature for diffuse functions. The GDMA program allows the user to use the older basis set approach or the hybrid basis set-spatial partitioning technique. The creators of the initialization tutorial ellected to use the original GDMA approach and that option is therefore selected in the GDMA input file.[136]

8. Run Tinker's poledit method.

   NOTE: The Tinker poledit method uses the output file of GDMA to convert the global-frame multipole values into Tinker permantent multipole values. Tinker offers some options for placement of the frame definitions, but this method selects the default options. Tinker also selects polarization groups in this step.

9. End of Part 1, before running Part 2, confirm that the current force field molecule.key has correctly assigned atom types. If not, manually edit the top lines to have the correct atom types. Tinker attempts to identify equivalent atoms (similar by molecular symmetry and chemically equivalent), but often does not correctly identify each of them. Equivalent atoms missed by Tinker need to assigned by the user before proceeding. This is the one step that requires human intervention in the form of utilizing chemical intuition and thus breaks this automated procedure into two parts.

10. **PART 2:** Run Tinker's prmedit (parameter edit) function.

NOTE: Tinker automatically indexes atom type and atom class starting from unity which is fine for force fields used in isolation, but becomes problematic when combining them. Force fields must have unique indices before being combined, because they will not function with doubly-assigned atom types/classes. Uniqueness of indices must be accounted for whether the user intends to combine the resulting force field with others they create, or with files such as amoeba09.prm. The indices are not easily edited once the force field is created. A side effect of custom indices is that it makes creating coordinate files more difficult. Tinker utilizes its own coordinate format (TXYZ) which unfortunately defaults to the same extension as XYZ even though they are different file formats. Unlike XYZ, TXYZ requires atom type information. Therefore, when creating TXYZ coordinates, the atom types need to be corrected or added to match the custom values of the force field. For clarity in this project, since both XYZ and TXYZ are used extensively in the tutorial, TXYZ coordinates are instead given a .txyz extension.

11. Insert atom class indices.

    NOTE: At some point, the Tinker methods do not include parameter classes, so a custom function in the code takes care of inserting them at this time. Even though atoms of different types can share atom classes, the default in this code is set to give each atom type its own unique atom class. Doing this avoids another human input step, but possibly increases the parameter bloat of the force field which may negatively affect parameter optimization down the line. Not sharing atom classes is not expected to negatively affect the accuracy of the force field after FB parameter optimization. However, the effects of atom class sharing should be verified with some test cases.

12. Run OpenBabel on the Gaussian log file produced in Step 5 to create molecule_opt.xyz (coordinates of the optimized structure).

13. Using molecule_opt.xyz, run another Gaussian optimization at a higher level of theory: MP2/aug-cc-pVTZ.

14. Run Gaussian formchk on the checkpoint file from the second Gaussian optimization and use OpenBabel to convert that formatted checkpoint file to a TXYZ file.

    NOTE: Because the XYZ file format used in the Gaussian optimizations has no atom types or classes, OpenBabel attempts to recognize and assign them in the file format conversion (possibly based on the most current AMOEBA force field). However, since the user is using custom atom types and classes (either indexed from unity by default or another starting point) that are not compatible with whatever reference OpenBabel is using for assignment, the code corrects those before the TXYZ file is used.

15. Run Tinker potential option #1. This creates an input file in order to run Gaussian's cubegen function.

16. Run Gaussian cubegen method. This method calculates the potential at each grid point and produces a .cube file.

17. Use the above .cube file from Gaussian as input to run Tinker potential option #2.

    NOTE: Here Tinker creates a .pot file which contains the QM potential (from Gaussian) at each grid point. A keyword is added after this current step and prior to the next step to exclude the monopole values from optimization.

18. Run Tinker potential option #6. The atomic multipoles are fit to the QM electrostatic potential.

19. A custom function then combines the newly fitted multipole parameters with previously obtained parameters into most current parameter file.

20. Run Tinker valence option #1 to assign a default set of VDW parameters.

21. Use OpenBabel to convert second Gaussian optimization coordinates from TXYZ to XYZ (molecule_opt_opt.xyz).

22. Run Gaussian frequency calculation using molecule_opt_opt.xyz as input.

23. Tinker valence option #2.

    NOTE: This step does not change the force field parameters, but rather compares the current AMOEBA force field vibrational frequencies to those from the Gaussian frequency calculation. A readout of this comparison is saved to the directory.

24. Tinker valence option #3.

    NOTE: This part of the valence method provides adjustment of the bond and angle values and the force constants based on the Gaussian frequency calculation.

25. Final versions of the parameters are combined via a custom function into a final initialized force field file named molecule_final.key.

The Tinker tutorial also details steps for refining the torsion parameters, but those steps were difficult to execute correctly. Therefore, the part of the tutorial involving Tinker refining the torsion parameters was excluded from the code. The optimization capabilities of ForceBalance will be relied on for refinement of torsion parameters.

Left out of the list of steps are various intermediate parameter file re-naming steps which help differentiate the changes made to the force field over time. The initialization code also creates and saves input and output files for each of the programs called as a paper trail of which options were used at each step. The slowest steps in this process are the Gaussian optimizations.

Part of the Tinker group created an automated AMOEBA force field initialization procedure called Polytype 2.[153,154] Polytype 2 takes (preferrably Spatial Data File, SDF) input coordinates and generates a force field. Coordinate files without bond order components are

converted to SDF and bond order estimates are used. The method outlined in this section instead uses XYZ and TXYZ files without need of bond order. Automation of AMOEBA force field initialization is still an interesting space because this project intends to replicate AMOEBA force fields in OpenMM and use both Tinker and OpenMM in tandem. A custom initialization method also allows features to be tailored to the needs of this DES project.

3 .2.2.1 Urea Polarizable Force Field One of the possible hydrogen bond donor molecules that can be paired with choline chloride to make a DES is urea. The following AMOEBA-Tinker polarizable force field was created for urea using the code described in Section 3 .2.2.

```
forcefield AMOEBA-urea

bond-cubic              -2.55
bond-quartic            3.793125
angle-cubic             -0.014
angle-quartic           0.000056
angle-pentic            -0.0000007
angle-sextic            0.000000022
opbendtype              ALLINGER
opbend-cubic            -0.014
opbend-quartic          0.000056
opbend-pentic           -0.0000007
opbend-sextic           0.000000022
torsionunit             0.5
vdwtype                 BUFFERED-14-7
radiusrule              CUBIC-MEAN
radiustype              R-MIN
radiussize              DIAMETER
epsilonrule             HHG
dielectric              1.0
polarization            MUTUAL
vdw-12-scale            0.0
vdw-13-scale            0.0
vdw-14-scale            1.0
vdw-15-scale            1.0
mpole-12-scale          0.0
mpole-13-scale          0.0
mpole-14-scale          0.4
mpole-15-scale          0.8
polar-12-scale          0.0
polar-13-scale          0.0
polar-14-scale          1.0
polar-15-scale          1.0
polar-12-intra          0.0
```

```
polar-13-intra           0.0
polar-14-intra           0.5
polar-15-intra           1.0
direct-11-scale          0.0
direct-12-scale          1.0
direct-13-scale          1.0
direct-14-scale          1.0
mutual-11-scale          1.0
mutual-12-scale          1.0
mutual-13-scale          1.0
mutual-14-scale          1.0

FIX-MONOPOLE
atom     600     300     C     "urea     "     6     12.011     3
atom     601     301     N     "urea     "     7     14.007     3
atom     602     302     H     "urea     "     1     1.008     1
atom     603     303     O     "urea     "     8     15.999     1

#
# Multipoles from Electrostatic Potential Fitting
#

multipole    600   603   601              1.03617
                                          0.00000     0.00000     0.10461
                                          0.00735
                                          0.00000    -0.06314
                                          0.00000     0.00000     0.05579
multipole    601   600   602             -0.36361
                                         -0.11826     0.00000    -0.16264
                                          0.65336
                                          0.00000    -0.80469
                                         -0.08658     0.00000     0.15133
multipole    602   601   600              0.13019
                                          0.01149     0.00000    -0.18495
                                          0.11568
                                          0.00000    -0.06971
                                          0.02419     0.00000    -0.04597
multipole    603   600   601             -0.82971
                                          0.00000     0.00000    -0.22109
                                         -0.43084
                                          0.00000     0.21152
                                          0.00000     0.00000     0.21932
polarize     600          1.3340      0.3900     601   603
polarize     601          1.0730      0.3900     600   602
polarize     602          0.4960      0.3900     601
polarize     603          0.8370      0.3900     600
 Estimated van der Waals Parameters :

 vdw          300              1.900      0.0890
 vdw          301              1.855      0.1050
 vdw          302              1.350      0.0200      0.91
 vdw          303              1.650      0.1120

#
```

```
# Results of Valence Parameter Fitting
#
bond         300   301           614.17      1.3347
bond         300   303           947.09      1.1994
bond         301   302           565.71      1.0017
angle        301   300   301      18.50      152.31
angle        301   300   303       4.08      195.32
angle        300   301   302      38.88      116.05
angle        302   301   302      42.82      114.75



 Estimated Stretch-Bend Parameters :

 strbnd       301   300   301      18.70       18.70
 strbnd       301   300   303      18.70       18.70
 strbnd       300   301   302       7.20        4.30

 Estimated Out-of-Plane Parameters :

 opbend       301   300     0     0            14.40
 opbend       303   300     0     0            14.40

 Estimated Torsional Parameters :

 torsion      301   300   301   302       0.000 0.0 1   0.500 180.0 2   0.250
   0.0 3
 torsion      303   300   301   302       0.000 0.0 1   0.500 180.0 2   0.250
   0.0 3
```

Listing 1: AMOEBA-Tinker-urea force field file generated by the semi-automated code in Supporting Information B 3.

In AMOEBA-Tinker-urea (Listing 1), atom types (non-bonded terms) are integers indexed from 600 and atom classes (bonded terms) are indexed from 300. For any Tinker force field, the multipole parameters are charge, dipole, and quadrupole moments.[155] The VDW parameters are size (in Ångstroms), homoatomic well depth, and an optional reduction factor. The bond parameters consist of a force constant and an ideal bond length. The angle parameters consist of a force constant and up to three ideal bond angles. The lone opbend parameter is a force constant. And finally, the torsion parameters are triplet sets of amplitude, phase offset, and periodicity.

Urea is a straightforward test case since it is a small molecule with four readily identifiable atom types and atom classes.

3 .2.3 Choline Chloride

The intended quaternary ion to be paired with urea and the other hydrogen-bond donors in this study was choline chloride (Figure 10). Choline chloride is a quaternary ammonium salt that also has an alcohol functional group. However, multiple problems emerged when attempting to initialize the AMOEBA force field for choline chloride.

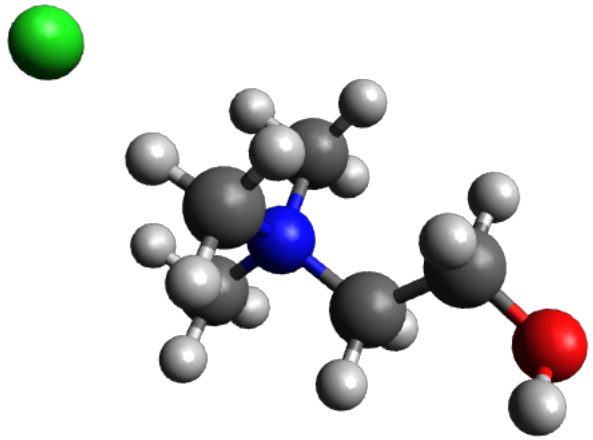

Figure 10: Choline chloride molecule, formula: $(CH_3)_3N(Cl)CH_2CH_2OH$. Atoms pictured are: carbon (dark grey), hydrogen (light grey), nitrogen (blue), oxygen (red), and chlorine (green)

Eventually, we decided to hold off on troubleshooting the various issues with choline and move forward with just the hydrogen bond donor molecules. The issues with choline chloride we experienced may be related to the polarization catastrophe.[156] The polarization catastrophe is when strong electrostatic interactions (like with ions such as Cl^- are involved) cause instability in the induced dipoles they cause.[130] The problems with the choline chloride force field were with the non-bonded terms, more specifically the multipole terms. Another explanation is that Tinker's internal process for multipole parameter assignment and refinement is not intended for ions. Whatever the issue ends up being, once fixed it will be straightforward to use the semi-automated code to initialize force field parameters for choline chloride. After which, those parameters can be combined with urea's or other hydrogen bond donors

to form a DES force field. This is where the parameter indexing of atom types and atom classes during initialization comes into play. The various hydrogen bond donor's parameters should be indexed such that they are compatible with those of choline chloride and the other donors.

3 .3 ForceBalance

ForceBalance (FB) is an optimization engine that can tune force field parameters by fitting to both experimental and calculated targets.[132] FB accomplishes this by allowing for diverse residuals to be included in a single objective function. Before the parameter optimization begins, priors are selected for force field parameters so they can be subject to a Gaussian prior probability distribution which informs FB about the expected range of the parameters.[157] The priors help penalize parameter deviations that stray far from expected physical behavior during the optimization process.

FB has already been used to tune TIP3P and TIP4P water force field models. FB improved their agreement with experimental values for density, dielectric constant, self-diffusion coefficient, and shear viscosity across a range of temperatures.[134] Another successful application of FB was an efficiency improvement of the AMOEBA water model with the creation of iAMOEBA.[133] The "inexpensive" iAMOEBA water model was able to reduce the cost of polarization while maintaining and sometimes improving overall accuracy. FB was also applied in the study of nanoporous graphene desalination processes and improvement of an AMBER protein force field.[135,158]

In this study, we fed 10,000 step QM MD data into FB to improve the parameterization of our small hydrogen-bond donor molecules. An important tool in FB's optimization process is constrained optimization. However, Tinker does not contain native constrained optimizations. A workaround for this issue is to convert the Tinker force field parameter file (.prm) into an OpenMM force field XML parameter file. This force field should be functionally the same, just existing in two different engines.

The OpenMM force field is an open source force field that executes simulations using Python scripts.[159,160] Because OpenMM uses Python, it is easy for users to customize force field settings, applications, and parameters.[161] OpenMM's Python core also grants it an extensibility advantage over other force fields. New features are a lot easier to add, so much so that users can participate. In this project, OpenMM is used primarily to get around limitations of the Tinker engine. OpenMM can be used for this purpose because it is capable of recreating the AMOEBA force field outside of Tinker.

The two versions of that same force field are referred to as AMOEBA-OpenMM and AMOEBA-Tinker in this project. OpenMM does have constrained optimization and FB is capable of optimizing the parameters of AMOEBA-OpenMM and AMOEBA-Tinker as if it were a single force field.[133] Depending on the step, FB uses the force field engine best suited for that type of calculation. The correctly updated parameters (in their correct units) are recorded for both versions of the force field.

3 .4 Troubleshooting

FB is capable of carrying forth the optimization of force field parameters simulataneously with AMOEBA-Tinker and AMOEBA-OpenMM. Each force field engine has advantages and shortcomings when attempting to optimize the force field parameters. For example, Tinker cannot run geometry-constrained optimizations, so steps where those optimizations are required are run using OpenMM. Tinker is also better suited for certain calculations than OpenMM which is why OpenMM cannot be used exclusively. Also, Tinker offers a force field parameter initialization routine for new molecules.[136] FB navigates both engines' nonoverlapping features by using OpenMM for potential energies and forces and Tinker for binding energies.[133] The newly refined parameters are updated for both force fields. Because FB relies on both force field engines, are must be taken to ensure that the AMOEBA-Tinker and AMOEBA-OpenMM are equal to each other with high precision.

The newly-parameterized AMOEBA-Tinker force field is converted into an AMOEBA-

OpenMM force field with a Python code. Peter Eastman, who works on the OpenMM project, kindly shared code with us that converts Tinker force fields into OpenMM force fields. That conversion code only needed some minor modifications until it was ready for this use case. Tinker force fields are stored in a list structure where each row containing a parameter first contains the keyword, then any necessary atom types, atom classes, and parameter values. OpenMM force fields are stored in XML files which have an element tree structure. The conversion code takes into account formatting differences, as well unit conversions.

Before beginning any parameter optimization, FB is used to verify the sameness of the AMOEBA-Tinker and AMOEBA-OpenMM force fields. To do this, each force field is set up as a target and FB computes the difference between what the force field computes and a QM reference. The reference-minus-target values are required be equivalent to decent precision before beginning parameter optimization. Attempting to use nonequivalent force fields when optimizing parameters with force balance would ruin the process. Each time FB switches engines, it would be using different inputs and would make each step force field dependant. This is why such care is taken to ensure AMOEBA-Tinker and AMEOBA-OpenMM are truly equivalent before beginning optimization.

Unfortunately, early attempts to initialize a Tinker force field and convert it to an exact replica in OpenMM were not successful. The force fields were not as similar as expected when comparing their reference-minus-target values obtained by FB. When FB is using both Tinker and OpenMM in its optimization process, it can execute a process to indirectly compare the force field energies. FB takes an input MD trajectory and can compare multiple targets to QM (or other) benchmarks. For this project, to verify the sameness of the two force fields, FB computes a QM energy for each frame in a 10,000 step MD simulation of urea. Next, FB computes the energy of Tinker, and subtracts it those frame-by-frame from the QM energy then does that for OpenMM. Those energy differences are then averaged to a final result.

This comparison of QM-minus-MM using FB was applied to AMOEBA-Tinker-urea in Listing 1 and AMOEBA-OpenMM-urea in SI Listing B 4. After much troubleshooting, we reduced the QM-minus-MM differences between AMOEBA-Tinker-urea and AMOEBA-OpenMM-urea down to about 0.1 kJ/mol which was a 0.2 percent difference. This difference (of differences) seems small, but it was noticeably larger than previous studies and spurred on other troubleshooting techniques.

An early suspicion we had was that something unusual could be happening to the geometry of the urea molecule over the course of the MD simulation which was disrupting how one engine handled the force field, but not the other. To check this, I calculated the total energy difference between AMOEBA-Tinker-urea and AMOEBA-OpenMM-urea for each of the 10,000 MD frames. Next, I created an average distance matrix for the top 100 "most similar" frames and also one for the top 100 "most different" frames. However, neither matrix appeared drastically different than the 0th frame distance matrix. To make the subtle distances stand out better, I then subtracted the 0th frame's matrix from each frame of interest before averaging them in their respective groups. Subtracting out the 0th frame changes the average matrices from showcasing the distances between atom pairs, to instead revealing how much the average of each grouping differs from the starting geometry.

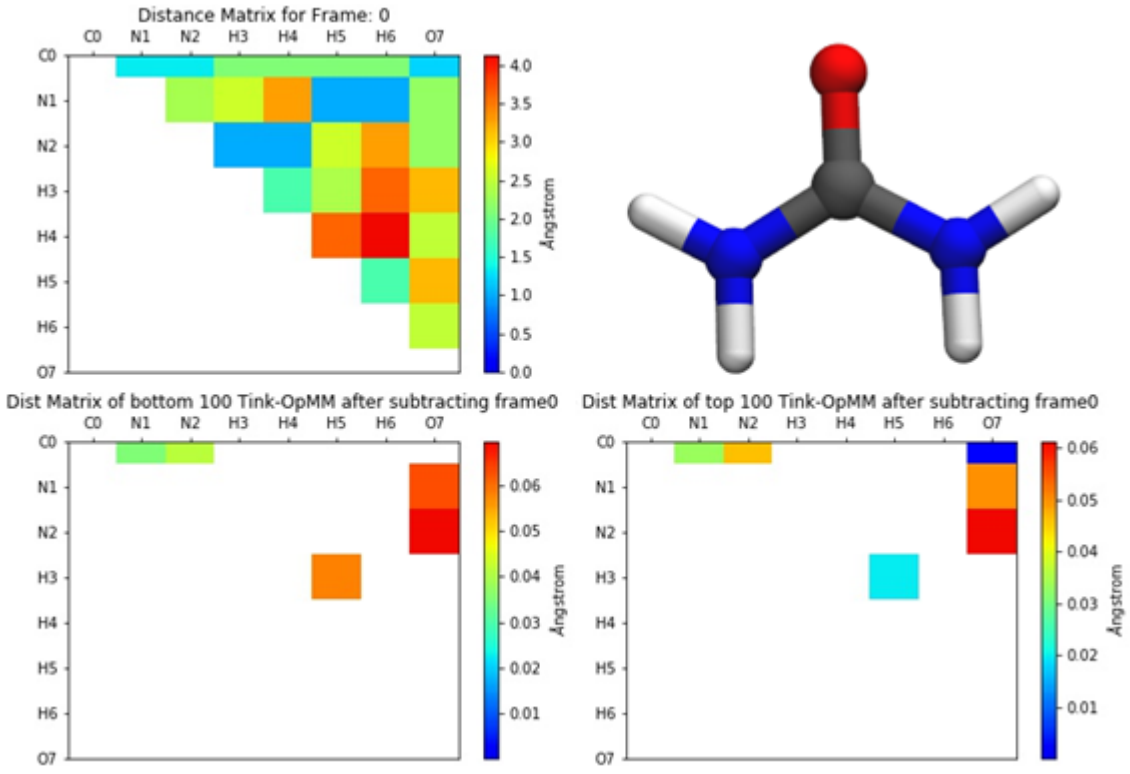

Figure 11: Upper left is the distance matrix for the 0th frame in the trial MD simulation for urea. Lower left is an average distance matrix of the 100 least energetically different MD frames between AMOEBA-Tinker and AMOEBA-OpenMM. Before averaging, each frame had the 0th difference matrix subtracted (to highlight deviation from the planar starting point). Lower right is the same as lower left, except using the 100 most energetically different MD frames. All distances in Ångstrom, redder color indicates a larger distance.

Ultimately, Figure 11 reveals that extreme geometry is not the culprit. Both the top 100 "most different" frames and the top 100 "most similar" frames experience geometric deviations from similar atom pairs and with similar magnitudes. The urea molecule only has so much flexibility at this level of energy and is constrained by the stiff C-N bonds which participate in the pi-system of the C-O bond. With extreme geometry being ruled out, it meant a different approach was necessary to diagnose the stark energy differences between AMOEBA-Tinker-urea and AMOEBA-OpenMM-urea.

Another Python code was created to attempt to unearth the reason behind the differences. That code computes and plots the differences between energy components (bond, angle, VDW, etc.) of Tinker and OpenMM. It is clear from the total energy difference plot in

Figure 12 that the significant differences arise when comparing certain frames (MD steps).

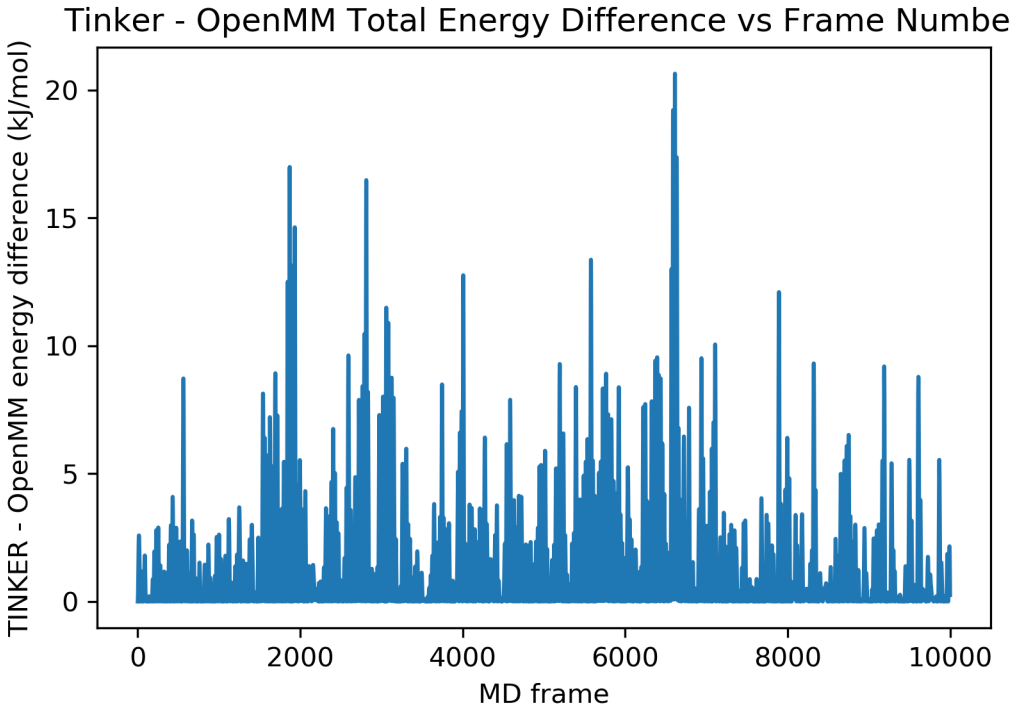

Figure 12: Troubleshooting plot with Tinker minus OpenMM total energy in kJ/mol vs MD frame

Figure 12 reveals energy differences that are far too great for force fields that are supposedly equal. The maximum differences between Tinker and OpenMM total energies eclipsed 20 kJ/mol, with a percent difference of up to 11%. The next step would be to check energy differences between Tinker and OpenMM across each of the energy components. Energies are used instead of forces or another metric because energy components are the output of Tinker's native diagnostic tools.

During the process of creating Figure 13 it became clear that the out-of-plane bonding energy between Tinker and OpenMM was in disagreement. This was difficult to uncover, because in early iterations of the Tinker force field the out-of-plane parameters were excluded and the capability of converting them to OpenMM was commented out of the conversion code. That, and the fact that the early, oft-referenced frames experienced no out-of-plane

bending hid the issue for a time. Once this issue was identified, the capability of converting out-of-plane was reinstated to the conversion code and the original urea Tinker force field was re-converted to OpenMM. After this fix, the out-of-plane energies were confirmed to match, but now the angle energies no longer did.

Figure 13 shows how the issue with the angle parameters had something to do with the in-plane-angle energies:

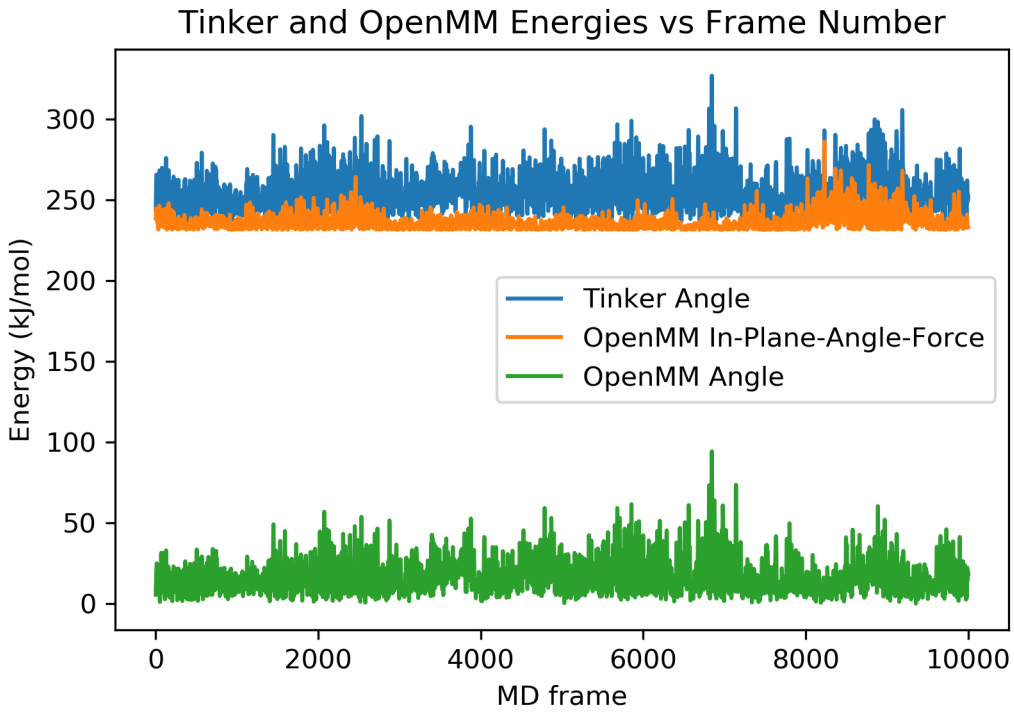

Figure 13: Tinker and OpenMM angle energy components in kJ/mol vs MD frame. The Tinker angle energy is in blue, and the OpenMM angle energy is split between angle (green) and in-plane-angle-force (orange).

Figure 13 reveals that OpenMM is deviating from the expected behavior. Neither force field file has any in-plane-angle parameters. The in-plane-angle energies appeared because OpenMM has some hidden mechanism where it adjusts how the angles are handled in the system. OpenMM was not executing the force field according to the parameter file and adapting angle parameters into in-plane-angle parameters and changing the overall energy.

Further verification revealed that subtracting the OpenMM in-plane-angle and angle values point-by-point from the Tinker angle values recreates the line in Figure 12.

This issue with OpenMM's built-in protocols for separating angle parameters into traditional angle-bending and in-plane-angle could be fixed by modifying OpenMM. The solution we found to this was to adjust the Python script used for testing OpenMM energies to remove the automatically generated out-of-plane-bending energies and re-insert them as traditional angle energies. The following code in Listing 2 is an OpenMM Python script containing the angle energy corrections. The correction section is found near the end of the file and is highlighted with repeated # symbols.

```
from simtk.openmm.app import *
from simtk.openmm import *
from simtk.unit import *
import simtk.unit as u
from forcebalance.nifty import printcool_dictionary
from forcebalance.molecule import Molecule
from collections import OrderedDict
import IPython

def energy_components(Sim):
    # Before using EnergyComponents, make sure each Force is set to a
   different group.
    EnergyTerms = OrderedDict()
    for i in range(Sim.system.getNumForces()):
        EnergyTerms[Sim.system.getForce(i).__class__.__name__] = Sim.
   context.getState(getEnergy=True,groups=2**i).getPotentialEnergy() / u.
   kilojoules_per_mole
    EnergyTerms['Potential'] = Sim.context.getState(getEnergy=True).
   getPotentialEnergy() / u.kilojoules_per_mole
    return EnergyTerms


pdb = PDBFile('conf.pdb')
forcefield = ForceField('AMOEBA-urea_NEW.xml') #, 'tip3p.xml')
#unmatched_residues = forcefield.getUnmatchedResidues(pdb.topology)
# use FB molecule object to set positions with xyz file
molecule = Molecule('frame_1913.xyz')
# create an openmm list of Vec3's to update positions
for x in range(len(molecule)):
    # FB molecule object
    xyz = molecule.xyzs[x]
    # openmm uses vec3
    xyz_vec3 = [Vec3(i[0], i[1], i[2]) for i in xyz]*angstrom

# create system
```

```
system = forcefield.createSystem(pdb.topology, nonbondedCutoff=1*nanometer
    )

fs = system.getForces()

#######################################
##### AmoebaAngleForce CORRECTION #####

# determine index of AmoebaInPlaneAngleForce, which changes for each ff
for i in range(len(fs)):
    print(fs[i])
    if "AmoebaInPlaneAngleForce" in str(fs[i]):
        ipa_index = i
    # determine index of AmoebaAngleForce, which also changes for each ff
    elif "AmoebaAngleForce" in str(fs[i]):
        angle_index = i

# use the correct index to access AmoebaInPlaneAngleForce
print(fs[ipa_index])
print("number of in-plane angles", fs[ipa_index].getNumAngles())
numinplane = fs[ipa_index].getNumAngles()
for i in range(numinplane):
    print(fs[ipa_index].getAngleParameters(i))


# using the correct index for AmoebaAngleForce,
# reinsert AmoebaInPlaneAngleForce as an AmoebaAngleForce
# example AmoebaInPlaneAngleForce: [1, 0, 2, 7, Quantity(value=152.31,
    unit=radian), Quantity(value=0.0235786067612, unit=kilojoule/(mole*
    radian**2))]
# example AmoebaAngleForce: [1, 0, 2, Quantity(value=152.31, unit=degree),
     Quantity(value=0.0235786067612, unit=kilojoule/(mole*radian**2))]
print(fs[angle_index])
for i in range(numinplane):
    fs[angle_index].addAngle(fs[ipa_index].getAngleParameters(i)[0], fs[
    ipa_index].getAngleParameters(i)[1], fs[ipa_index].getAngleParameters(i
    )[2], float(str(fs[ipa_index].getAngleParameters(i)[4]).split(' ')[0]),
     float(str(fs[ipa_index].getAngleParameters(i)[5]).split(' ')[0]))

print("AmoebaInPlaneAngleForces reinserted as AmoebaAngleForce")
# remove in-plane angle force
system.removeForce(ipa_index)
print("AmoebaInPlaneAngleForces removed from the system")

##### END OF CORRECTION SECTION #####
#####################################

# give each force group its own index
for ind, f in enumerate(system.getForces()):
    f.setForceGroup(ind)
integrator = LangevinIntegrator(300*kelvin, 1/picosecond, 0.002*
    picoseconds)
simulation = Simulation(pdb.topology, system, integrator)
#simulation.context.setPositions(pdb.positions)
```

```
simulation.context.setPositions(xyz_vec3)
# create force component dictionary using above function
Ecomps_OMM = energy_components(simulation)
# print out force component dictionary
printcool_dictionary(Ecomps_OMM, title="OpenMM energy components")
```

Listing 2: Sample OpenMM Python script containing a correction that removes AmoebaInPlaneAngleForce inclusions and reinserts them as AmoebaAngleForce instances. Subsection containing the correction noted with repeated # symbols.

This solution in Listing 2 produced sufficiently equivalent force fields across both Tinker and OpenMM as seen with the post-fix total energy differences in Figure 14.

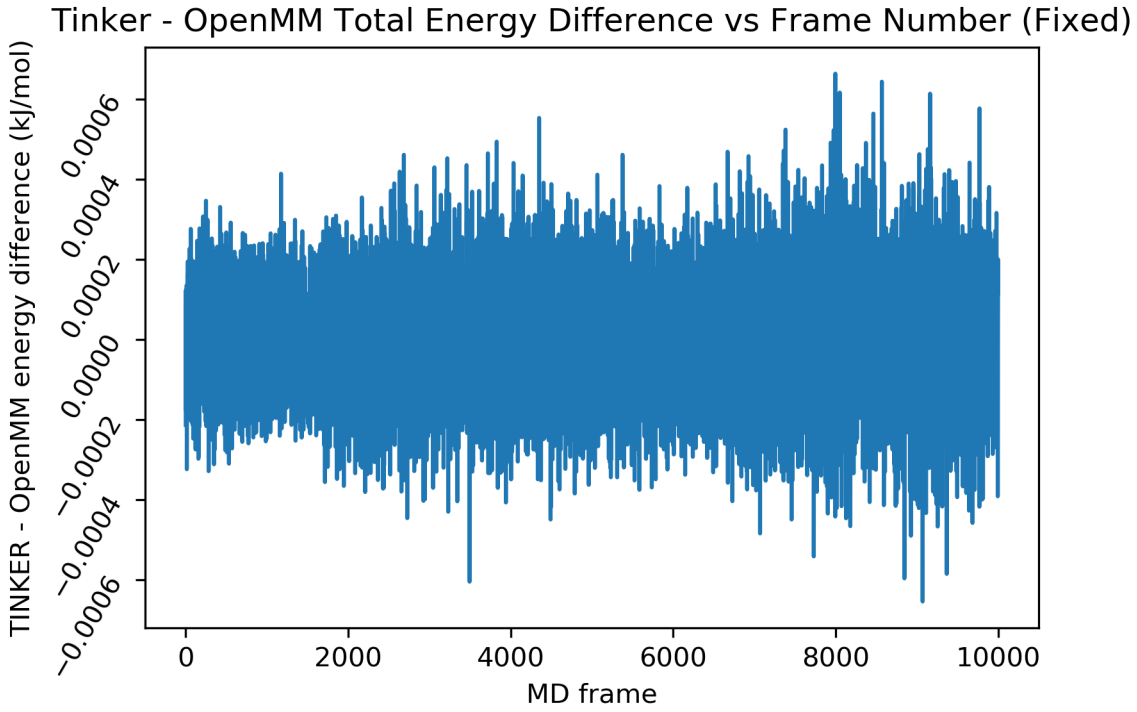

Figure 14: Tinker minus OpenMM total energy in kJ/mol vs MD frame after applying the OpenMM fix in Listing 2. The magnitude of the energy differences matches the expected behavior of the two force fields.

The total energy differences in post-fix Figure 14 are much smaller than in Figure 12. After the fix, the maximum energy differences are around ± 0.0006 kJ/mol with a percent difference of 0.0003%. This is an acceptable energy difference between AMOEBA-Tinker-urea and AMOEBA-OpenMM-urea for later FB usage. Figure 15 contains a breakdown of

each of the contributing individual energy components:

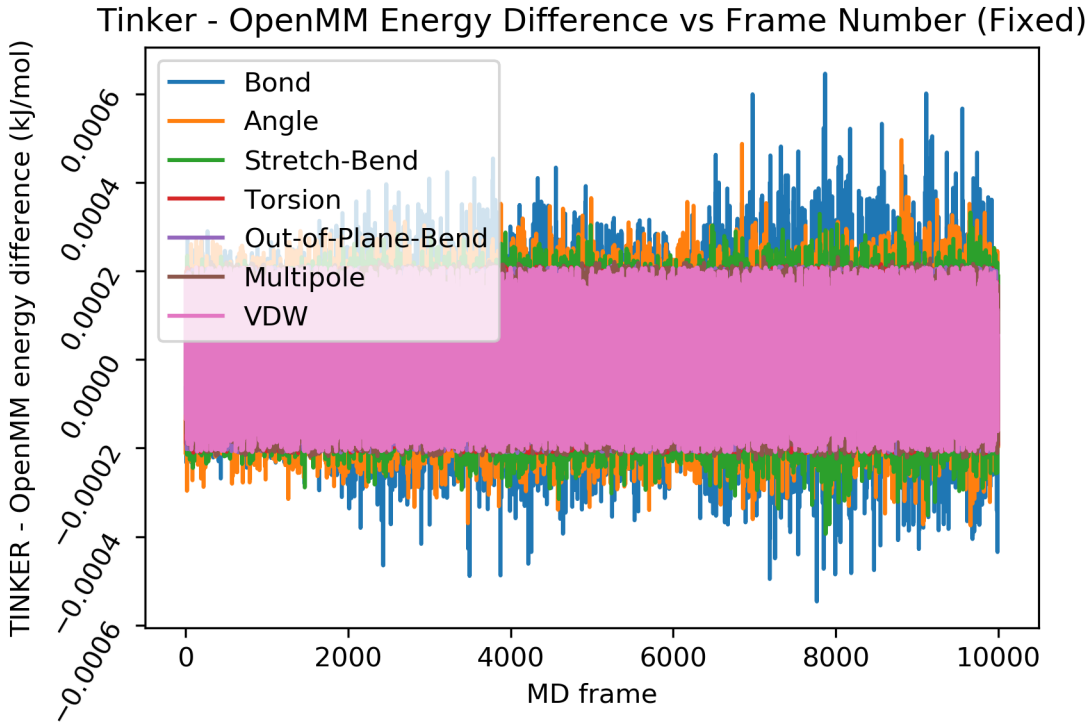

Figure 15: Troubleshooting plot (after applying the OpenMM fix in Listing 2) where the y-axis is the value in kJ/mol of the OpenMM total energy subtracted from the Tinker total energy for each MD frame (x-axis).

Figure 15 shows that post-fix each energy component is effectively the same across both force fields with differences at most ± 0.0006 kJ/mol. The angle energy component differences are corrected from how they appeared as raw energy values in Figure 13.

3 .5 Conclusion and Future Work

The conclusions are fairly interwoven with the future work for this project, so those thoughts will be combined here. The next step is to implement the angle correction into how FB applies the OpenMM force field. One possible way to add this fix into FB is by inserting a Python script into AMOEBA-OpenMM. OpenMM force field files support Python, so it should just be a matter of how to call the OpenMM system that FB creates and update

the angle parameters. Another idea is to try and activate the in-plane-angle behavior in AMOEBA-Tinker and leave AMOEBA-OpenMM unchanged.

With a semi-automated means of initializing Tinker force fields, a code to convert them to OpenMM, and a solution to bring AMOEBA in Tinker and OpenMM into equivalence, FB can soon be used to tune these force field parameters.

This project for the most part was mired in constant troubleshooting. It was only near the end that enough significant hurdles were overcome to successfully generate both the AMOEBA-Tinker and AMOEBA-OpenMM force fields that were sufficiently equivalent for FB to optimize their parameters in unison. Therefore, the future work is to use FB to optimize these DES hydrogen bond-donor molecules and compare features such as radial distribution function, density, and other factors to experimental data and alternative force fields. To complete the DES polarizable force field a solution also needs to be found to the ion problem found when attempting to initialize choline chloride. Once both choline chloride and the hydrogen bond donor force field parameters are initialized and converted into AMOEBA-OpenMM, they can be combined into a DES and FB can be used to refine their parameters.

A Supporting Information for Chapter 2: Bond-Order Time Series Analysis for Detecting Reaction Events in *Ab Initio* Molecular Dynamics Simulations

Marshall Hutchings[†], Johnson Liu[†], Yudong Qiu[†], Chenchen Song[‡], Lee-Ping Wang[*†]

†Department of Chemistry, University of California; 1 Shields Ave; Davis, CA 95616.

‡Department of Chemistry, Stanford University; Stanford, CA 94305.

¶SLAC National Accelerator Laboratory; Menlo Park, CA 94025.

leeping@ucdavis.edu

A .1 Clustering

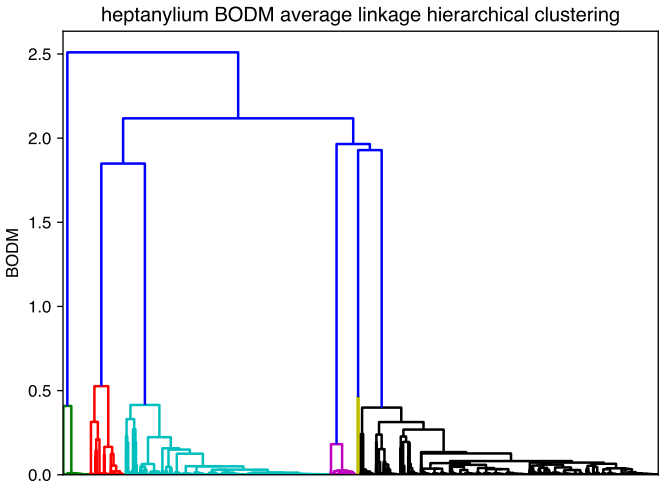

Figure A .1: Dendrogram for the heptanylium system using BODM (Equation 3) as a clustering threshold. Clustering cutoff of 1.0 was selected to create this set of reference reaction events.

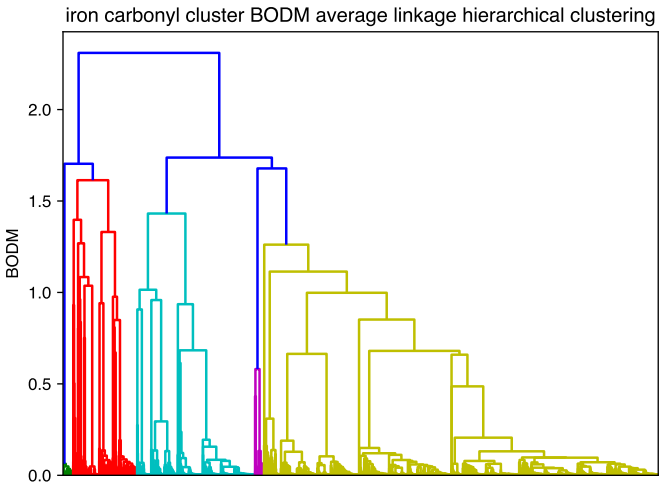

Figure A .2: Dendrogram for the iron carbonyl cluster system using BODM (Equation 3) as a clustering threshold. Clustering cutoff of 1.0 was selected to create the set of reference reaction events.

A .2 Bond-wise Objective Function

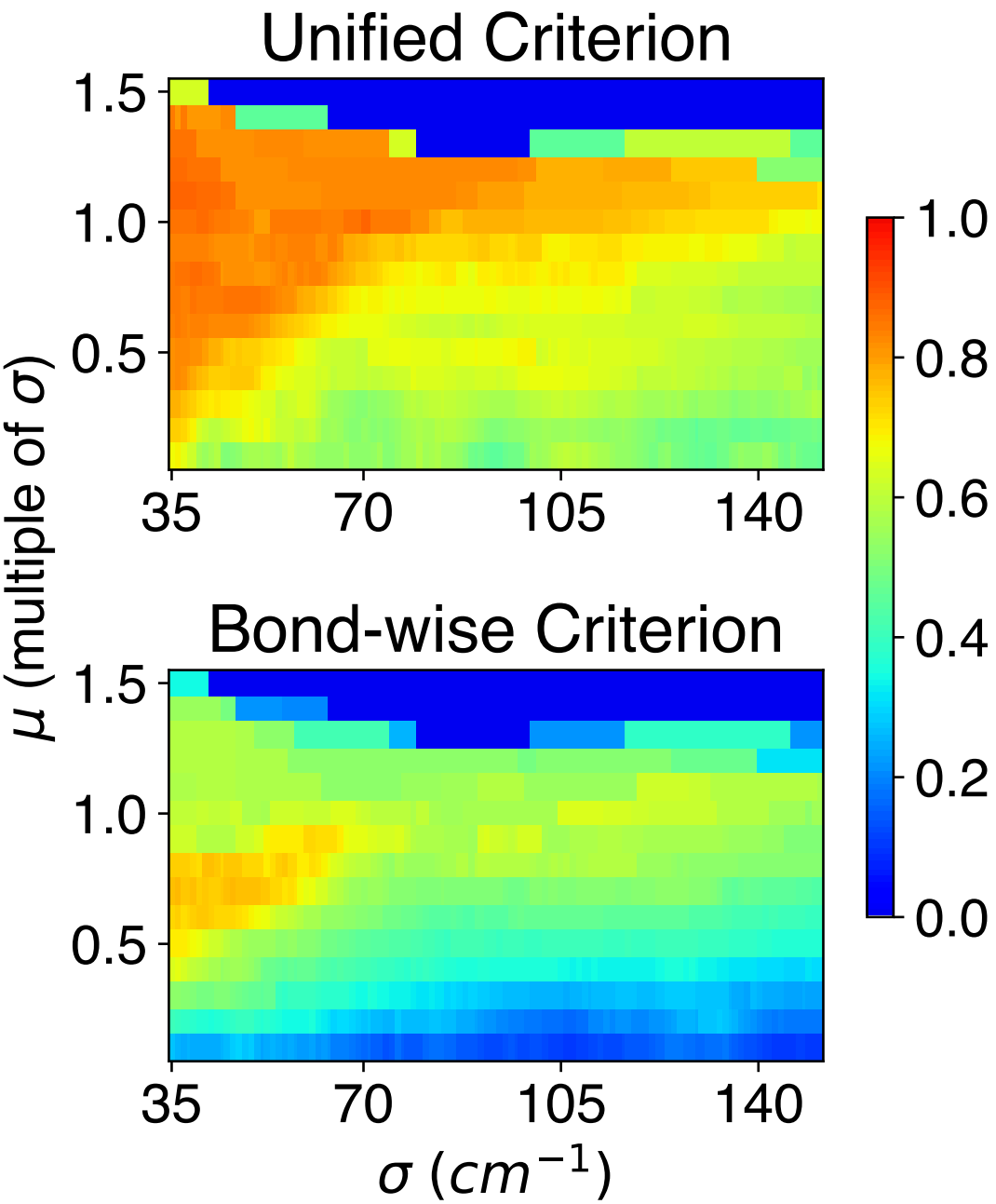

Figure A .3: Top: Heat map for the iron carbonyl cluster system with σ and μ combinations using the 'unified' criterion objective function. Bottom: Heat map of the same system, but with σ and μ combinations using the 'bond-wise' criterion objective function.

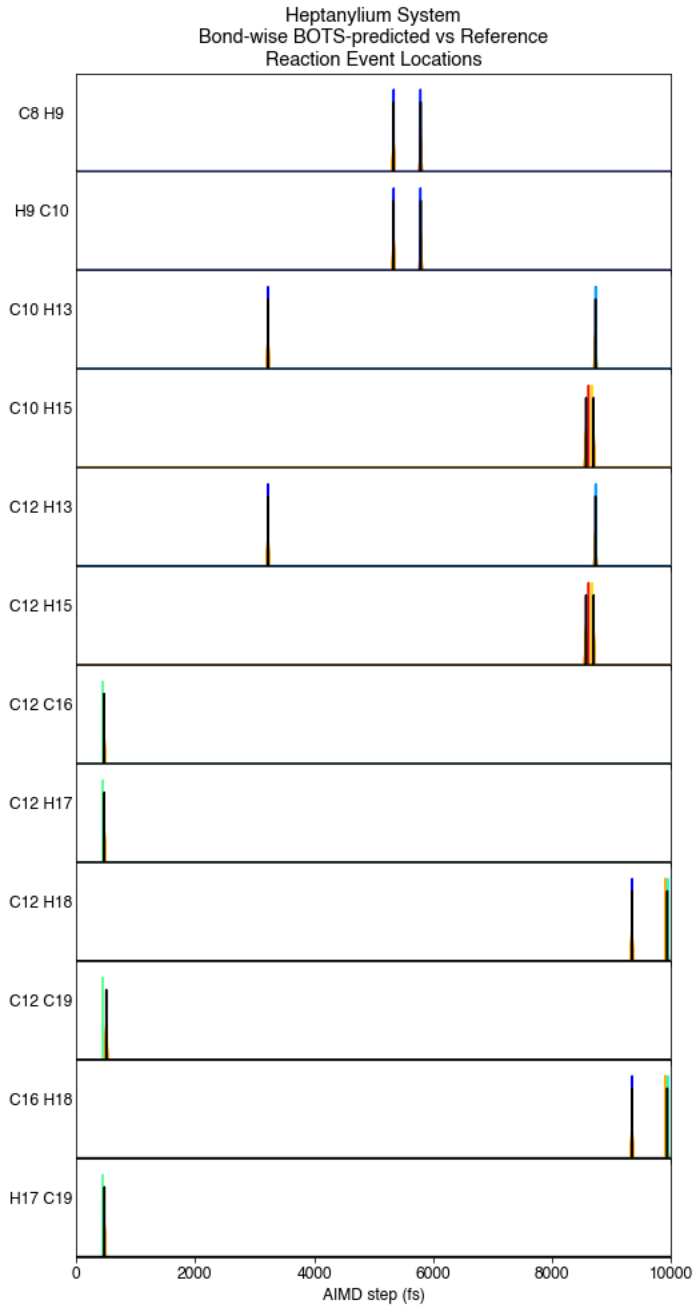

Figure A .4: Reference reaction events (colors from Figure 7) and BO time series predictions (black) for the heptanylium system. Orange time windows expand around predictions with height inversely proportional to window size.

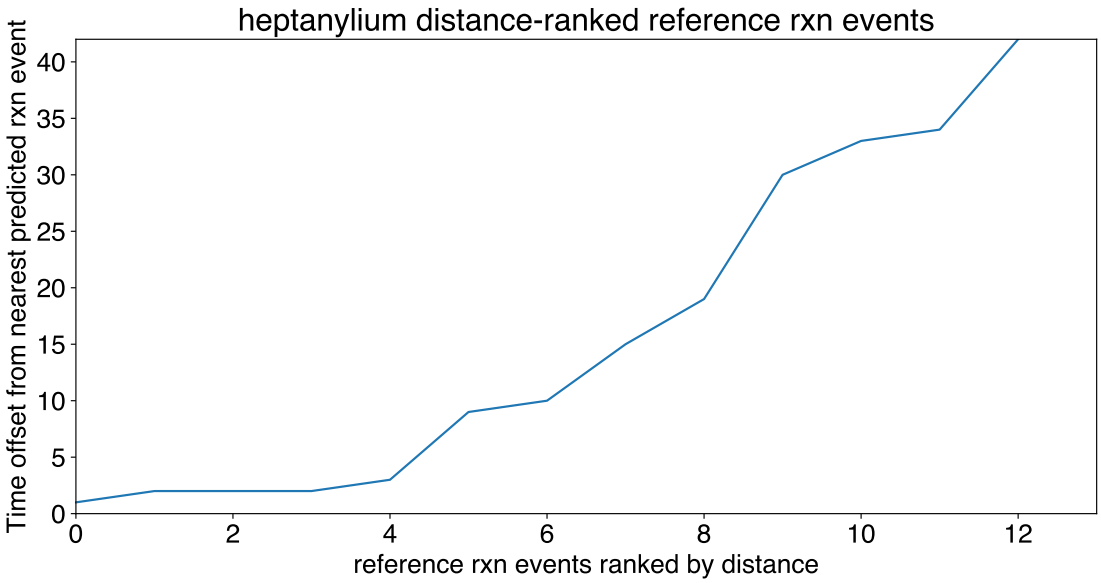

Figure A .5: Reference reaction events ranked by temporal distance from nearest predicted reaction event for the heptanylium system.

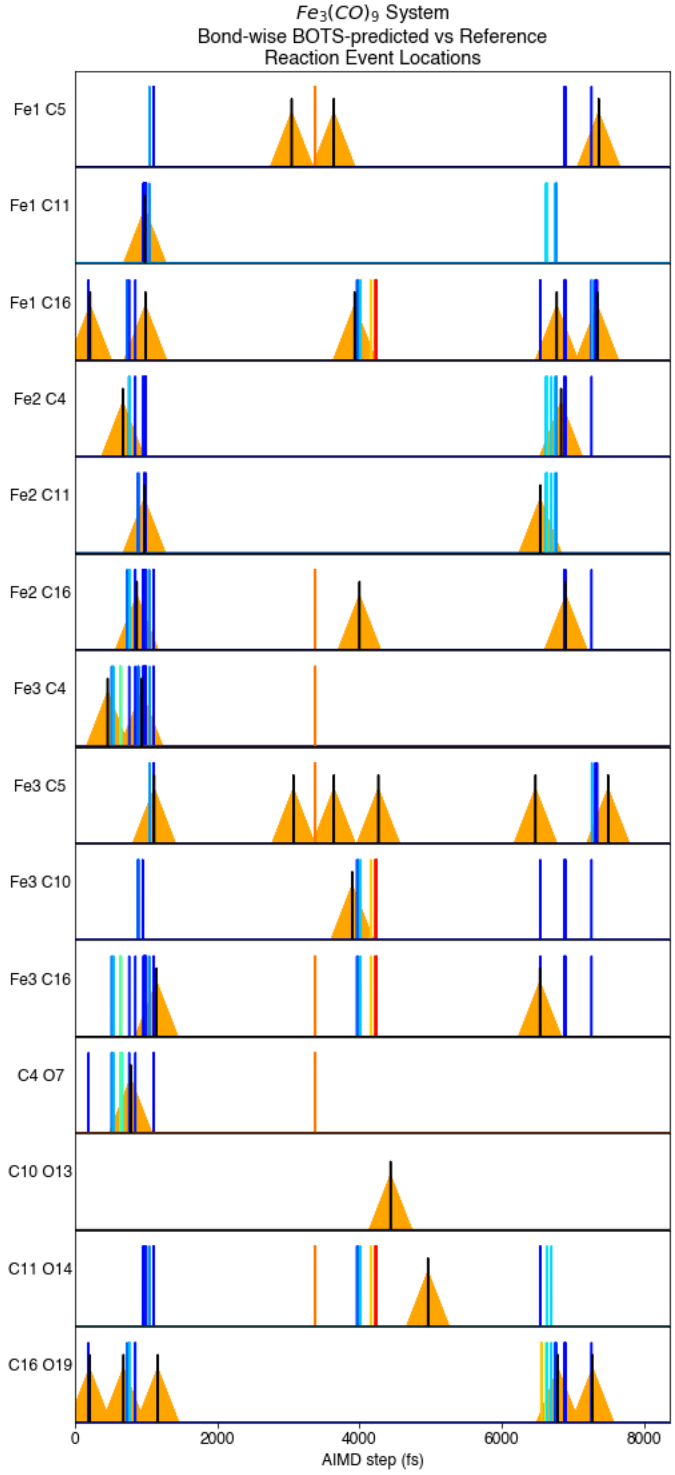

Figure A .6: Reference reaction events (colors from Figure 9) and BO time series predictions (black) for the iron carbonyl cluster system. Orange time windows expand around predictions with height inversely proportional to window size.

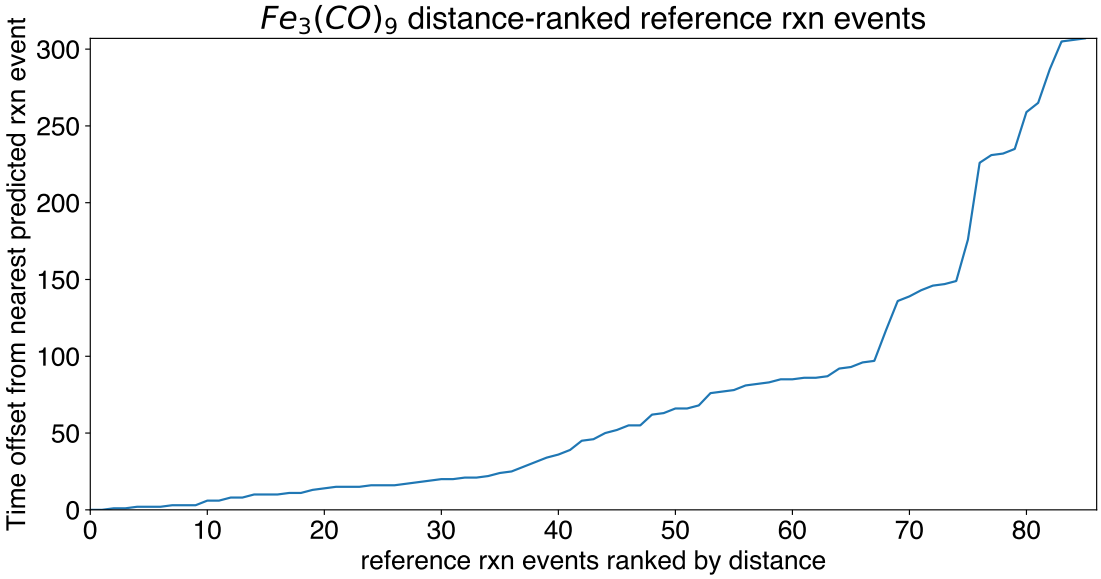

Figure A .7: Reference reaction events ranked by temporal distance from nearest predicted reaction event for the iron carbonyl cluster system.

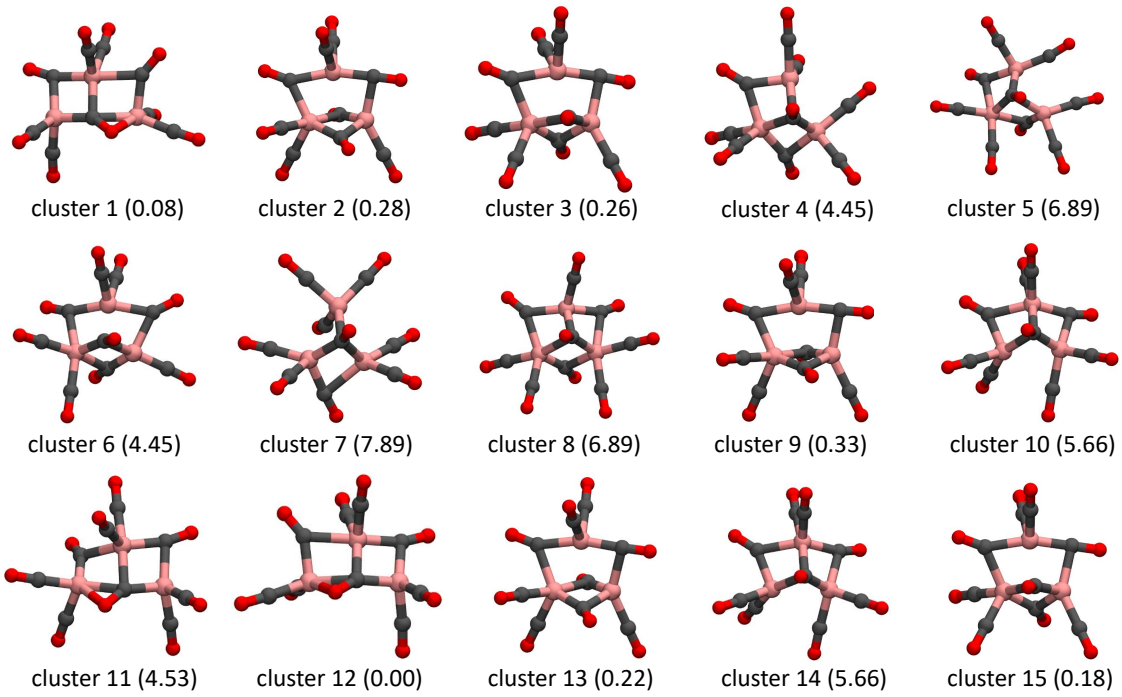

Figure A .8: The 15 clusters identified when creating the $\mathcal{E}_{ref}$ for the iron carbonyl cluster $Fe_3(CO)_9$ with relative energies in kcal/mol shown in parentheses. Molecule color scheme: iron: pink, carbon: gray, and oxygen: red. Bonds drawn using a 2.4 Å cutoff. Some clusters are very similar because permutation of atomic indices is not accounted for in this method.

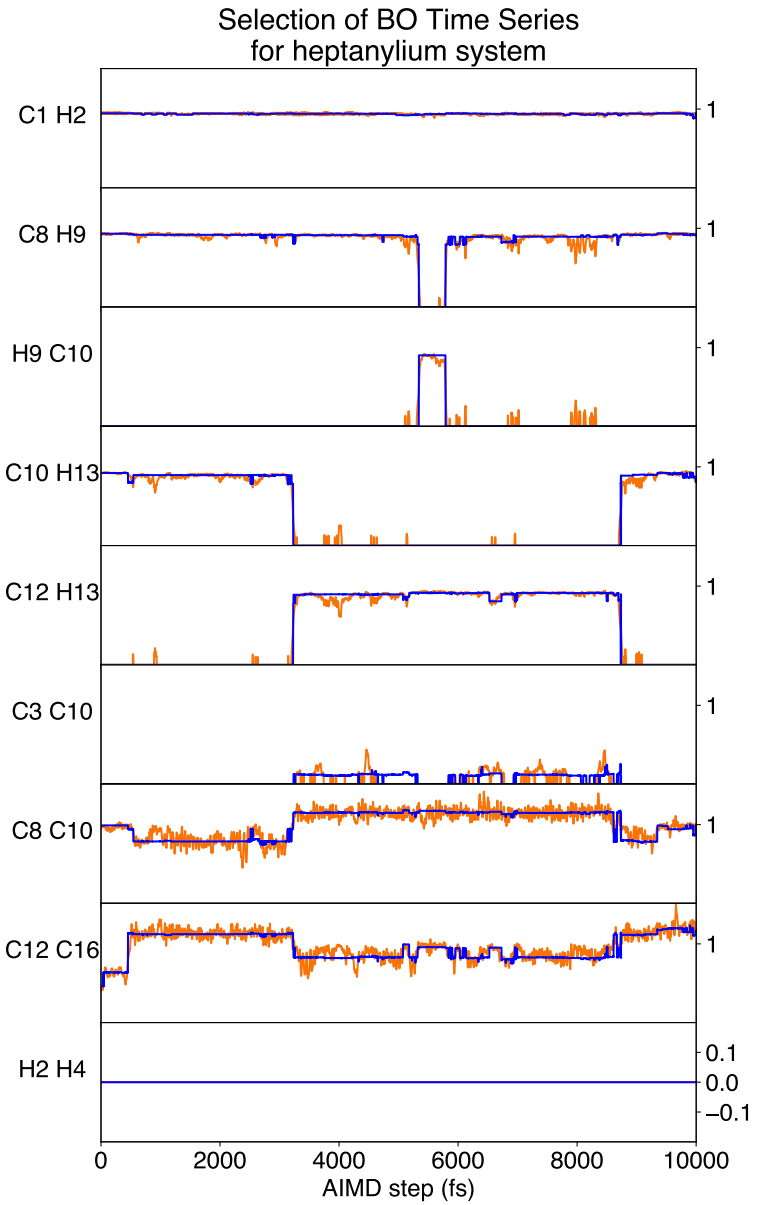

Figure A .9: Selected bond order (orange) and energy-minimized bond order (blue) time series for heptanylium trajectory.

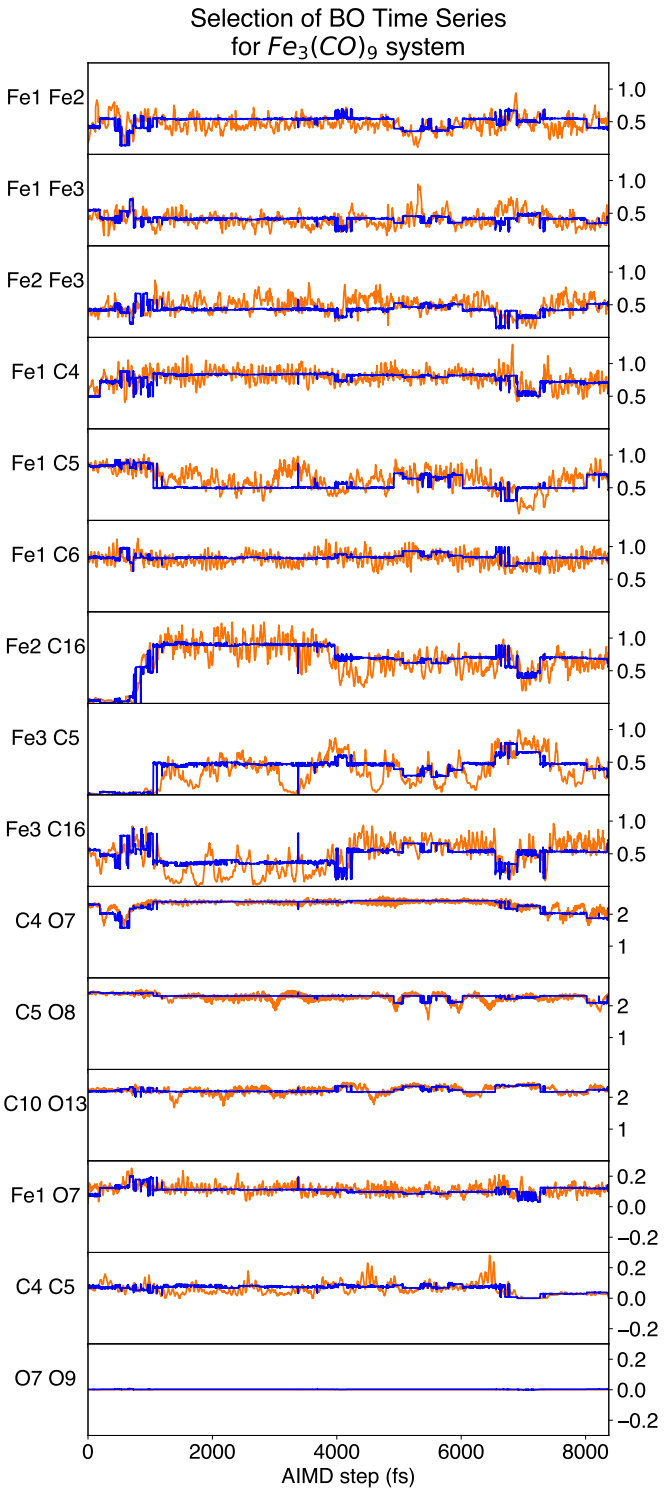

Figure A .10: Selected bond order (orange) and energy-minimized bond order (blue) time series for $Fe_3(CO)_9$ trajectory.

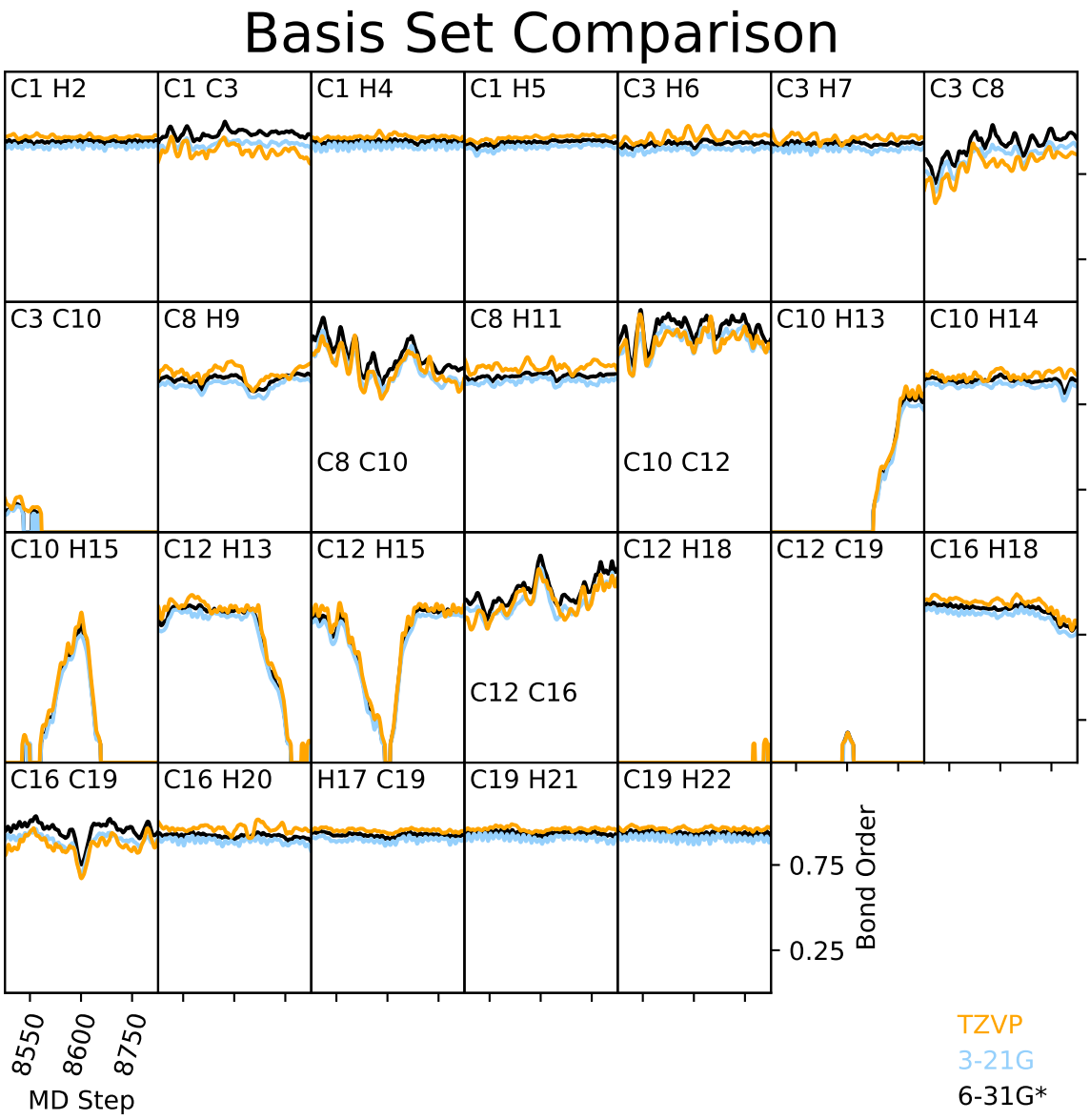

Figure A .11: Bond order time series comparison for trajectory segment 8500-8800 fs of heptanylium cation ($C_7H_{15}^+$) using the B3LYP method and the 6-31G*, 3-21G, and TZVP basis sets.

RMS error between basis sets			
atom pairs	6-31G* vs 3-21G	6-31G* vs TZVP	3-21G vs TZVP
C1 H2	0.029	0.028	0.056
C1 C3	0.062	0.102	0.045
C1 H4	0.030	0.029	0.058
C1 H5	0.029	0.028	0.058
C3 H6	0.034	0.046	0.078
C3 H7	0.033	0.036	0.069
C3 C8	0.066	**0.116**	0.055
C3 C10	0.019	0.035	0.044
C8 H9	0.035	0.045	0.078
C8 C10	0.074	0.076	0.035
C8 H11	0.033	0.059	**0.091**
C10 C12	**0.075**	0.075	0.041
C10 H13	0.020	0.019	0.038
C10 H14	0.031	0.040	0.070
C10 H15	0.021	0.030	0.046
C12 H13	0.030	0.043	0.069
C12 H15	0.030	0.033	0.060
C12 C16	0.072	0.075	0.043
C12 H18	0.000	0.020	0.020
C12 C19	0.009	0.006	0.011
C16 H18	0.036	0.038	0.073
C16 C19	0.066	0.093	0.037
C16 H20	0.032	0.046	0.076
H17 C19	0.030	0.027	0.057
C19 H21	0.030	0.025	0.055
C19 H22	0.030	0.029	0.058

Table A .1: RMS error values between the basis sets 6-31G*, 3-21G, and TZVP in the trajectory segment 8500-8800 fs of heptanylium cation ($C_7H_{15}^+$) . Corresponding time series in Figure A .11. Largest RMS error for each pair of bases shown in bold.

B Supporting Information for Chapter 3: AMOEBA Polarizable Force Fields for Deep Eutectic Solvents

```
import numpy as np
import argparse
import os
from os import path
import sys
import subprocess

def make_gauss_input(name, coord_name, option, charge=0, spin=1):
    ''' option #1 : create a Gaussian .com input file to run initial optimization
    option #2 : create a Gaussian .com input file to run second optimization
    option #3 : create a Gaussian .com input file to run frequency calculation
    '''
    coords_block = get_coordinates(coord_name)
    #coords_list = list_splitter(coords_block)
    #file_name = '%s.com'
    if option == 1:
        filename = '%s.com'%name
    elif option == 2:
        filename = '%s_re-opt.com'%name
    elif option == 3:
        filename = '%s_freq.com'%name
    with open(filename, 'w') as outfile:
        #outfile.write('%' + 'Chk=%s.chk\n'%name)
        if option == 1:
            outfile.write('%' + 'Chk=%s.chk\n'%name)
            outfile.write('# MP2/6-311G(1d,1p) Opt Density=MP2\n\n')
        elif option == 2:
            outfile.write('%' + 'Chk=%s_re-opt.chk\n'%name)
            outfile.write('# opt mp2/aug-cc-pVTZ density=mp2\n\n')
        elif option == 3:
            outfile.write('%' + 'Chk=%s_freq.chk\n'%name)
            outfile.write('# freq MP2/6-311G(1d,1p) density=mp2\n\n')

        outfile.write('remark line\n\n')
        outfile.write('%s %s\n'%(charge, spin))
        for i in range(2, len(coords_block)):
            outfile.write(coords_block[i])
        outfile.write('\n')

def make_gdma_input(name):
    ''' creates a GDMA input file
    '''
    with open('%s.gdmain'%name, 'w') as outfile:
        outfile.write('Title \"%s at MP2/6-311G(1d,1p)\"\n\n'%name)
```

```
45          outfile.write('Verbose\n')
46          outfile.write('Density MP2\n')
47          outfile.write('File %s.fchk\n\n'%name)
48          outfile.write('Angstrom\n')
49          outfile.write('AU\n\n')
50          outfile.write('Multipoles\n')
51          outfile.write('Switch 0\n')
52          outfile.write('Radius H 0.65\n')
53          outfile.write('Limit 2\n')
54          outfile.write('Start\n\n')
55          outfile.write('Finish')
56
57  def make_poledit_input(name):
58      ''' creates a TINKER poledit input file called input_poledit.txt
59      as a reference of the inputs into that program. This program often
        does not
60      identify equivalent atoms, and it is recommended that the user reviews
        the output
61      parameter file name_new.key (using TINKER poledit) before moving on
62      '''
63      with open('input_poledit.txt', 'w') as outfile:
64          outfile.write('1\n')
65          outfile.write('%s.gdmaout\n'%name)
66          # need this symbolic link to run everything
67          outfile.write('amoeba09.prm\n\n')
68          outfile.write('A\n\n\n')
69          outfile.write('Y\n\n')
70
71  def make_prmedit_input(name, starting_atom_type):
72      ''' creates a TINKER prmedit input file called input_prmedit.txt
73      prmedit renumbers atom type indices for us. It is also capable of
74      renumbering atom class indices, but they are missing from the .key
75      file at the point in time this is called. The function
        insert_atom_class_indices()
76      inserts the atom class indices (correctly ordered).
77      '''
78      with open('input_prmedit.txt', 'w') as outfile:
79          outfile.write('%s.key\n'%name)
80          # option #3 is to edit atom types
81          outfile.write('3\n')
82          outfile.write('%s\n'%starting_atom_type)
83          #outfile.write('%i'%starting_atom_class)
84
85  def insert_atom_class_indices(name, start_index):
86      ''' this function does two main things: it copies the parameters
87      from parameter.prm into name_new.key AND it inserts atom class
88      indices starting from the input start_index.
89      '''
90      start_index = int(start_index)
91      filename = "parameter.prm"
92      with open(filename) as fileblock:
93          filelines = fileblock.readlines()
94          # split lines by spaces
95          filepieces = list_splitter(filelines)
```

```
        with open(name + "_new.key", 'w') as edited:
            for i in range(len(filepieces)):
                if i == 0:
                    print("first line", filepieces[i])
                if len(filepieces[i]) != 0:
                    if filepieces[i][0] == 'atom':
                        filepieces[i].insert(2, str(start_index))
                        start_index += 1
                        editline = ""
                        for x in filepieces[i]:
                            editline += x + "    "
                        edited.write(editline + "\n")
                    else:
                        edited.write(filelines[i])


def edit_indices(filename, type_index, class_index):
    ''' not currently in use, but was used a starting point
    for some other functions
    '''
    """
    with open(filename) as fileblock:
        filelines = fileblock.readlines()
    # split lines by spaces
    filepieces = list_splitter(filelines)
    with open("new" + filename):
        for i in range(len(filepieces)):
            if i == 0:
                print("first line", filepieces[i])
#            if filepieces[i] == "atom":
    """
    with open(filename) as fileblock:
        filelines = fileblock.readlines()
        # split lines by spaces
        filepieces = list_splitter(filelines)
        with open("new" + filename, 'w') as edited:
            for i in range(len(filepieces)):
                if i == 0:
                    print("first line", filepieces[i])
                if filepieces[i][0] == 'atom':
                    newline = filepieces[i]
                    newline[1] = str(type_index + i)
                    newline[2] = str(class_index + i)
                    print("edited line %i"%i, newline)
                    editline = ""
                    for j in newline:
                        editline += j
                    edited.write(editline)
                else:
                    edited.write(filelines[i])

def make_potential_input(name, option):
    ''' makes input files for TINKER potential function calls (information
    from TINKER program)
```

```
    option #1 : creates an input file for Gaussian CUBEGEN
    option #2 : Obtain a QM Potential from a Gaussian CUBE file
    option #6 : Fits electrostatic potential parameters to a grid
    '''
    if option == 1:
        with open('input_potential_1.txt', 'w') as outfile:
            outfile.write('1\n')
            outfile.write('%s_re-opt.txyz\n'%name)
            outfile.write('%s_new.key'%name)
    elif option == 2:
        with open('input_potential_2.txt', 'w') as outfile:
            outfile.write('2\n')
            outfile.write('%s_re-opt.cube'%name)
    elif option == 6:
        with open('input_potential_6.txt', 'w') as outfile:
            outfile.write('6\n')
            outfile.write('%s_re-opt.txyz\n'%name)
            outfile.write('%s_new.key\n'%name)
            outfile.write('%s_re-opt.pot\n'%name)
            outfile.write('y\n\n')
            outfile.write('%s_new.key'%name)

def make_valence_input(name, option):
    ''' makes input files for TINKER valence function calls (information from TINKER program)
    option #1 : sets intitial values for valence parameters
    option #2 : compare QM and MM vibrational parameters (just for evaluation)
    option #3 : fits force parameters to the QM results
    '''
    if option == 1:
        with open('input_valence_1.txt', 'w') as outfile:
            outfile.write('1\n')
            outfile.write('%s_re-opt.txyz\n'%name)
            outfile.write('%s_re-opt.log\n'%name)
            outfile.write('%s_new_new.key'%name)
    elif option == 2 or option == 3:
        with open('input_valence_%i.txt'%option, 'w') as outfile:
            outfile.write('%i\n'%option)
            outfile.write('%s_re-opt.txyz\n'%name)
            outfile.write('%s_freq.log\n'%name)
            outfile.write('%s_new_new_new.key'%name)
            if option == 3:
                outfile.write('\n\n')

# function for splitting raw line-lists
def list_splitter(raw_list):
    ''' function for splitting raw line-lists into lines of separate elements
    '''
    split_list = []
    # split a list (by spaces) into separate elements
    for i in range(len(raw_list)):
        split_list.append(raw_list[i].split())
```

```
200     return split_list
201
202 def get_coordinates(coord_name):
203     ''' open coordinates file and split into separate lines
204     '''
205     with open(coord_name) as coors_file:
206         coors_file_block = coors_file.readlines()
207     return coors_file_block
208
209 def line_prepender(filename, line):
210     ''' function that inserts a line at the begining of a file
211     '''
212     with open(filename, 'r+') as f:
213         content = f.read()
214         f.seek(0, 0)
215         f.write(line.rstrip('\r\n') + '\n' + content)
216
217 def combine_mp_parameters(old_params, opt_multipoles, name):
218     save_file = "%s_new_new.key"%name
219     # boolean to track copying of new multipole values
220     copied = False
221     with open(old_params) as fileblock1:
222         filelines1 = fileblock1.readlines()
223         # split lines by spaces
224         filepieces1 = list_splitter(filelines1)
225         with open(opt_multipoles) as fileblock2:
226             filelines2 = fileblock2.readlines()
227             # split lines by spaces
228             filepieces2 = list_splitter(filelines2)
229
230             with open(save_file, 'w') as edited:
231                 # iterate over each line of the original (old) parameter
    (.key) file
232                 for i in range(len(filepieces1)):
233                     if i == 0:
234                         print("first line", filepieces1[i])
235                     if len(filepieces1[i]) != 0:
236                         # identify "atom" OR "polarize" OR "FIX-MULTIPOLE"
     lines and copy those
237                         if filepieces1[i][0] == "atom" or filepieces1[i
    ][0] == "polarize" or filepieces1[i][0] == "FIX-MONOPOLE":
238                             edited.write(filelines1[i])
239                         # otherwise, skip the old multipole values and
    copy the new multipole values
240                         elif filepieces1[i][0] == "multipole" and copied
    == False:
241                             # iterate over each line of the optimized
    multipoles file and write them into the edit
242                             for j in range(len(filelines2)):
243                                 edited.write(filelines2[j])
244                             copied = True
245                     # if line empty, include a line return to preserve
    formatting
246                     else:
```

```
                        edited.write("\n")
    print("newest parameter file is %s"%save_file)

def combine_val_parameters(old_params, new_params, name):
    save_file = "%s_new_new_new.key"%name
    # boolean to track copying of new parameter values
    keep_copying = False
    # boolean to skip copying the Urey-Bradley parameters (which are zero for most of these systems)
    copy_urey = False
    with open(old_params) as fileblock1:
        filelines1 = fileblock1.readlines()
        # split lines by spaces
        filepieces1 = list_splitter(filelines1)
        with open(new_params) as fileblock2:
            filelines2 = fileblock2.readlines()
            # split lines by spaces
            filepieces2 = list_splitter(filelines2)

            with open(save_file, 'w') as edited:
                # iterate over and copy each line of the original (old) parameter (.key) file
                for i in range(len(filelines1)):
                    edited.write(filelines1[i])
                #edited.write("\n")
                # iterate over each line in the new parameter file (output_valence_1.txt) and copy the relevant lines
                for j in range(len(filepieces2)):
                    # skip empty lines
                    if len(filepieces2[j]) != 0:
                        # only start copying once the first "Estimated" appears, then copy every line afterward
                        if filepieces2[j][0] == "Estimated":
                            # add line returns after Estimated to aid the formatting
                            #edited.write("\n")
                            keep_copying = True
                        if keep_copying == True:
                            #print(filelines2[j])
                            # skip the ureybrad parameter lines
                            if copy_urey == False and filepieces2[j][0] != "ureybrad":
                                edited.write(filelines2[j])
#                             else:
#                                 edited.write(filelines2[j])
                    # if line empty, include a line return to preserve formatting (only after encountering the first instance of "Estimated"
                    elif keep_copying == True:
                        edited.write("\n")
    print("newest parameter file is %s"%save_file)


def combine_final_parameters(old_params, new_params, name):
    save_file = "%s_final.key"%name
```

```
    # boolean to track copying of new multipole values
    copy_new = False
    # boolean to track copying of old multipole values
    copy_old = True
    with open(old_params) as fileblock1:
        filelines1 = fileblock1.readlines()
        # split lines by spaces
        filepieces1 = list_splitter(filelines1)
        with open(new_params) as fileblock2:
            filelines2 = fileblock2.readlines()
            # split lines by spaces
            filepieces2 = list_splitter(filelines2)

            with open(save_file, 'w') as edited:
                # iterate over each line of the original (old) parameter (.key) file
                for i in range(len(filepieces1)):
                    if i == 0:
                        print("first line", filepieces1[i])
                    if len(filepieces1[i]) != 0:
                        # update copy_old boolean once the old bond and angle section is skipped
                        if filepieces1[i][0] == "Estimated" and filepieces1[i][1] == "Stretch-Bend":
                            # resume copying the old parameter lines once past the old bond and angle section
                            copy_old = True
                        # once "Estimate Bond" is reached, begin copying new bond and angle parameters
                        if filepieces1[i][0] == "Estimated" and filepieces1[i][1] == "Bond":
                            # iterate over each line of the new parameter file and write them into the edit
                            for j in range(len(filepieces2)):
                                if len(filepieces2[j]) != 0:
                                    # start copying the new parameters once "# Results" is detected
                                    if filepieces2[j][0] == "bond" and copy_new == False:
                                        edited.write("#\n")
                                        edited.write("# Results of Valence Parameter Fitting\n")
                                        edited.write("#\n")
                                        copy_new = True
                                    # copy new bond and angle parameters
                                    if copy_new == True:
                                        edited.write(filelines2[j])
                            # stop copying until "Estimated Stretch-Bend" is found in the old parameter file
                            copy_old = False
                        # by default, copy old parameter lines (except old bond and angle parameters)
                        elif copy_old == True:
                            edited.write(filelines1[i])
```

```
                    # include line returns to preserve formatting
                    else:
                        edited.write("\n")
    print("final parameter file is %s"%save_file)

def insert_params(old_params, name):
    save_file = "%s_new_new_2.key"%name
    # boolean to track copying of new multipole values
    copy_new = False
    # boolean to track copying of old multipole values
    copy_old = True
    with open(old_params) as fileblock:
        filelines = fileblock.readlines()
        # split lines by spaces
        filepieces1 = list_splitter(filelines)
        with open(save_file, 'w') as edited:
            # first line file name (needed for conversion to OpenMM later
    on
            edited.write("forcefield AMOEBA-%s\n\n"%name)
            # insert each of the Amoeba setup parameters
            edited.write("bond-cubic              -2.55\n")
            edited.write("bond-quartic            3.793125\n")
            edited.write("angle-cubic             -0.014\n")
            edited.write("angle-quartic           0.000056\n")
            edited.write("angle-pentic            -0.0000007\n")
            edited.write("angle-sextic            0.000000022\n")
            edited.write("opbendtype              ALLINGER\n")
            edited.write("opbend-cubic            -0.014\n")
            edited.write("opbend-quartic          0.000056\n")
            edited.write("opbend-pentic           -0.0000007\n")
            edited.write("opbend-sextic           0.000000022\n")
            edited.write("torsionunit             0.5\n")
            edited.write("vdwtype                 BUFFERED-14-7\n")
            edited.write("radiusrule              CUBIC-MEAN\n")
            edited.write("radiustype              R-MIN\n")
            edited.write("radiussize              DIAMETER\n")
            edited.write("epsilonrule             HHG\n")
            edited.write("dielectric              1.0\n")
            edited.write("polarization            MUTUAL\n")
            edited.write("vdw-12-scale            0.0\n")
            edited.write("vdw-13-scale            0.0\n")
            edited.write("vdw-14-scale            1.0\n")
            edited.write("vdw-15-scale            1.0\n")
            edited.write("mpole-12-scale          0.0\n")
            edited.write("mpole-13-scale          0.0\n")
            edited.write("mpole-14-scale          0.4\n")
            edited.write("mpole-15-scale          0.8\n")
            edited.write("polar-12-scale          0.0\n")
            edited.write("polar-13-scale          0.0\n")
            edited.write("polar-14-scale          1.0\n")
            edited.write("polar-15-scale          1.0\n")
            edited.write("polar-12-intra          0.0\n")
            edited.write("polar-13-intra          0.0\n")
            edited.write("polar-14-intra          0.5\n")
```

```
            edited.write("polar-15-intra          1.0\n")
            edited.write("direct-11-scale         0.0\n")
            edited.write("direct-12-scale         1.0\n")
            edited.write("direct-13-scale         1.0\n")
            edited.write("direct-14-scale         1.0\n")
            edited.write("mutual-11-scale         1.0\n")
            edited.write("mutual-12-scale         1.0\n")
            edited.write("mutual-13-scale         1.0\n")
            edited.write("mutual-14-scale         1.0\n\n")

            # iterate over each line of the original (old) parameter (.key) file and copy as is
            for i in range(len(filelines)):
                edited.write(filelines[i])
    print("newest parameter file is %s"%save_file)

def find_atom_types(old_params, new_params, name, starting_index):
    ''' uses an input file to find the atom types to reorder atom types the name_re-opt.txyz
    When babel is called to convert to .txyz it uses atom type assignments that do not apply to these custom setups.

    '''
    save_file = "%s_re-opt.txyz"%name
    # boolean to track copying of new multipole values
    copy_new = False
    # boolean to track copying of old multipole values
    copy_old = False
    # dict to hold atom index and associated atom types from the poledit output file
    atypes = {}
    with open(old_params) as fileblock1:
        filelines1 = fileblock1.readlines()
        # split lines by spaces
        filepieces1 = list_splitter(filelines1)
        with open(new_params) as fileblock2:
            filelines2 = fileblock2.readlines()
            # split lines by spaces
            filepieces2 = list_splitter(filelines2)

            with open(save_file, 'w') as edited:
                # iterate over each line of the original (old) parameter (.key) file
                for i in range(len(filepieces1)):
                    if i == 0:
                        print("first line", filepieces1[i])
                    if len(filepieces1[i]) != 0:
                        # update copy_old boolean once the old bond and angle section is skipped
                        if filelines1[i] == " Atom Type and Local Frame Definition for Each Atom :\n":
                            print("MADE IT HERE")
                            # resume copying the old parameter lines once past the old bond and angle section
```

```
435                             copy_old = True
436                         elif filelines1[i] == " Final Atomic Multipole
    Moments after Regularization :\n":
437                             # halt copying (dictionary creation) once this
     line is encountered
438                             copy_old = False
439 
440                         # skip empty lines
441                         if len(filepieces1[i]) != 0 and copy_old == True:
442                             # skip "Atom" line to get to index lines
443                             if filepieces1[i][0] != "Atom":
444                                 #print("current line" + filelines1[i])
445                                 a_index = filepieces1[i][0]
446                                 a_type = filepieces1[i][1]
447                                 #print("atom index %s with atom type %s"%(
    a_index, a_type))
448                                 atypes.update({int(a_index) : int(a_type)
    })
449                 # iterate over each line of the new parameter file and
    write them into the edit
450                 for j in range(len(filepieces2)):
451                     # just copy the first line
452                     if j == 0:
453                         edited.write(filelines2[j])
454                     else:
455                         edited_line = ""
456                         for k in range(len(filepieces2[j])):
457                             # if not the 5th (starting from zero) element
    of the list, include the original line fragment
458                             if k != 5:
459                                 edited_line += "    %s"%filepieces2[j][k]
460                             else:
461                                 # the new type is based on the old type
    which starts from 1 (hence the subtraction)
462                                 new_type = atypes[int(filepieces2[j][0])]
    + int(starting_index) -1
463                                 edited_line += "    %i"%new_type
464                         edited.write(edited_line + "\n")
465                         # iterate over the segments of each line, updating
     the atom types
466 
467     print("newest parameter file is %s"%save_file)
468 
469 def main():
470     parser = argparse.ArgumentParser()
471     parser.add_argument("-c", "--coordinates", help="starting coordinates
    for molecule of interest")
472     parser.add_argument("-d", "--charge", default=0, help="charge of
    molecule")
473     parser.add_argument("-s", "--spin", default=1, help="spin multiplicity
     of molecule")
474     parser.add_argument("-n", "--name", help="name of molecule")
475     parser.add_argument("-p", "--part", help="part of process to run, run
    part 1 before part 2")
```

```
    parser.add_argument("-t", "--type_index", help="starting atom type index")
    parser.add_argument("-i", "--class_index", help="starting atom class index")
    args = parser.parse_args()

    # Programs needed: Gaussian, Tinker, GDMA, Babel

    current_dir = os.getcwd() + '/'

    command_01 = ["mehutchi/opt/g16/g16", "%s.com"%args.name]
    print("command", command_01)
    command_02 = ""

    # start of code (in the urea folder)
    # PART ONE
    # start out with a traditional (not-TINKER) name.xyz file and  a symbolic link to amoeba09.prm
    if args.part == "1":
        print("***** running part 1 *****")
        #make_gauss_input(args.name=name, args.coordinates=coord_name, args.charge=charge, args.spin=spin)
        # create the gaussian input
        make_gauss_input(args.name, args.coordinates, 1, args.charge, args.spin)

        # run initial gaussian optimization (command_1)
        print("running initial Guassian optimization")
        os.system("g16 %s.com"%args.name)

        # run formchk (command_2)
        print("running Gaussian formchk")
        os.system("formchk %s.chk %s.fchk"%(args.name, args.name))
        # create gdma input
        make_gdma_input(args.name)
        # run gdma program (command_3)
        print("running the GDMA program")
        os.system("gdma < %s.gdmain > %s.gdmaout"%(args.name, args.name))
        # create Tinker poledit input (command_4)
        make_poledit_input(args.name)
        ### add error check here if no amoeba09.prm file in folder!!!
        # run Tinker poledit
        print("running TINKER poledit")
        os.system("poledit < input_poledit.txt > output_poledit.txt")
        # edit starting atom types and atom classes indices of .key file (command_4.5)
        # choline+ starts atom type indices at 500 and atom classes at 200, so the H-bond donors start from 600 and 300 respectively
        #starting_atom_type = 500
        #starting_atom_class = 200
        print("Before running part 2, review %s.key to make sure that the atom types were correctly assigned.\nSometimes TINKER doesn't correcly identify all equivalent atoms.\nUse TINKER poledit to \"Condense Symmetric Atoms to Equivalent Types\""%args.name)
```

```
520         print("If changes are made, make sure to name the correct key file
    : %s_new.key. \nRe-running TINKER poledit manually should save the
    updated .key file as %s_new.key_2 (or higher), also in that file it
    will append the new parameters after the old ones, make sure the old
    ones are removed."%(args.name, args.name))
521         print("output_poledit.txt is a required file for part 2 of this
    auto code. Capturing the entire screen output from manually running
    TINKER poledit is not compatable with the function")
522 
523     if args.part == "2":
524         print("***** running part 2 *****")
525         make_prmedit_input(args.name, args.type_index)
526         print("running TINKER prmedit")
527         # run prmedit (creates parameter.prm) !!! this removes atom
    classes instead of editing them!!!
528         os.system("prmedit < input_prmedit.txt > output_prmedit.txt")
529 
530         # make a copy of parameter.prm
531         #if path.exists("parameter.prm") == True:
532         #    os.system("cp parameter.prm parameter_orig.prm")
533         # manually insert missing atom class indices
534         print("copying parameter.prm into %s_new.key and correcting atom
    type and atom class indices"%args.name)
535         insert_atom_class_indices(args.name, args.class_index)
536 
537         # OUTDATED: before running part 2, run part 1, then edit parameter
    .prm to include atom class incides
538         #print("***** running (OLD) part 2 *****")
539 #         """
540         # rename parameter.prm (command_5)
541         #if path.exists("parameter.prm") == True:
542         #    os.system("mv parameter.prm %s_new.key"%args.name)
543         # get initial optimization coordinates -> name_opt.xyz
544         print("running babel to convert .log to .xyz")
545         os.system("babel %s.log %s_opt.xyz"%(args.name, args.name))
546         # create input for next Gaussian optimization
547         make_gauss_input(args.name, "%s_opt.xyz"%args.name, 2, args.charge
    , args.spin)
548         # run second gaussian optimization (command_6)
549         print("running second Gaussian optimization")
550         os.system("g16 %s_re-opt.com"%args.name)
551 #         """
552 
553         # run formchk on the gaussion re-optimization (command_7)
554         print("running Gaussian formchk (on second opt results)")
555         os.system("formchk %s_re-opt.chk %s_re-opt.fchk"%(args.name, args.
    name))
556         # run babel to get the re-opt coordinates (command_8)
557         print("running babel to convert .fchk to .txyz")
558         os.system("babel -ifchk %s_re-opt.fchk -otxyz %s_re-
    opt_WRONG_TYPES.txyz"%(args.name, args.name))
559         print("the next step before running part 3, is to edit %s_re-opt.
    txyz to have to the correct atom types"%args.name)
560 #     if args.part == "t":
```

```
561         print("made it to function call")
562         # trying out the function that takes care of the atom types
563         find_atom_types("output_poledit.txt", "%s_re-opt_WRONG_TYPES.txyz"
    %args.name, args.name, args.type_index)
564 
565     # OUTDATED: before running part 3, run part 1 then part 2 and edit
    name_re-opt.txyz to have the correct atom type
566 #    if args.part == "3":
567         #print("***** running (OLD) part 3 *****")
568         # create input file for potential option #1
569         make_potential_input(args.name, 1)
570         # run tinker potential option #1 (command_9)
571         print("running TINKER potential option #1")
572         os.system("potential < input_potential_1.txt > output_potential_1.
    txt")
573         # run cubegen (command_10)
574         print("running Gaussian cubegen")
575         os.system("cubegen 0 potential=mp2 %s_re-opt.fchk %s_re-opt.cube
    -5 h < %s_re-opt.grid"%(args.name, args.name, args.name))
576         # create input file for potential option #2
577         make_potential_input(args.name, 2)
578         # run tinker potential option #2 (command_11)
579         print("running TINKER potential option #2")
580         os.system("potential < input_potential_2.txt > output_potential_2.
    txt")
581         # insert "FIX-MONOPOLE" into .key file produced in step 5 (
    command_12)
582         #???i
583         line_prepender("%s_new.key"%args.name , "FIX-MONOPOLE")
584         # create input file for potential option #6
585         print("running TINKER potential option #6")
586         make_potential_input(args.name, 6)
587         # run tinker potential option #6 (command_13)
588         os.system("potential < input_potential_6.txt > output_potential_6.
    txt")
589 
590         # combine new fitted multipole results from name_re-opt.key with
    atom definitions AND polarize values from name_new.key to create
    name_new_new.key
591         combine_mp_parameters("%s_new.key"%args.name, "%s_re-opt.key"%args
    .name, args.name)
592         # insert amoeba parameters before running valence later on ->
    name_new_new_2.key
593         insert_params("%s_new_new.key"%args.name, args.name)
594 
595     # OUTDATED: before running part 4, run part 1 then part 2 then part 3
    and combine the multipole values from name_re-opt.key with the monopole
     values from name_new.key to create name_new_new.key (command_14)
596         #print("***** running (OLD) part 4 *****")
597         # create input file for valence option #1
598         make_valence_input(args.name, 1)
599         # run tinker valence option #1 (command_15)
600         print("running TINKER valence option #1")
601         os.system("valence < input_valence_1.txt > output_valence_1.txt")
```

```
        # combine new valence parameter values with existing parameter values into new .key file
        combine_val_parameters("%s_new_new_2.key"%args.name, "output_valence_1.txt", args.name)

     # OUTDATED: before running part 5, run parts 1-4 in sequence, then combine the parameters in output_valence_1.txt with those in name_new_new.key to form name_new_new_new.key (command_16)
        #print("***** running (OLD) part 5 *****")
        # create .xyz coordinate file to be used to create a gaussian input file
        print("running babel to convert .txyz to .xyz")
        os.system("babel %s_re-opt.txyz %s_re-opt.xyz"%(args.name, args.name))
        # make gaussian input file for frequency calculation
        make_gauss_input(args.name, "%s_re-opt.xyz"%args.name, 3, args.charge, args.spin)
        # run gaussian frequency calculation (command_17)
        print("running Gaussian frequency calculation")
        os.system("g16 %s_freq.com"%args.name)
        # create input file for valence option #2
        make_valence_input(args.name, 2)
        # run tinker valence option #2 which is just a comparison (command_18)
        print("running TINKER valence option #2")
        os.system("valence < input_valence_2.txt > output_valence_2.txt")
        # create input file for valence option #3
        make_valence_input(args.name, 3)
        # run tinker valence option #3 (functional command_19)
        print("running TINKER valence option #3")
        os.system("valence < input_valence_3.txt > output_valence_3.txt")

        # OUTDATED: after running parts 1-5 the new bond and angle parameters will be named after the input coordinates (name_re-opt.txyz), so these will be named name_re-opt.key_2 because we have name_re-opt.key from before
        # OUTDATED: the next step is to replace those parameters in name_new_new_new.key to form name_final.key. Note, the multipole parameters are included in name_re-opt.key_2 even though they didn't change from earlier
        combine_final_parameters("%s_new_new_new.key"%args.name, "%s_re-opt.key_2"%args.name, args.name)

if __name__ == '__main__':
    main()
```

Listing 3: Semi-Automated Polarizable Force Field Initialization Code

```
<ForceField>
 <Info>
  <Source>urea_final.key</Source>
  <DateGenerated>2022-06-05</DateGenerated>
  <Reference></Reference>
```

```
</Info>
<AtomTypes>
 <Type name="600" class="300" element="C" mass="12.011"/>
 <Type name="601" class="301" element="N" mass="14.007"/>
 <Type name="602" class="302" element="H" mass="1.008"/>
 <Type name="603" class="303" element="O" mass="15.999"/>
</AtomTypes>
<Residues>
 <Residue name="URE">
  <Atom name="C1" type="600" />
  <Atom name="H4" type="602" />
  <Atom name="H5" type="602" />
  <Atom name="H6" type="602" />
  <Atom name="H7" type="602" />
  <Atom name="N2" type="601" />
  <Atom name="N3" type="601" />
  <Atom name="O8" type="603" />
  <Bond from="0" to="5" />
  <Bond from="0" to="6" />
  <Bond from="0" to="7" />
  <Bond from="1" to="6" />
  <Bond from="2" to="6" />
  <Bond from="3" to="5" />
  <Bond from="4" to="5" />
</Residue>
</Residues>
<AmoebaBondForce bond-cubic="-25.5" bond-quartic="379.3125">
 <Bond class1="300" class2="301" length="0.13347" k="256968.728"/>
 <Bond class1="300" class2="303" length="0.11994" k="396262.456"/>
 <Bond class1="301" class2="302" length="0.10017" k="236693.064"/>
</AmoebaBondForce>
<AmoebaAngleForce angle-cubic="-0.014" angle-quartic="5.6e-05" angle-
  pentic="-7e-07" angle-sextic="2.2e-08">
 <Angle class1="301" class2="300" class3="301" k="0.0235786067612" angle1
  ="152.31"  />
 <Angle class1="301" class2="300" class3="303" k="0.00520003868032"
  angle1="195.32"  />
 <Angle class1="300" class2="301" class3="302" k="0.0495533097771" angle1
  ="116.05"  />
 <Angle class1="302" class2="301" class3="302" k="0.0545749157576" angle1
  ="114.75"  />
</AmoebaAngleForce>
<AmoebaOutOfPlaneBendForce type="ALLINGER" opbend-cubic="-0.014" opbend-
  quartic="5.6e-05" opbend-pentic="-7e-07" opbend-sextic="2.2e-08">
 <Angle class1="301" class2="300" class3="0" class4="0" k="
  0.0183530776952"/>
 <Angle class1="303" class2="300" class3="0" class4="0" k="
  0.0183530776952"/>
</AmoebaOutOfPlaneBendForce>
<PeriodicTorsionForce>
 <Proper class1="301" class2="300" class3="301" class4="302"   k1="0.0"
  phase1="0.0" periodicity1="1"   k2="1.046" phase2="3.14159265359"
  periodicity2="2"   k3="0.523" phase3="0.0" periodicity3="3" />
 <Proper class1="303" class2="300" class3="301" class4="302"   k1="0.0"
```

```
   phase1="0.0" periodicity1="1"   k2="1.046" phase2="3.14159265359"
   periodicity2="2"   k3="0.523" phase3="0.0" periodicity3="3" />
 </PeriodicTorsionForce>
 <AmoebaStretchBendForce stretchBendUnit="1.0">
  <StretchBend class1="301" class2="300" class3="301" k1="13.655595694" k2
   ="13.655595694" />
  <StretchBend class1="301" class2="300" class3="303" k1="13.655595694" k2
   ="13.655595694" />
  <StretchBend class1="300" class2="301" class3="302" k1="5.25776946506"
   k2="3.14005676385" />
</AmoebaStretchBendForce>
 <AmoebaVdwForce type="BUFFERED-14-7" radiusrule="CUBIC-MEAN" radiustype="
   R-MIN" radiussize="DIAMETER" epsilonrule="HHG" vdw-13-scale="0.0" vdw
   -14-scale="1.0" vdw-15-scale="1.0" >
  <Vdw class="300" sigma="0.19" epsilon="0.372376" reduction="1.0" />
  <Vdw class="301" sigma="0.1855" epsilon="0.43932" reduction="1.0" />
  <Vdw class="302" sigma="0.135" epsilon="0.08368" reduction="0.91" />
  <Vdw class="303" sigma="0.165" epsilon="0.468608" reduction="1.0" />
 </AmoebaVdwForce>
 <AmoebaMultipoleForce  direct11Scale="0.0"  direct12Scale="1.0"
   direct13Scale="1.0"  direct14Scale="1.0"  mpole12Scale="0.0"
   mpole13Scale="0.0"  mpole14Scale="0.4"  mpole15Scale="0.8"
   mutual11Scale="1.0"  mutual12Scale="1.0"  mutual13Scale="1.0"
   mutual14Scale="1.0"  polar12Scale="0.0"  polar13Scale="0.0"
   polar14Intra="0.5"  polar14Scale="1.0"  polar15Scale="1.0"  >
  <Multipole type="600" kz="603" kx="601" c0="1.03617" d1="0.0" d2="0.0"
   d3="0.00553572277906" q11="6.86069869323e-06" q21="0.0" q22="
   -5.89366687742e-05" q31="0.0" q32="0.0" q33="5.2075970081e-05"  />
  <Multipole type="601" kz="600" kx="602" c0="-0.36361" d1="
   -0.00625804966879" d2="0.0" d3="-0.00860653812051" q11="
   0.000609864775267" q21="0.0" q22="-0.000751120494076" q31="
   -8.08162303211e-05" q32="0.0" q33="0.000141255718809"  />
  <Multipole type="602" kz="601" kx="600" c0="0.13019" d1="
   0.00060802461267" d2="0.0" d3="-0.00978713247287" q11="
   0.000107978996576" q21="0.0" q22="-6.50692933204e-05" q31="
   2.25796328421e-05" q32="0.0" q33="-4.29097032555e-05"  />
  <Multipole type="603" kz="600" kx="601" c0="-0.82971" d1="0.0" d2="0.0"
   d3="-0.0116995789047" q11="-0.000402158289115" q21="0.0" q22="
   0.000197438773822" q31="0.0" q32="0.0" q33="0.000204719515292"  />
  <Polarize type="600" polarizability="0.001334" thole="0.3900" pgrp1="601
   " pgrp2="603" />
  <Polarize type="601" polarizability="0.001073" thole="0.3900" pgrp1="600
   " pgrp2="602" />
  <Polarize type="602" polarizability="0.000496" thole="0.3900" pgrp1="601
   " />
  <Polarize type="603" polarizability="0.000837" thole="0.3900" pgrp1="600
   " />
 </AmoebaMultipoleForce>
</ForceField>
```

Listing 4: AMOEBA-OpenMM-urea force field. Identical to AMOEBA-Tinker-urea in Listing 1 but uses the OpenMM engine.

References

[1] Marshall Hutchings, Johnson Liu, Yudong Qiu, Chenchen Song, and Lee-Ping Wang. Bond-order time series analysis for detecting reaction events in ab initio molecular dynamics simulations. *Journal of Chemical Theory and Computation*, 16(3):1606–1617, Mar 2020.

[2] GEORGE E. SMITH. J. j. thomson and the electron: 1897–1899 an introduction. *The Chemical Educator*, 2(6):1–42, Dec 1997.

[3] E. Rutherford. The scattering of αand βparticles by matter and the structure of the atom. *Philosophical Magazine*, 1911.

[4] J.R Rydberg. On the structure of the line-spectra of the chemical elements. *Philosophical Magazine*, 1890.

[5] Kyle D. Shaffer. Determination of the rydberg constant from the emission spectra of h and he+. *Ramifications*, 2, 2020.

[6] A. Pais. Einstein and the quantum theory. *Rev. Mod. Phys.*, 51:863–914, Oct 1979.

[7] N. Nilius, T. M. Wallis, and W. Ho. Realization of a particle-in-a-box: electron in an atomic pd chain. *The Journal of Physical Chemistry B*, 109(44):20657–20660, Nov 2005.

[8] P. Heenen and M. R. Godefroid. The Hartree-Fock method. *Scholarpedia*, 7(10):10545, 2012. revision #129749.

[9] S. M. Blinder. Basic concepts of self-consistent-field theory. *American Journal of Physics*, 33(6):431–443, 1965.

[10] Chr. Møller and M. S. Plesset. Note on an approximation treatment for many-electron systems. *Physical Review*, 1934.

[11] Dieter Cremer. Møller–plesset perturbation theory: from small molecule methods to methods for thousands of atoms. *WIREs Computational Molecular Science*, 1(4):509–530, 2011.

[12] C. David Sherrill and Henry F. Schaefer. The configuration interaction method: Advances in highly correlated approaches. In Per-Olov Löwdin, John R. Sabin, Michael C. Zerner, and Erkki Brändas, editors, *Advances in Quantum Chemistry*, volume 34 of *Advances in Quantum Chemistry*, pages 143–269. Academic Press, 1999.

[13] Włodzisław Duch. Configuration interaction method: the past and future perspectives. *Journal of Molecular Structure: THEOCHEM*, 234:27–49, 1991.

[14] C. David Sherrill and Henry F. Schaefer. The configuration interaction method: Advances in highly correlated approaches. *Advances in Quantum Chemistry*, 34:143–269, 1999.

[15] Yutaka Shikano, Hiroshi C. Watanabe, Ken M. Nakanishi, and Yu ya Ohnishi. Post-hartree–fock method in quantum chemistry for quantum computer. *The European Physical Journal Special Topics*, 230(4):1037–1051, apr 2021.

[16] Nathan Argaman and Guy Makov. Density functional theory: An introduction. *American Journal of Physics*, 68(1):69–79, 2000.

[17] P. Hohenberg and W. Kohn. Inhomogeneous electron gas. *Phys. Rev.*, 136:B864–B871, Nov 1964.

[18] Wolfram Koch and Max C. Holthausen. *The Hohenberg-Kohn Theorems*, chapter 4, pages 33–40. John Wiley & Sons, Ltd, 2001.

[19] W. Kohn and L. J. Sham. Self-consistent equations including exchange and correlation effects. *Phys. Rev.*, 140:A1133–A1138, Nov 1965.

[20] Axel D. Becke. Density-functional thermochemistry. iii. the role of exact exchange. *The Journal of Chemical Physics*, 98(7):5648–5652, 1993.

[21] C Lee, W Yang, and R G Parr. Development of the Colle-Salvetti correlation-energy formula into a functional of the electron density. *Phys Rev B Condens Matter*, 37(2):785–789, January 1988.

[22] S. H. Vosko, L. Wilk, and M. Nusair. Accurate spin-dependent electron liquid correlation energies for local spin density calculations: a critical analysis. *Canadian Journal of Physics*, 59:1200, August 1980.

[23] P. J. Stephens, F. J. Devlin, C. F. Chabalowski, and M. J. Frisch. Ab initio calculation of vibrational absorption and circular dichroism spectra using density functional force fields. *The Journal of Physical Chemistry*, 98(45):11623–11627, Nov 1994.

[24] Diola Bagayoko. Understanding density functional theory (dft) and completing it in practice. *AIP Advances*, 4(12):127104, 2014.

[25] D. M. Ceperley and B. J. Alder. Ground state of the electron gas by a stochastic method. *Phys. Rev. Lett.*, 45:566–569, Aug 1980.

[26] Marco Arrigoni and Georg K.H. Madsen. Comparing the performance of lda and gga functionals in predicting the lattice thermal conductivity of iii-v semiconductor materials in the zincblende structure: The cases of alas and bas. *Computational Materials Science*, 156:354–360, 2019.

[27] John P. Perdew, J. A. Chevary, S. H. Vosko, Koblar A. Jackson, Mark R. Pederson, D. J. Singh, and Carlos Fiolhais. Atoms, molecules, solids, and surfaces: Applications of the generalized gradient approximation for exchange and correlation. *Phys. Rev. B*, 46:6671–6687, Sep 1992.

[28] John P. Perdew, Kieron Burke, and Matthias Ernzerhof. Generalized gradient approximation made simple. *Phys. Rev. Lett.*, 77:3865–3868, Oct 1996.

[29] A. D. Becke. Density-functional exchange-energy approximation with correct asymptotic behavior. *Phys. Rev. A*, 38:3098–3100, Sep 1988.

[30] Viktor N. Staroverov, Gustavo E. Scuseria, Jianmin Tao, and John P. Perdew. Comparative assessment of a new nonempirical density functional: Molecules and hydrogen-bonded complexes. *The Journal of Chemical Physics*, 119(23):12129–12137, 2003.

[31] Narbe Mardirossian and Martin Head-Gordon. Thirty years of density functional theory in computational chemistry: an overview and extensive assessment of 200 density functionals. *Molecular Physics*, 115(19):2315–2372, 2017.

[32] Stephen K. Gray, Donald W. Noid, and Bobby G. Sumpter. Symplectic integrators for large scale molecular dynamics simulations: A comparison of several explicit methods. *The Journal of Chemical Physics*, 101(5):4062–4072, 1994.

[33] Benedict J. Leimkuhler, Sebastian Reich, and Robert D. Skeel. Integration methods for molecular dynamics. In Jill P. Mesirov, Klaus Schulten, and De Witt Sumners, editors, *Mathematical Approaches to Biomolecular Structure and Dynamics*, pages 161–185. Springer New York, New York, NY, 1996.

[34] Lee-Ping Wang, Alexey Titov, Robert McGibbon, Fang Liu, Vijay S. Pande, and Todd J. Martínez. Discovering chemistry with an ab initio nanoreactor. *Nat. Chem.*, 6:1044–1046, 2014.

[35] Miguel Steiner and Markus Reiher. Autonomous reaction network exploration in homogeneous and heterogeneous catalysis. *Topics in Catalysis*, 65(1):6–39, Feb 2022.

[36] Zachary W Ulissi, Andrew J Medford, Thomas Bligaard, and Jens K Nørskov. To address surface reaction network complexity using scaling relations machine learning and DFT calculations. *Nat Commun*, 8:14621, March 2017.

[37] Connie W. Gao, Joshua W. Allen, William H. Green, and Richard H. West. Reaction Mechanism Generator: Automatic construction of chemical kinetic mechanisms. *Comput. Phys. Commun.*, 203:212–225, 2016.

[38] Mengjie Liu, Alon Grinberg Dana, Matthew S. Johnson, Mark J. Goldman, Agnes Jocher, A. Mark Payne, Colin A. Grambow, Kehang Han, Nathan W. Yee, Emily J. Mazeau, Katrin Blondal, Richard H. West, C. Franklin Goldsmith, and William H. Green. Reaction mechanism generator v3.0: Advances in automatic mechanism generation. *Journal of Chemical Information and Modeling*, 61(6):2686–2696, Jun 2021.

[39] Emily J. Mazeau, Priyanka Satpute, Katrín Blöndal, C. Franklin Goldsmith, and Richard H. West. Automated mechanism generation using linear scaling relationships and sensitivity analyses applied to catalytic partial oxidation of methane. *ACS Catalysis*, 11(12):7114–7125, Jun 2021.

[40] Gregor N. Simm and Markus Reiher. Context-driven exploration of complex chemical reaction networks. *Journal of Chemical Theory and Computation*, 13(12):6108–6119, Dec 2017.

[41] Jan P Unsleber, Stephanie A Grimmel, and Markus Reiher. Chemoton 2.0: Autonomous exploration of chemical reaction networks. *Journal of chemical theory and computation : JCTC*, 2022-08-04.

[42] Satoshi Maeda, Koichi Ohno, and Keiji Morokuma. Systematic exploration of the mechanism of chemical reactions: the global reaction route mapping (grrm) strategy using the addf and afir methods. *Phys. Chem. Chem. Phys.*, 15:3683–3701, 2013.

[43] Koichi Ohno, Satoshi Maeda, Yuto Osada, and Keiji Morokuma. Grrm website. https://iqce.jp/GRRM/category/Gaiyo_e.shtml.

[44] Paul M. Zimmerman. Automated discovery of chemically reasonable elementary reaction steps. *J. Comput. Chem*, 34(16):1385–1392, 2013.

[45] Mina Jafari and Paul M. Zimmerman. Uncovering reaction sequences on surfaces through graphical methods. *Phys. Chem. Chem. Phys.*, 20:7721–7729, 2018.

[46] Alberto Baiardi, Stephanie A. Grimmel, Miguel Steiner, Paul L. Türtscher, Jan P. Unsleber, Thomas Weymuth, and Markus Reiher. Expansive quantum mechanical exploration of chemical reaction paths. *Accounts of Chemical Research*, 55(1):35–43, Jan 2022.

[47] Xin Chen, Meiyi Liu, and Jiali Gao. Carnot: a fragment-based direct molecular dynamics and virtual–reality simulation package for reactive systems. *Journal of Chemical Theory and Computation*, 18(3):1297–1313, Mar 2022.

[48] Guan Zhang, Jin Li, Xinxin Liang, and Zongkuan Liu. Automated reaction mechanisms and kinetics with the nudged elastic band method-based amk_mountain and its description of the preliminary alkaline hydrolysis of nitrocellulose monomer. *Journal of Computational Chemistry*, 43(22):1513–1523, 2022.

[49] Emilio Martínez-Núñez, George L. Barnes, David R. Glowacki, Sabine Kopec, Daniel Peláez, Aurelio Rodríguez, Roberto Rodríguez-Fernández, Robin J. Shannon, James J. P. Stewart, Pablo G. Tahoces, and Saulo A. Vazquez. Automekin2021: An open-source program for automated reaction discovery. *Journal of Computational Chemistry*, 42(28):2036–2048, 2021.

[50] Betsy M. Rice, William D. Mattson, James P. Larentzos, and Edward F. C. Byrd. Heuristics for chemical species identification in dense systems. *The Journal of Chemical Physics*, 153(6):064102, 2020.

[51] Adri C. T. van Duin, Siddharth Dasgupta, Francois Lorant, and William A. Goddard. Reaxff: a reactive force field for hydrocarbons. *The Journal of Physical Chemistry A*, 105(41):9396–9409, Oct 2001.

[52] Thomas P. Senftle, Sungwook Hong, Md Mahbubul Islam, Sudhir B. Kylasa, Yuanxia Zheng, Yun Kyung Shin, Chad Junkermeier, Roman Engel-Herbert, Michael J. Janik,

Hasan Metin Aktulga, Toon Verstraelen, Ananth Grama, and Adri C. T. van Duin. The reaxff reactive force-field: development, applications and future directions. *npj Computational Materials*, 2(1):15011, Mar 2016.

[53] Bernd Hartke and Stefan Grimme. Reactive force fields made simple. *Phys. Chem. Chem. Phys.*, 17:16715–16718, 2015.

[54] Tzeliou CE, Mermigki MA, and Tzeli D and. Review on the qm/mm methodologies and their application to metalloproteins. *Molecules*, 2022.

[55] Matthias Hofmann and Henry F. Schaefer. Computational chemistry. In Robert A. Meyers, editor, *Encyclopedia of Physical Science and Technology (Third Edition)*, pages 487–506. Academic Press, New York, third edition edition, 2003.

[56] Alexander D. MacKerell, Michael Feig, and Charles L. Brooks. Improved treatment of the protein backbone in empirical force fields. *Journal of the American Chemical Society*, 126(3):698–699, Jan 2004.

[57] Alexander D. Mackerell Jr. Empirical force fields for biological macromolecules: Overview and issues. *Journal of Computational Chemistry*, 25(13):1584–1604, 2004.

[58] Matthias Buck, Sabine Bouguet-Bonnet, Richard W Pastor, and Alexander D MacKerell, Jr. Importance of the CMAP correction to the CHARMM22 protein force field: dynamics of hen lysozyme. *Biophys J*, 90(4):L36–8, December 2005.

[59] Judith Harrison, J. Schall, Sabina Maskey, Paul Mikulski, M. Knippenberg, and Brian Morrow. Review of force fields and intermolecular potentials used in atomistic computational materials research. *Applied Physics Reviews*, 5:031104, 09 2018.

[60] Arieh Warshel, Mitsunori Kato, and Andrei V. Pisliakov. Polarizable force fields: history, test cases, and prospects. *Journal of Chemical Theory and Computation*, 3(6):2034–2045, Nov 2007.

[61] Anna S. Kamenik, Philip H. Handle, Florian Hofer, Ursula Kahler, Johannes Kraml, and Klaus R. Liedl. Polarizable and non-polarizable force fields: Protein folding, unfolding, and misfolding. *The Journal of Chemical Physics*, 153(18):185102, 2020.

[62] Fang-Yu Lin, Jing Huang, Poonam Pandey, Chetan Rupakheti, Jing Li, Benoît Roux, and Alexander D. MacKerell. Further optimization and validation of the classical drude polarizable protein force field. *Journal of Chemical Theory and Computation*, 16(5):3221–3239, May 2020.

[63] Andrew P. Abbott, Glen Capper, David L. Davies, Helen L. Munro, Raymond K. Rasheed, and Vasuki Tambyrajah. Preparation of novel, moisture-stable, lewis-acidic ionic liquids containing quaternary ammonium salts with functional side chains. *Chem. Commun.*, pages 2010–2011, 2001.

[64] Emma L. Smith, Andrew P. Abbott, and Karl S. Ryder. Deep eutectic solvents (dess) and their applications. *Chemical Reviews*, 114(21):11060–11082, Nov 2014.

[65] Yuntao Dai, Jaap van Spronsen, Geert-Jan Witkamp, Robert Verpoorte, and Young Hae Choi. Ionic liquids and deep eutectic solvents in natural products research: Mixtures of solids as extraction solvents. *Journal of Natural Products*, 76(11):2162–2173, Nov 2013.

[66] Christopher P. Cabry, Lucía D'Andrea, Karina Shimizu, Isabelle Grillo, Peixun Li, Sarah Rogers, Duncan W. Bruce, José N. Canongia Lopes, and John M. Slattery. Exploring the bulk-phase structure of ionic liquid mixtures using small-angle neutron scattering. *Faraday Discuss.*, 206:265–289, 2018.

[67] John M Slattery, Corinne Daguenet, Paul J Dyson, Thomas J S Schubert, and Ingo Krossing. How to predict the physical properties of ionic liquids: A volume-based approach. *Angewandte Chemie*, 46(28), 2007-07-09.

[68] Thomas Welton. Room-temperature ionic liquids. solvents for synthesis and catalysis. *Chemical Reviews*, 99(8):2071–2084, Aug 1999.

[69] Zhigang Lei, Biaohua Chen, Yoon-Mo Koo, and Douglas R. MacFarlane. Introduction: Ionic liquids. *Chemical Reviews*, 117(10):6633–6635, May 2017.

[70] Benworth B. Hansen, Stephanie Spittle, Brian Chen, Derrick Poe, Yong Zhang, Jeffrey M. Klein, Alexandre Horton, Laxmi Adhikari, Tamar Zelovich, Brian W. Doherty, Burcu Gurkan, Edward J. Maginn, Arthur Ragauskas, Mark Dadmun, Thomas A. Zawodzinski, Gary A. Baker, Mark E. Tuckerman, Robert F. Savinell, and Joshua R. Sangoro. Deep eutectic solvents: A review of fundamentals and applications. *Chemical Reviews*, 121(3):1232–1285, Feb 2021.

[71] Alexandre Paiva, Rita Craveiro, Ivo Aroso, Marta Martins, Rui L. Reis, and Ana Rita C. Duarte. Natural deep eutectic solvents – solvents for the 21st century. *ACS Sustainable Chemistry & Engineering*, 2(5):1063–1071, May 2014.

[72] Kyeong-jun Jeong, Jesse G. McDaniel, and Arun Yethiraj. Deep eutectic solvents: Molecular simulations with a first-principles polarizable force field. *The Journal of Physical Chemistry B*, 125(26):7177–7186, Jul 2021.

[73] David Mecerreyes and Luca Porcarelli. 8 - green electrolyte-based organic electronic devices. In Assunta Marrocchi, editor, *Sustainable Strategies in Organic Electronics*, Woodhead Publishing Series in Electronic and Optical Materials, pages 281–295. Woodhead Publishing, 2022.

[74] Elena Soriano and Jose Marco-Contelles. Mechanistic Insights on the Cycloisomerization of Polyunsaturated Precursors Catalyzed by Platinum and Gold Complexes. *Acc. Chem. Res.*, 42(8):1026–1036, AUG 2009.

[75] Ellen M. Sletten and Carolyn R. Bertozzi. From Mechanism to Mouse: A Tale of Two Bioorthogonal Reactions. *Acc. Chem. Res.*, 44(9, SI):666–676, SEP 2011.

[76] Sara Szymkuc, Ewa P. Gajewska, Tomasz Klucznik, Karol Molga, Piotr Dittwald, Michal Startek, Michal Bajczyk, and Bartosz A. Grzybowski. Computer-Assisted Synthetic Plan-

ning: The End of the Beginning. *Angew. Chem. Int. Edit.*, 55(20):5904–5937, MAY 10 2016.

[77] CY Peng, PY Ayala, HB Schlegel, and MJ Frisch. Using redundant internal coordinates to optimize equilibrium geometries and transition states. *J. Comput. Chem.*, 17(1):49–56, JAN 15 1996.

[78] G Henkelman, BP Uberuaga, and H Jonsson. A climbing image nudged elastic band method for finding saddle points and minimum energy paths. *J. Chem. Phys.*, 113(22):9901–9904, DEC 8 2000.

[79] AM Mebel, EWG Diau, MC Lin, and K Morokuma. Ab initio and RRKM calculations for multichannel rate constants of the C2H3+O-2 reaction. *J. Am. Chem. Soc.*, 118(40):9759–9771, OCT 9 1996.

[80] Egill Skulason, Vladimir Tripkovic, Marten E. Bjorketun, Sigridur Gudmundsdottir, Gustav Karlberg, Jan Rossmeisl, Thomas Bligaard, Hannes Jonsson, and Jens K. Norskov. Modeling the Electrochemical Hydrogen Oxidation and Evolution Reactions on the Basis of Density Functional Theory Calculations. *J. Phys. Chem. C*, 114(42):18182–18197, OCT 28 2010.

[81] Scott Habershon. Automated Prediction of Catalytic Mechanism and Rate Law Using Graph-Based Reaction Path Sampling. *J. Chem. Theory Comput.*, 12(4):1786–1798, Apr 2016.

[82] Idil Ismail, Holly B.V.A. Stuttaford-Fowler, Curtis Ochan Ashok, Christopher Robertson, and Scott Habershon. Automatic Proposal of Multistep Reaction Mechanisms using a Graph-Driven Search. *J. Phys. Chem. A*, 123(15):3407–3417, apr 2019.

[83] Amanda L. Dewyer, Alonso J. Arguelles, and Paul M. Zimmerman. Methods for exploring reaction space in molecular systems. *Wiley Interdisciplinary Reviews: Computational Molecular Science*, 8(2):e1354, 2018.

[84] Yeonjoon Kim, Jin Woo Kim, Zeehyo Kim, and Woo Youn Kim. Efficient prediction of reaction paths through molecular graph and reaction network analysis. *Chem. Sci.*, 9(4):825–835, 2018.

[85] Dmitrij Rappoport, Cooper J. Galvin, Dmitry Yu Zubarev, and Alán Aspuru-Guzik. Complex chemical reaction networks from heuristics-aided quantum chemistry. *J. Chem. Theory Comput.*, 10(3):897–907, 2014.

[86] Jan P. Unsleber and Markus Reiher. The exploration of chemical reaction networks. arxiv:1906.10223 [physics.chem-ph], 2019.

[87] Scott Habershon. Sampling reactive pathways with random walks in chemical space: Applications to molecular dissociation and catalysis. *J. Chem. Phys.*, 143(9):094106, sep 2015.

[88] Ian M. Pendleton, Monica H. Perez-Temprano, Melanie S. Sanford, and Paul M. Zimmerman. Experimental and Computational Assessment of Reactivity and Mechanism in C(sp(3))-N Bond-Forming Reductive Elimination from Palladium(IV). *J. Am. Chem. Soc.*, 138(18):6049–6060, MAY 11 2016.

[89] Dmitrij Rappoport and Alán Aspuru-Guzik. Predicting feasible organic reaction pathways using heuristically aided quantum chemistry. *J. Chem. Theory Comput.*, 15(7):4099–4112, 2019.

[90] Y Sugita and Y Okamoto. Replica-exchange molecular dynamics method for protein folding. *Chem. Phys. Lett.*, 314(1-2):141–151, NOV 26 1999.

[91] JN Onuchic, Z LutheySchulten, and PG Wolynes. Theory of protein folding: The energy landscape perspective. *Annu. Rev. Phys. Chem.*, 48:545–600, 1997.

[92] Vincent A. Voelz, Gregory R. Bowman, Kyle Beauchamp, and Vijay S. Pande. Molecular Simulation of ab Initio Protein Folding for a Millisecond Folder NTL9(1-39). *J. Am. Chem. Soc.*, 132(5):1526+, FEB 10 2010.

[93] Kresten Lindorff-Larsen, Stefano Piana, Ron O. Dror, and David E. Shaw. How Fast-Folding Proteins Fold. *Science*, 334(6055):517–520, OCT 28 2011.

[94] Lisa J. Lapidus, Srabasti Acharya, Christian R. Schwantes, Ling Wu, Diwakar Shukla, Michael King, Stephen J. Decamp, and Vijay S. Pande. Complex pathways in folding of protein G explored by simulation and experiment. *Biophys. J.*, 107(4):947–955, 2014.

[95] Ivan S. Ufimtsev and Todd J. Martínez. Quantum chemistry on graphical processing units. 1. strategies for two-electron integral evaluation. *J. Chem. Theory Comput.*, 4(2):222–231, 2008.

[96] Ivan S. Ufimtsev and Todd J. Martínez. Quantum chemistry on graphical processing units. 2. direct self-consistent-field implementation. *J. Chem. Theory Comput.*, 5(4):1004–1015, 2009.

[97] Ivan S. Ufimtsev and Todd J. Martínez. Quantum chemistry on graphical processing units. 3. Analytical energy gradients, geometry optimization, and first principles molecular dynamics. *J. Chem. Theory Comput.*, 5(10):2619–2628, 2009.

[98] John E. Stone, David J. Hardy, Ivan S. Ufimtsev, and Klaus Schulten. GPU-accelerated molecular modeling coming of age. *J. Mol. Graph. Model.*, 29(2):116–125, SEP 2010.

[99] Alexey V. Titov, Ivan S. Ufimtsev, Nathan Luehr, and Todd J. Martinez. Generating Efficient Quantum Chemistry Codes for Novel Architectures. *J. Chem. Theory Comput.*, 9(1):213–221, JAN 2013.

[100] Leslie Vogt, Roberto Olivares-Amaya, Sean Kermes, Yihan Shao, Carlos Amador-Bedolla, and Alan Aspuru-Guzik. Accelerating resolution-of-the-identity second-order Moller-

Plesset quantum chemistry calculations with graphical processing units. *J. Phys. Chem. A*, 112(10):2049–2057, MAR 13 2008.

[101] Luigi Genovese, Matthieu Ospici, Thierry Deutsch, Jean-Francois Mehaut, Alexey Neelov, and Stefan Goedecker. Density functional theory calculation on many-cores hybrid central processing unit-graphic processing unit architectures. *J. Chem. Phys.*, 131(3), JUL 21 2009.

[102] Koji Yasuda. Accelerating density functional calculations with graphics processing unit. *J. Chem. Theory Comput.*, 4(8):1230–1236, AUG 2008.

[103] A. Eugene DePrince, III and Jeff R. Hammond. Coupled Cluster Theory on Graphics Processing Units I. The Coupled Cluster Doubles Method. *J. Chem. Theory Comput.*, 7(5):1287–1295, MAY 2011.

[104] Xin Wu, Axel Koslowski, and Walter Thiel. Semiempirical Quantum Chemical Calculations Accelerated on a Hybrid Multicore CPU-GPU Computing Platform. *J. Chem. Theory Comput.*, 8(7):2272–2281, JUL 2012.

[105] Mohamed Hacene, Ani Anciaux-Sedrakian, Xavier Rozanska, Diego Klahr, Thomas Guignon, and Paul Fleurat-Lessard. Accelerating VASP electronic structure calculations using graphic processing units. *J. Comput. Chem.*, 33(32):2581–2589, DEC 15 2012.

[106] Chenchen Song, Lee-Ping Wang, Torsten Sachse, Julia Preiss, Martin Presselt, and Todd J. Martinez. Efficient implementation of effective core potential integrals and gradients on graphical processing units. *J. Chem. Phys.*, 143(1), JUL 7 2015.

[107] Nir Goldman, Evan J. Reed, Laurence E. Fried, I. F. William Kuo, and Amitesh Maiti. Synthesis of glycine-containing complexes in impacts of comets on early Earth. *Nat. Chem.*, 2(11):949–954, 2010.

[108] A. M. Saitta and F. Saija. Miller experiments in atomistic computer simulations. *P. Natl. Acad. Sci. USA*, 111(38):13768–13773, 2014.

[109] Andrea Pérez-Villa, A. Marco Saitta, Thomas Georgelin, Jean François Lambert, François Guyot, Marie Christine Maurel, and Fabio Pietrucci. Synthesis of RNA Nucleotides in Plausible Prebiotic Conditions from ab Initio Computer Simulations. *J. Phys. Chem. Lett.*, 9(17):4981–4987, 2018.

[110] Emilio Martínez-Núñez. An automated method to find transition states using chemical dynamics simulations. *J. Comput. Chem.*, 36(4):222–234, 2015.

[111] Aurelio Rodríguez, Roberto Rodríguez-Fernández, Saulo A. Vázquez, George L. Barnes, James J. P. Stewart, and Emilio Martínez-Núñez. tsscds2018: A code for automated discovery of chemical reaction mechanisms and solving the kinetics. *J. Comput. Chem.*, 39(23):1922–1930, 2018.

[112] Fabio Pietrucci and Wanda Andreoni. Graph theory meets ab initio molecular dynamics: Atomic structures and transformations at the nanoscale. *Phys. Rev. Lett.*, 107(8), 2011.

[113] Christopher D. Fu and Jim Pfaendtner. Lifting the Curse of Dimensionality on Enhanced Sampling of Reaction Networks with Parallel Bias Metadynamics. *J. Chem. Theory Comput.*, 14(5):2516–2525, 2018.

[114] Tamal Das, Siddharth Ghule, and Kumar Vanka. Insights into the origin of life: Did it begin from hcn and h2o? *ACS Cent. Sci.*, 5(9):1532–1540, 2019.

[115] Jan Meisner, Xiaolei Zhu, and Todd J. Martínez. Computational discovery of the origins of life. *ACS Cent. Sci.*, 5(9):1493–1495, 2019.

[116] Lee Ping Wang, Robert T. McGibbon, Vijay S. Pande, and Todd J. Martinez. Automated Discovery and Refinement of Reactive Molecular Dynamics Pathways. *J. Chem. Theory Comput.*, 12(2):638–649, feb 2016.

[117] Malte Döntgen, Marie-Dominique Przybylski-Freund, Leif C. Kröger, Wassja A. Kopp, Ahmed E. Ismail, and Kai Leonhard. Automated discovery of reaction pathways, rate constants, and transition states using reactive molecular dynamics simulations. *J. Chem. Theory Comput.*, 11(6):2517–2524, 2015. PMID: 26575551.

[118] I. Mayer. Bond order and valence indices: A personal account. *J. Comput. Chem.*, 28(1):204–221, 2007.

[119] Hongyan Wang, Yaoming Xie, R. Bruce King, and Henry F. Schaefer. Remarkable aspects of unsaturation in trinuclear metal carbonyl clusters: The triiron species fe3(co)n (n = 12, 11, 10, 9). *J. Am. Chem. Soc.*, 128(35):11376–11384, 2006. PMID: 16939260.

[120] MR Sorensen and AF Voter. Temperature-accelerated dynamics for simulation of infrequent events. *J. Chem. Phys.*, 112(21):9599–9606, JUN 1 2000.

[121] Lisi Xie, Qing Zhao, Klavs F. Jensen, and Heather J. Kulik. Direct Observation of Early-Stage Quantum Dot Growth Mechanisms with High-Temperature Ab Initio Molecular Dynamics. *J. Phys. Chem. C*, 120(4):2472–2483, FEB 2016.

[122] Lee Ping Wang and Chenchen Song. Geometry optimization made simple with translation and rotation coordinates. *J. Chem. Phys.*, 144(21), 2016.

[123] Matthew P. Harrigan, Mohammad M. Sultan, Carlos X. Hernández, Brooke E. Husic, Peter Eastman, Christian R. Schwantes, Kyle A. Beauchamp, Robert T. McGibbon, and Vijay S. Pande. Msmbuilder: Statistical models for biomolecular dynamics. *Biophys. J.*, 112(1):10 – 15, 2017.

[124] Brooke E. Husic and Vijay S. Pande. Markov state models: From an art to a science. *J. Am. Chem. Soc.*, 140(7):2386–2396, 2018. PMID: 29323881.

[125] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright,

Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, CJ Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake Vand erPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1. 0 Contributors. SciPy 1.0–Fundamental Algorithms for Scientific Computing in Python. *arXiv e-prints*, page arXiv:1907.10121, Jul 2019.

[126] Ziv Bar-Joseph, David K. Gifford, and Tommi S. Jaakkola. Fast optimal leaf ordering for hierarchical clustering. *Bioinformatics*, 17(1):S22–S29, 06 2001.

[127] J A Hanley and B J McNeil. The meaning and use of the area under a receiver operating characteristic (ROC) curve. *Radiology*, 143(1):29–36, 1982.

[128] Tom Fawcett. An introduction to ROC analysis. *Pattern Recogn. Lett.*, 27(8):861–874, 2006.

[129] Brian Doherty and Orlando Acevedo. Opls force field for choline chloride-based deep eutectic solvents. *The Journal of Physical Chemistry B*, 122(43):9982–9993, Nov 2018.

[130] Kateryna Goloviznina, Zheng Gong, Margarida F. Costa Gomes, and Agílio A. H. Pádua. Extension of the cl&pol polarizable force field to electrolytes, protic ionic liquids, and deep eutectic solvents. *Journal of Chemical Theory and Computation*, 17(3):1606–1617, Mar 2021.

[131] Kateryna Goloviznina, Zheng Gong, and Agílio Pádua. The cl&pol polarizable force field for the simulation of ionic liquids and eutectic solvents. *WIREs Computational Molecular Science*, 08 2021.

[132] Lee-Ping Wang, Jiahao Chen, and Troy Van Voorhis. Systematic parametrization of polarizable force fields from quantum chemistry data. *Journal of Chemical Theory and Computation*, 9(1):452–460, Jan 2013.

[133] Lee-Ping Wang, Teresa Head-Gordon, Jay W. Ponder, Pengyu Ren, John D. Chodera, Peter K. Eastman, Todd J. Martinez, and Vijay S. Pande. Systematic improvement of a classical molecular model of water. *The Journal of Physical Chemistry B*, 117(34):9956–9972, Aug 2013.

[134] Lee-Ping Wang, Todd J. Martinez, and Vijay S. Pande. Building force fields: An automatic, systematic, and reproducible approach. *The Journal of Physical Chemistry Letters*, 5(11):1885–1891, Jun 2014.

[135] Yudong Qiu, Benedict R. Schwegler, and Lee-Ping Wang. Polarizable molecular simulations reveal how silicon-containing functional groups govern the desalination mechanism in nanoporous graphene. *Journal of Chemical Theory and Computation*, 14(8):4279–4290, Aug 2018.

[136] Pengyu Ren, Chuanjie Wu, and Jay W Ponder. Polarizable atomic multipole-based molecular mechanics for organic molecules. *J. Chem. Theory Comput.*, Oct 11 2011. https://doi.org/10.1021/ct200304d.

[137] Xiongwu Wu. A double exponential potential for van der waals interaction. *AIP Adv.*, 2019.

[138] Rui Qi, Qiantao Wang, and Pengyu Ren. General van der waals potential for common organic molecules. *Bioorg Med Chem*, 24(20):4911–4919, August 2016.

[139] Thomas A. Halgren. The representation of van der waals (vdw) interactions in molecular mechanics force fields: potential form, combination rules, and vdw parameters. *Journal of the American Chemical Society*, 114(20):7827–7843, Sep 1992.

[140] Joshua A Rackers and Jay W Ponder. Classical pauli repulsion: An anisotropic, atomic multipole model. *The Journal of chemical physics*, 150(8):084104, February 2019.

[141] Akshaya K. Das, Omar N. Demerdash, and Teresa Head-Gordon. Improvements to the amoeba force field by introducing anisotropic atomic polarizability of the water molecule. *Journal of Chemical Theory and Computation*, 14(12):6722–6733, Dec 2018.

[142] Changsheng Zhang, Chao Lu, Zhifeng Jing, Chuanjie Wu, Jean-Philip Piquemal, Jay W. Ponder, and Pengyu Ren. Amoeba polarizable atomic multipole force field for nucleic acids. *Journal of Chemical Theory and Computation*, 14(4):2084–2108, Apr 2018.

[143] Yue Shi, Zhen Xia, Jiajing Zhang, Robert Best, Chuanjie Wu, Jay W Ponder, and Pengyu Ren. The polarizable atomic multipole-based AMOEBA force field for proteins. *J Chem Theory Comput*, 9(9):4046–4063, 2013.

[144] Joshua A. Rackers, Zhi Wang, Chao Lu, Marie L. Laury, Louis Lagardère, Michael J. Schnieders, Jean-Philip Piquemal, Pengyu Ren, and Jay W. Ponder. Tinker 8: Software Tools for Molecular Design. *Journal of Chemical Theory and Computation*, 14(10):5273–5289, October 2018.

[145] M. J. Frisch, G. W. Trucks, H. B. Schlegel, G. E. Scuseria, M. A. Robb, J. R. Cheeseman, G. Scalmani, V. Barone, G. A. Petersson, H. Nakatsuji, X. Li, M. Caricato, A. V. Marenich, J. Bloino, B. G. Janesko, R. Gomperts, B. Mennucci, H. P. Hratchian, J. V. Ortiz, A. F. Izmaylov, J. L. Sonnenberg, D. Williams-Young, F. Ding, F. Lipparini, F. Egidi, J. Goings, B. Peng, A. Petrone, T. Henderson, D. Ranasinghe, V. G. Zakrzewski, J. Gao, N. Rega, G. Zheng, W. Liang, M. Hada, M. Ehara, K. Toyota, R. Fukuda, J. Hasegawa, M. Ishida, T. Nakajima, Y. Honda, O. Kitao, H. Nakai, T. Vreven, K. Throssell, J. A. Montgomery, Jr., J. E. Peralta, F. Ogliaro, M. J. Bearpark, J. J. Heyd, E. N. Brothers, K. N. Kudin, V. N. Staroverov, T. A. Keith, R. Kobayashi, J. Normand, K. Raghavachari, A. P. Rendell, J. C. Burant, S. S. Iyengar, J. Tomasi, M. Cossi, J. M. Millam, M. Klene, C. Adamo, R. Cammi, J. W. Ochterski, R. L. Martin, K. Morokuma, O. Farkas, J. B. Foresman, and D. J. Fox. Gaussian˜16 Revision C.01, 2016. Gaussian Inc. Wallingford CT.

[146] Stone AJ. Distributed multipole analysis Stability for large basis sets. *J Chem Theory Comput*, pages 1128–32, November 2005.

[147] Noel M O'Boyle, Michael Banck, Craig A. James, Chris Morley, Tim Vandermeersch, and Geoffrey R. Hutchison. Open Babel: An open chemical toolbox. *J. Cheminf.*, 3, Oct 7 2011.

[148] The Open Babel Package. http://openbabel.org. accessed jan 2014, 2014.

[149] G. van Rossum. Python tutorial. Technical Report CS-R9526, Centrum voor Wiskunde en Informatica (CWI), Amsterdam, May 1995.

[150] Avogadro: an open-source molecular builder and visualization tool. http://avogadro.cc/.

[151] Marcus D Hanwell, Donald E Curtis, David C Lonie, Tim Vandermeersch, Eva Zurek, and Geoffrey R Hutchison. Avogadro: An advanced semantic chemical editor, visualization, and analysis platform. *J. Cheminf.*, 2012. https://doi.org/10.1186/1758-2946-4-17.

[152] Gdma manual version 2.3.0. https://www-stone.ch.cam.ac.uk/documentation/gdma/manual.pdf.

[153] Johnny C Wu, Gaurav Chattree, and Pengyu Ren. Automation of amoeba polarizable force field parameterization for small molecules. *Theor Chem Acc*, 2012.

[154] Brandon Walker, Chengwen Liu, Elizabeth Wait, and Pengyu Ren. Automation of amoeba polarizable force field for small molecules: Poltype 2. *Journal of Computational Chemistry*, 43(23):1530–1542, 2022.

[155] Tinker user's guide, 2022. https://dasher.wustl.edu/tinker/downloads/tinker-guide.pdf.

[156] P. Nikitas. The origin of the polarization catastrophe: the improper choice of the independent electrical variable. *Journal of Electroanalytical Chemistry and Interfacial Electrochemistry*, 306(1):13–29, 1991.

[157] Edwin T. Jaynes. Prior probabilities. *Ieee Transactions on Systems and Cybernetics*, pages 227–241, 1968.

[158] Lee-Ping Wang, Keri A. McKiernan, Joseph Gomes, Kyle A. Beauchamp, Teresa Head-Gordon, Julia E. Rice, William C. Swope, Todd J. Martínez, and Vijay S. Pande. Building a more predictive protein force field: A systematic and reproducible route to amber-fb15. *The Journal of Physical Chemistry B*, 121(16):4023–4039, Apr 2017.

[159] Peter Eastman, Jason Swails, John D. Chodera, Robert T. McGibbon, Yutong Zhao, Kyle A. Beauchamp, Lee-Ping Wang, Andrew C. Simmonett, Matthew P. Harrigan, Chaya D. Stern, Rafal P. Wiewiora, Bernard R. Brooks, and Vijay S. Pande. Openmm 7: Rapid development of high performance algorithms for molecular dynamics. *PLOS Computational Biology*, 13(7):1–17, 07 2017.

[160] V. Pande and P. Eastman. Openmm: A hardware-independent framework for molecular simulations. *Computing in Science & Engineering*, 12(04):34–39, jul 2010.

[161] Matthew M. Copeland, Hung N. Do, Lane Votapka, Keya Joshi, Jinan Wang, Rommie E. Amaro, and Yinglong Miao. Gaussian accelerated molecular dynamics in openmm. *The Journal of Physical Chemistry B*, 126(31):5810–5820, Aug 2022.