EXPLORING CRITICAL CONFORMATIONS:

STATE SEARCHING AND SAMPLING IN BOTH

GERMANIUM CHAINS AND ICE INTERFACES

{EARLY DRAFT}

By

GENTRY H. SMITH

B.S.

Southern Nazarene University

Bethany, OK, USA

2016

Submitted to the Faculty of the
Graduate College of
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
Master of Science
November 2018

EXPLORING CRITICAL CONFORMATIONS:

STATE SEARCHING AND SAMPLING IN BOTH

GERMANIUM CHAINS AND ICE INTERFACES

{EARLY DRAFT}

Thesis Approved:

_____

Thesis Advisor

_____

_____

ACKNOWLEDGMENTS

ACKNOWLEDGMENTS

To Oklahoma State University, for providing the environment in which I have been able to study, teach, and research.

To the HPCC and the individuals within **(get first/last names)** for providing a powerful cluster for computations and continuous support for technical issues.

To my advisor, who instructed and assisted me in research

To my parents, by blood and marriage, who have always encouraged me toward higher goals.

To my wife, Miranda, who has supported me for over 5 years.

Name: GENTRY H SMITH

Date of Degree: November 2018

Title of Study: EXPLORING CRITICAL CONFORMATIONS

Major Field: COMPUTATIONAL CHEMISTRY

Abstract: Molecular conformation plays a critical role in the properties of systems in either the condensed or vapor states. The ensemble of conformations dictates structural properties, energies, heat capacities, and other thermodynamic and dynamic quantities. Here, we explore the role of conformation in proton ordering and orientational defect formation in ice as well as strategies for exhaustive conformer searching for molecules using Group IV element backbones. In the ice systems, we show algorithmic strategies for seeking optimized proton disordered crystals that satisfy the Bernal-Fowler ice rules. In the Group IV molecule investigations, we develop an automated strategy for seeking the optimal low energy conformer and uncover previously unreported deficiencies in common computational software used in investigating Germanium complex energies.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

NOMENCLATURE

Variables

ϵ .... Lennard-Jones Potential well depth
r .... Lennard-Jones Potential distance between center of two particles
σ .... Lennard-Jones Potential intermolecular contact distance
V .... Lennard-Jones Potential intermolecular potential

Subscripts/Superscripts

0 .... Initial condition

CHAPTER 1

Introduction

I want a first paragraph that inspires the reader to continue reading. Maybe something with a quote or a question. Maybe not.

The idea for an introduction is to interest the reader and provide general background information. For my work, the interesting part is how impactful computational science (specifically chemistry) has been in research and on society. The background information will start extremely generic and then go into some overarching themes. Generically, I'll include the development of computational science in the 20th century and the vast applications of computational chemistry specifically. Overarching themes are related, but don't have to be explicitly relevant. For example, Levinthal's paradox is a fun example showing the problem of conformation landscape searches. Also,

1.1 Reason for Study

The studies conducted that comprise this work were determined as a combination of collaborative efforts and larger research group goals with novel discoveries worth reporting. Collectively, they explore conformations of internal bond dihedrals, molecule orientations of microstates, and properties of interfacing substrates. These efforts are categorized and separated into three categories: ice crystal states, conformation landscapes, and ice interfaces. A brief introduction of each and a literature review of relevant information is given below.

1.1.1 Generation of Ice I$_h$ Crystal Structure

Ice crystals can take many forms based on properties like temperature and pressure. The proton-disorganized orthorhombic form of ice known as I$_h$ is the form of ice most commonly found on earth (general understanding, CITATION NEEDED). Due to the inherent randomness of the disorganization of the molecules within the crystal, computational efforts are limited in scope or instead utilize the proton-ordered orthorhombic form of ice XI (general understanding, CITATION NEEDED). This project explores a method to produce a high quality pseudorandom ice I$_h$ crystal structure.

1.1.2 Conformation Landscapes of Group IV Chains

Any molecule with a chain length of at least four contains at least one dihedral. In small molecules, the steric hindrance between the head and tail atoms are usually minimized in the fully gauche conformation to produce the lowest-energy conformer. In larger and more bulky molecules, additional interactions may cause the dihedral to take other conformations in search of the lowest-energy conformation. This project details the search for the lowest-energy conformer of a bulky hexagermane molecule in collaboration with Oklahoma State University's Charles Weinert and the complications and curiosities found within.

1.1.3 Ice Interfaces in Two Dimensions

Ice is capable of interfacing with hydrophobic and amphipathic molecules. These interfaces often have a specific range of polarity and **physicochemical** properties like atom type, charge, and the spacing of interfaces. While usually considered in the biochemical sense, molecular interfaces with water have additional applications. If, for example, an interface had the properties to encourage ice growth, then it may be possible to observe significant ice growth above the expected freezing temperature.

Modeling water systems can be computationally intensive and require significant resources or a limited scope of system. One proven method of simplifying water modeling is to reduce the system by one dimension, creating a two-dimensional water. The 'Mercedez Benz' BN2D model and the more recent rose potential model are two examples of two-dimensional water models. Exploring and studying a two-dimensional water system allows for computationally-efficient investigating and can provide relevant information for a similar simulation with three-dimensional water. This project covers the usage of the two-dimensional rose potential model to observe ice growth on a substrate with varying properties.

1.2 Ice Annealing

EACH SUBSECTION: DEFINITION OF TERMS

1.2.1 Bernal - Fowler Ice Rules

Citation! Bernal and Fowler (1933)

There have been many works published on this topic, so I will have no problem obtaining a cohesive review. I plan to include the various forms of ice and focus on I_h and XI. This includes differences between the two in prevalence, environment, and synthesis (is synthesis the correct term here?).

I expect to focus on many thermodynamic properties and to focus on entropy as a driving force of difference in the modeled system. I will include publications of the effort to model a truly proton-disordered system and focus on the computational aspect.

1.2.2 Structures of Ice

Ice contains many structures. These structures are typically orthorhombic, which means rectangular at non-90 angles. They can also be hexagonal, cubic, else. These

different structures form based on temperature and pressure. Ice I_h is the most prevalent form of ice found normally on earth, forming at pressures around 1 atm and temperatures around 0C. (source needed?).

More detail on ice I_h.

More detail on ice XI.

1.2.3 Residual Entropy of Ice I_h and Ice XI

Residual entropy is a thermodynamic property that greatly differs between ice I_h and XI. In general, entropy can be calculated for a system of N molecules as $S = Nk \ln(w)$, where k is the Boltzmann constant and w is the number of real microstates corresponding to any macrostate. Residual entropy differs in calculation from entropy in that it generally refers to the entropy of a crystal near zero kelvin. Linus Pauling (1935) described the w of very low temperature crystals to approach the number of orientations possible for each molecule with consideration to immediate neighbors. Since ice I_h is a

For a proton-disordered ice I_h crystal, w becomes $\frac{x}{2^4}$ where x is the number of acceptable orientations within the crystal.

1.2.4 Hydrogen Bond Defects in Ice Crystals

Speak here about defects and how they quantitatively harm stability and why the defects need to be reduced in general ice I_h structures.

1.2.5 Literature Review on Relevant Works

1.3 Conformation Landscapes

EACH SUBSECTION: DEFINITION OF TERMS

For ROUGHLY forty years, computational programs have allowed investigators to model chemical systems with high accuracy to determine their physical properties.

1.3.1 A Brief History of Conformation Landscapes

1.3.1.1 Levinthal's Paradox

Discuss history of Levinthal and his paradox. Provide the non-paradoxical solution.

Next: Levinthal golf courses by Ken Dill.

1.3.2 Computational Modeling

Introduce importance and impact. Bigly important.

1.3.2.1 History of Chemical Modeling

Here I will introduce molecular modeling. This will begin with a brief history of the development of the field. It will continue through to mention the styles and goals of molecular modeling. Upon reaching modern techniques, I will discuss the benefits and costs associated with the major types of calculations (QM, MD, MC, etc).

1.3.2.2 Hardware: Oklahoma State University's Cowboy Cluster

It would also be appropriate to mention the computational capabilities of OSU's Cowboy cluster.

1.3.2.3 Software

Software to mention:

VisualizingAvogadro, UCSF Chimera(?),

ComputingGAMESS(?), Gaussian, NWChem(?), OOPSE

This also includes brief pros and cons about the programs and the general purpose of use in case.

1.3.2.4 Programming Languages

Another hugely important portion of this will include a choice in the programming languages used (mostly python (Cython-compiled!), some Perl and Bash).

1.3.3 Literature Review on Relevant Works

1.4 Modeling Germanium Compounds

EACH SUBSECTION: DEFINITION OF TERMS

This will be an interesting section as there is extremely little in terms of Ge computational work. Perhaps a broader search will yield interesting results. For sake of thoroughness, I will also include work on computational energy optimization in general and work through complications brought by the size of Ge. I might also include a portion on the statistical spread of conformations at a given temperature (internal energy?) I may include a sentence or paragraph on Gaussian-based publications.

1.4.1 Tools for Modeling Germanium

Computational Requirements and reasons for those requirements.

Germanium is not the most-studied atom in computational works. The majority of Germanium studies are done with Gaussian (citation needed?).

1.4.2 Computational Complexity of Germanium Compounds

Draw-backs of modeling Germanium. Uncommon but still necessary for wetwork.

1.4.3 Literature Review on Relevant Works

Make note of various Germanium modeling research. Make note of tools and methods used.

1.5 Two-Dimensional Water

EACH SUBSECTION: DEFINITION OF TERMS

Rose water is fairly new on the computational scene and so I may also include a review on the Mercedez-Benz water system as well as any other attempts to model water in two dimensions. For the rose potential system, I will review the Lennard-Jones potential as well as any other equations/systems related to the rose potential.

1.5.1 Lennard-Jones Potential

The Lennard-Jones Potential well is a soft-sphere model of interaction between two spheres described with

$$V_{LJ} = 4\epsilon[(\frac{\sigma}{r})^{12} - (\frac{\sigma}{r})^{6}] \tag{1.1}$$

where V is the potential, r is the distance between the center of two particles , σ is the specific distance between the two particles where the potential is zero, and $-\epsilon$ is the minimum potential of the plot. **REFINE WORDING:** The plot is defined in $[0, \infty)$. As two particles approach from infinity, their interaction become negative - which is an attractive force - and will approach the global minimum of $-\epsilon$. The r of this interaction is slightly larger than the combined radii of the two particles - which means they aren't quite touching - and is the equilibrium distance between the two particles. As r decreases beyond the minimum and toward σ, the interaction strength increases and reaches zero as $r = \sigma$. At $r < \sigma$,

Potential digression: In a "hard-sphere" model, a particle's radius is firm, which is to say that the interaction potential is infinite at r less than σ. Basically a vertical line between two discrete values (usually ϵ and ∞)as the potential shifts from $r \geq \sigma$ to $r < \sigma$ (maybe include image?). The Lennard-Jones potential is a "soft-sphere" model, which blurs the line and replaces the vertical line with a functional representation.

This breaks with reality as the particles become "squishy" and the potential ramps up toward infinity as r decreases. The benefit to the soft-sphere model is that modeling programs can more-easily account for overlaps in particles during time steps than with hard-sphere models. For example, a hard-sphere model of two particles interacting will likely not have a position where $r = \sigma$ and will potentially overlap. At this overlap, the potential is infinity and will introduce a nearly infinite force at that instant of time. Computer systems do not like having infinitely large repulsions suddenly introduced into a simulation.

1.5.2 Modeling Water in Two Dimensions

Modeling in two dimensions sacrifice the "realism" of models in three dimensions, but reduce the computational load significantly. This allows researchers (scientists, chemists, digital magicians?) to test more simple designs in two dimensions as well as a higher volume of simulations at the same time/computational cost.

1.5.2.1 Mercedes-Benz Model

The "Mercedes-Benz" BN2D model of water first proposed by Ben-Naim (1971) as "waterlike particles" are a popular two-dimensional representation of water. ROUGH: details of shape of MB water

The mathematical model used in the BN2D model is generated from the Percus-Yevick equation by substituting the approximation

$$c(X_1, X_2) = y(X_1, X_2)f(X_1, X_2) \tag{1.2}$$

into the Percus-Yevick equation obtained from the Ornstein-Zernike relation

$$h(X_1, X_2) = c(X_1, X_2) + \frac{\rho}{2\pi} \int c(X_1, X_3)h(X_3, X_2)dX_3 \tag{1.3}$$

to produce the overall relation

$$y(X_1, X_2) = 1 + \frac{\rho}{2\pi} \int y(X_1, X_3) f(X_1, X_3) \times \Big[y(X_3, X_2) f(X_3, X_2) + y(X_3, X_2) - 1 \Big] dX_3$$

(1.4)

1.5.2.2 Rose Potential Model

The rose potential is another model first introduced by Williamson et al. (2017). This model, while similar to the three-pronged BN2D, is notably different in that the rose potential model simplifies the model by use of a radial sinusoidal plot to make the three "prongs" of the particle.

1.5.2.3 Two-Dimensional Modeling

Something other than OOPSE? (not seeing obvious answer other than "custom code modified/forked from existing 3D tools")

The Object Oriented Parallel Simulation Engine (OOPSE) was introduced by A. et al. (n.d.) as a relatively light-weight molecular dynamics simulation package focused on "efficiently integrating equations of motion for atom types with orientational degrees of freedom" (from abstract). While OOPSE was further developed and renamed OpenMD, a fork of OOPSE was developed specifically to model water in two dimensions.

1.5.3 Literature Review on Relevant Works

TODO: Work with MB water,

CHAPTER 2

On Algorithms for Building and Sampling Disordered Crystal States

TODO: FIX ICE IH - I_h As discussed in the introduction, ice I_h modeling comes with computational challenges. The problem of generating a random proton-disordered crystal with only ordered pseudorandom tools presents as a technical impossibility. However, through the usage of averages and distributions, it is possible to generate a pseudorandom proton-disordered crystal that behaves similar to a truly random proton-disordered crystal.

2.1   Ice Annealing

EACH SUBSECTION: DEFINITION OF TERMS

2.1.1   Bernal - Fowler Ice Rules

Citation! Bernal and Fowler (1933)

There have been many works published on this topic, so I will have no problem obtaining a cohesive review. I plan to include the various forms of ice and focus on I_h and XI. This includes differences between the two in prevalence, environment, and synthesis (is synthesis the correct term here?).

I expect to focus on many thermodynamic properties and to focus on entropy as a driving force of difference in the modeled system. I will include publications of the effort to model a truly proton-disordered system and focus on the computational aspect.

2.1.2 Structures of Ice

Ice contains many structures. These structures are typically orthorhombic, which means rectangular at non-90 angles. They can also be hexagonal, cubic, else. These different structures form based on temperature and pressure. Ice I_h is the most prevalent form of ice found normally on earth, forming at pressures around 1 atm and temperatures around 0C. (source needed?).

More detail on ice I_h.

More detail on ice XI.

2.1.3 Residual Entropy of Ice I_h and Ice XI

Residual entropy is a thermodynamic property that greatly differs between ice I_h and XI. In general, entropy can be calculated for a system of N molecules as $S = Nk\ln(w)$, where k is the Boltzmann constant and w is the number of real microstates corresponding to any macrostate. Residual entropy differs in calculation from entropy in that it generally refers to the entropy of a crystal near zero kelvin. Linus Pauling (1935) described the w of very low temperature crystals to approach the number of orientations possible for each molecule with consideration to immediate neighbors. Since ice I_h is a

For a proton-disordered ice I_h crystal, w becomes $\frac{x}{2^4}$ where x is the number of acceptable orientations within the crystal.

2.1.4 Hydrogen Bond Defects in Ice Crystals

Speak here about defects and how they quantitatively harm stability and why the defects need to be reduced in general ice I_h structures.

2.1.5 Literature Review on Relevant Works

words????

2.2 Framing the Problem

While ice I_h is known as the most common form of ice found on the planet, it is much more difficult to generate than an ice XI lattice. The ease of generation of an ice XI structure stems from the repetition of a unit cell with consistent layering and orientation throughout the crystal lattice. All protons and lone pairs are ordered in a consistent manner that produce proper hydrogen bonds. As such, lattice defects do not naturally arise in the computational structure without running a simulation with sufficient energy to perturb the system.

With ice I_h crystals, the proton-disordered form introduces entropy by way of rotational disorder. As the protons and lone pairs are no longer consistently ordered, hydrogen bonds may no longer form properly at all interaction sites. The interaction of proton with proton or lone pair with lone pair are not hydrogen bonds and are considered defects in the lattice. An ice structure of randomly oriented molecules without consideration of hydrogen bonds will likely produce defects at many interaction sites across the lattice and weaken the integrity of the system, leading to stability problems while running simulations. In generating the crystal, the cause of these defects must be considered and countered effectively.

2.3 Method of Generating an Ice I_h Crystal

This approach to generating an ice I_h structure involves beginning with a proton-ordered ice XI structure, randomly rotating all water molecules, and then recursively perturbing high-defect regions to reduce the number of overall defects. As detailed later, defect thresholds can be adjusted to fit the desired system.

2.3.1 Generation of an ice XI crystal

TODO This details Dr. Fennells method of making an ice XI crystal.

2.3.2 Algorithm to rotate water molecules (TODO Code inside?)

Once the ice XI PDB file has been generated, the next goal is to introduce disorder into the lattice by rotating each water molecule. Using the Euler-Rodrigues formula for rotating a three-dimensional point about a vector (detailed below), the water molecule is rotated about the center oxygen atom along the vector of one proton to determine the four spatial potential locations for the two protons. These locations are stored in a water object variable for later potential use. Using a pseudo-random number generator, two of the four location potentials are selected and the PDB file is modified so that the protons occupy those locations. The lone pairs are then implied to occupy the two remaining spaces. As PDB files do not specify lone pair location, no action is taken to specify this. Considering each hydrogen bond location individually, the six possible molecule orientations will yield three orientations each of a proton-occupied or lone pair-occupied location. Assuming the rotation is close-enough to random, the probability of occupation by either type is

$3/6 = 1/2$

While considering each molecule as a whole, (12 distinguishable, 6 indistinguishable)

Because there are six potential orientations of the indistinguishable protons in the four locations each containing three proton-facing and three lone pair-facing orientations, the molecule stands a $2/6 = 1/3$ chance of occupying the same orientation after the rotation. After each molecule is randomly rotated, the molecule likely contains defects at half of all junctions due to the $1/2$ chance of any one junction containing either a proton - proton or lone pair - lone pair interaction. These defects introduce instability in the lattice and need to be reduced.

2.3.2.1 The Euler-Rodrigues Formula

The Euler-Rodrigues Formula is a method of rotating a three-dimensional object in space by use of four-dimensional quaternions. Specifically, it has been used for object rotation in video games for years due to its computational efficiency and analytic accuracy.

https://en.wikipedia.org/wiki/Euler

2.3.2.2 Center of Rotation for Water Molecules

During the rotation, the center of rotation was set to be the center of the oxygen atom. The center of mass of the water molecule is toward the two hydrogens (calculate?). The crystal structure isnt COM-based. Its based on the hydrogen bonds and those line up as an extention from the center of each oxygen atom through each hydrogen atom toward a neighboring oxygen atom. Because of that connection, the rotation formula uses the center of the oxygen atom as the basis for molecule rotation.

2.3.3 Determining Crystal Defects

Crystal defects are determined by distance between central oxygen and neighboring oxygen atoms between two water molecules. Specifically, the squared distance is calculated to remain computationally efficient. This value contains all the pertinent distance information, but squared to prevent the hardware from computing a large number of relatively costly square root calculations. If an oxygen atom contains a hydrogen atom in the direction of the neighboring hydrogen atom (or lone pair to lone pair), then a defect is recognized and counted. The method utilized to count this is to compare the squared distance between two oxygen atoms with the squared distances between each oxygen atom and neighboring proton of every two neighboring water molecules. The neighboring water molecules are determined by calculating the squared distance between oxygen atoms and storing the four closest values.

NEED TO ACTUALLY DO: Recognize edges by wrapping crystal in periodic boundary conditions, recognize neighbor (HOW DO?).

Due to the 50% likelihood of a defect existing at an interface, the average defect count for all internal water molecules will be near 2. This can easily be confirmed by iterating through all internal water molecules, counting the defective interfaces, and dividing the value by 2 times the number of internal molecules.

$$\text{DefectsAverage,Internal} = \text{DefectsInternal (2.1) 2 NInternalMolecules}$$

The method allows for a variable average number of defects to be considered and input as a value by the user. Future work may also include a maximum number of defects per molecule.

2.3.4 Removing Defects from the Lattice

This is ongoing, but basically it randomly rotates to reduce the number of defects. If impossible, then just move on. Also mention the criteria of deciding the number of defects allowed as well as the average defect count (per molecule, not bond pairs). Not yet perfect!!!

If the calculated average number of defects is greater than the specified maximum, then actions need to be taken to remove defects without sacrificing the disordered nature of the molecule. The method repeats the random rotation mentioned earlier for every molecule and redetermines the defect count for each molecule. After each molecule is rotated the new defect count is determined. If the defect value is reduced, the rotation is kept. If it is equal to or increases the defect prior to the rotation, then the rotation is discarded. Once the system has completed iterating through the lattice, the average defect count is redetermined. If it remains above the threshold, then the process outlined above is repeated until the criteria is met.

2.4 Results of Approach

Initial results show a successful rotation of water molecules in the lattice to disorder the protons and successfully reclassify the system as ice I_h. An example ice XI lattice of 432 water molecules converted to an ice Ih lattice is shown in figures 2.1 and 2.2.

Figure 2.1: Ice XI Lattice Before Rotation Figure 2.2: Ice Ih Lattice After Rotation

2.4.1 Difficulties of Method / Known Issues

Upon closer inspection of figure **??**, there appears to be fragments of order within layers. Curiously, multiple regions of repetitive ordered fragments are in different orientations from the original structure. Due to the nature of the pseudo-random orientation and the defect removal, molecules rotated to the original orientation may encourage neighboring high-defect molecules to rotate back to their original rotation or to a new pseudo-ordered orientation for multiple layers in localized fragments. Additional study is necessary to confirm this. Some inherent order may be retained by the method and might become a focus of future work. With the caveat of some retained order, the method successfully and consistently rotates the provided structure in a pseudo-random fashion.

A known issue exists where the user-specified average-defect value can cause an infinite loop where the method approaches, but does not achieve, the desired average defect count.

2.4.2 Suggested Improvements for Future Work

Future work on this method can improve the code base of the method as well as utilize thermodynamic values such as entropy and chemical potential to better generate a proton-disordered lattice. The calculation of entropy in regions within the lattice may specifically help reduce the proton-ordered fragments.

CHAPTER 3

Sampling Conformation Landscapes by Rotatable Bond Degrees of Freedom

3.1 Conformation Landscapes

EACH SUBSECTION: DEFINITION OF TERMS

For ROUGHLY forty years, computational programs have allowed investigators to model chemical systems with high accuracy to determine their physical properties.

3.1.1 A Brief History of Conformation Landscapes

3.1.1.1 Levinthal's Paradox

Discuss history of Levinthal and his paradox. Provide the non-paradoxical solution.

Next: Levinthal golf courses by Ken Dill.

3.1.2 Computational Modeling

Introduce importance and impact. Bigly important.

3.1.2.1 History of Chemical Modeling

Here I will introduce molecular modeling. This will begin with a brief history of the development of the field. It will continue through to mention the styles and goals of molecular modeling. Upon reaching modern techniques, I will discuss the benefits and costs associated with the major types of calculations (QM, MD, MC, etc).

3.1.2.2 Hardware: Oklahoma State University's Cowboy Cluster

It would also be appropriate to mention the computational capabilities of OSU's Cowboy cluster.

3.1.2.3 Software

Software to mention:

VisualizingAvogadro, UCSF Chimera(?),

ComputingGAMESS(?), Gaussian, NWChem(?), OOPSE

This also includes brief pros and cons about the programs and the general purpose of use in case.

3.1.2.4 Programming Languages

Another hugely important portion of this will include a choice in the programming languages used (mostly python (Cython-compiled!), some Perl and Bash).

3.1.3 Literature Review on Relevant Works

intro: Reference introduction: inherent complexity of "objective" or "exhaustive" search, levinthal's paradox as it applies to non-biological systems, ELSE?

3.2 Approach to Solution: Variable-Resolution Search

Run through a set of dihedral positions at a constant interval. Selection of lowest-energy optimization organized on dihedral values. Quick determination of importance of dihedral based on how heavily it impacts internal energy. Splitting "best" dihedral into smaller interval to repeat the process.

This method produces an interesting visual plot with varying resolution at different X values (if plotting energy vs. dihedral). Example given on 3.1.
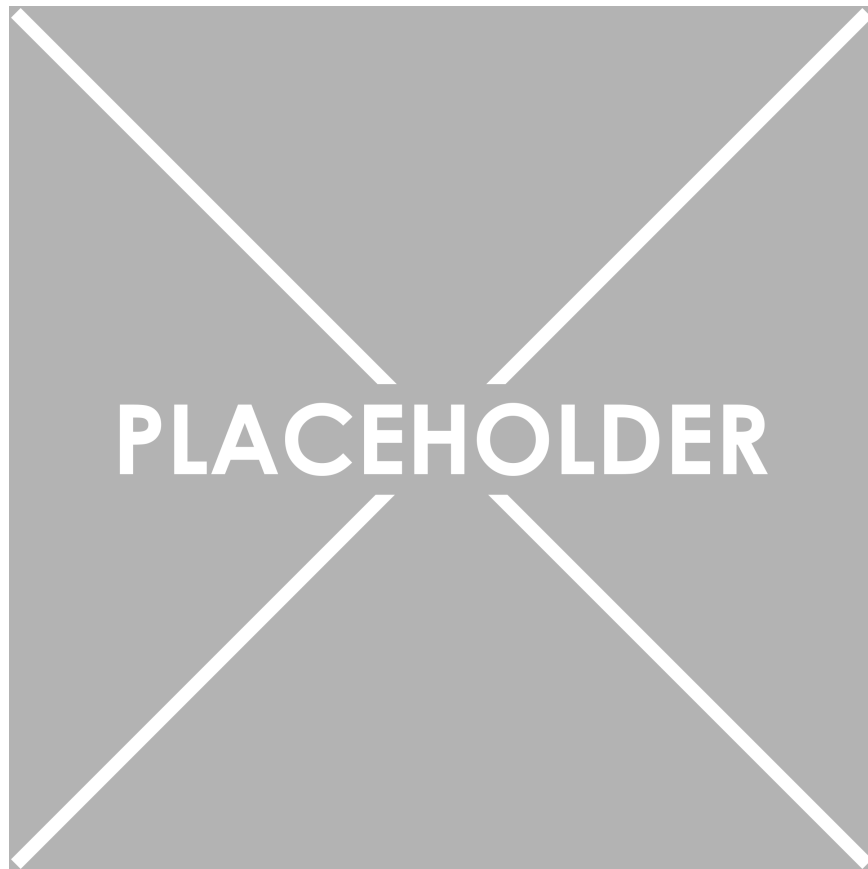
Figure 3.1: example VR chart (exploding like molecule/cell/fiber/muscle)
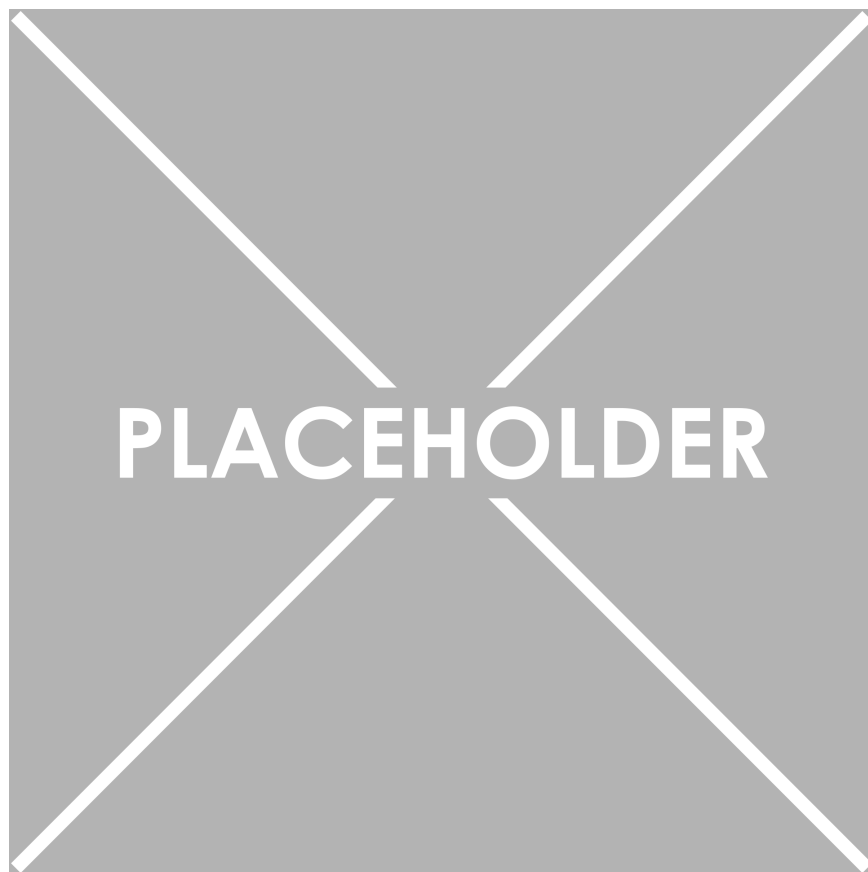
Figure 3.2: Flow of method design for variable resolution conformation landscape search.

3.2.1 Design of System

System designed in Python for ease of development and compiled via Cython for computational efficiency. Utilizes Gaussian and UCSF Chimera, but can be redesigned for any computational programs that accomplish the desired tasks. Overview of system flow given in 3.2.

This design, with implementation being a current work in progress, but should™ work as a cascade toward the lowest energy conformation in each case.

3.2.1.1 Variation of Theory and Basis Set Usage by System Size and largest atom type

System will have inherent restrictions. Give an example of large system with simple atoms, small system with complex atoms. System estimates quantity and cost of calculations based on computational limits defined by user for various theory-basis set pairings. System optimizes calculations for the scale of run (is it the first broad-scope search, or a final near-exact search).

Good to have: data on how theory and basis set alter computational requirement. Available in literature? Create data from runs?

3.2.1.2 Computational Optimization by Varying Resolution

Extant work not optimized for a general search (negative claim: make sure literature has nothing). Design should work with additional development (primary focus this semester) as a general search tool.

3.2.2 Inherent Complications

Complications of size and atom type, impossible conformations, duplications, limited computational resources, GERMANIUM.

3.3 Results

Current success: finding accepted lowest energy conformer of a two-dihedral system by manually cranking each step. Self-running is still a work in progress.

3.3.1 Problems

Difficulty in defining an abstract system based on arbitrary hardware limitations. Propose a test-run to determine efficiency and resource availability.

3.3.2 Anticipated Approaches for Future Work

Putting system into a single cohesive program. Further optimizing Theory/Basis Set determination by computational efficiency as well as system size (determine an upper-limit of computation?)

CHAPTER 4

Germanium Compounds and QM Concerns

4.1   Modeling Germanium Compounds

EACH SUBSECTION: DEFINITION OF TERMS

This will be an interesting section as there is extremely little in terms of Ge computational work. Perhaps a broader search will yield interesting results. For sake of thoroughness, I will also include work on computational energy optimization in general and work through complications brought by the size of Ge. I might also include a portion on the statistical spread of conformations at a given temperature (internal energy?) I may include a sentence or paragraph on Gaussian-based publications.

4.1.1   Tools for Modeling Germanium

Computational Requirements and reasons for those requirements.

Germanium is not the most-studied atom in computational works. The majority of Germanium studies are done with Gaussian (citation needed?).

4.1.2   Computational Complexity of Germanium Compounds

Draw-backs of modeling Germanium. Uncommon but still necessary for wetwork.

4.1.3   Literature Review on Relevant Works

Make note of various Germanium modeling research. Make note of tools and methods used.

Table 4.1: Collaborator's Hexagermane Energies by Conformation
(Density Function Theory, Unknown Basis Set)

Conformation	Energy (E_h)	Δ Energy (E_h)	Δ Energy $(\frac{Kj}{mol})$
Trans-coplanar	-15014.8403143	0.0066255	17.39525025
Cis-Trans-Cis	-15014.7983311	0.0486087	127.6221418
Trans-Cis-Trans	-15014.8469398	0.0000000	0
Cis-Trans-Trans	-15014.8246918	0.0222480	58.412124

In the fall of 2016, the Fennell Group was contacted by Dr. Scott Weinert to study a germanium-based compound comprised of a backbone hexagermane chain, two phenyl rings off the internal atoms, and isopropyl groups on the terminal germanium atoms (hereafter referred to as "hexagermane"). Dr. Weinert's lab had received curious results from a prior collaborator and sought confirmation by a second computational group.

4.2   Current Methods

Reiterate the scarcity of extant Germanium studies. Probably definitely reference Dr. Weinert's work.

4.2.1   Precursory Work

Dr. Weinert's lab had received curious results from a prior collaborator and sought confirmation by a second group.

From the information provided in 4.1, the energy from Density Functional Theory calculations suggest that the most energetically-favorable conformation of hexagermane be the trans-cis-trans form.

4.3   Modeling Germanium molecules in Gaussian

This is the meat of the chapter - the Gaussian problem.

4.3.1 Recognition of Problem

Include Dr. Fennell's image. Relate to strangeness of collaborator's data. FRAME PROBLEM WITH THEORY AND BASIS SET

4.3.2 Confirmation of Germanium consistency from other systems

A minor note of confirmation of Gaussian problem from consistent results from other programs with same theory and basis set.

4.3.3 Correcting the Issue

Use basis set from Basis Set Exchange, detail selection and usage in input settings.

CHAPTER 5

Crystal and Liquid 2D Water at Interfaces

5.1 Two-Dimensional Water

EACH SUBSECTION: DEFINITION OF TERMS

Rose water is fairly new on the computational scene and so I may also include a review on the Mercedez-Benz water system as well as any other attempts to model water in two dimensions. For the rose potential system, I will review the Lennard-Jones potential as well as any other equations/systems related to the rose potential.

5.1.1 Lennard-Jones Potential

The Lennard-Jones Potential well is a soft-sphere model of interaction between two spheres described with

$$V_{LJ} = 4\epsilon[(\frac{\sigma}{r})^{12} - (\frac{\sigma}{r})^6]$$

(5.1)

where V is the potential, r is the distance between the center of two particles , σ is the specific distance between the two particles where the potential is zero, and $-\epsilon$ is the minimum potential of the plot. **REFINE WORDING:** The plot is defined in $[0, \infty)$. As two particles approach from infinity, their interaction become negative - which is an attractive force - and will approach the global minimum of $-\epsilon$. The r of this interaction is slightly larger than the combined radii of the two particles - which means they aren't quite touching - and is the equilibrium distance between the two particles. As r decreases beyond the minimum and toward σ, the interaction strength

increases and reaches zero as $r = \sigma$. At $r < \sigma$,

Potential digression: In a "hard-sphere" model, a particle's radius is firm, which is to say that the interaction potential is infinite at r less than σ. Basically a vertical line between two discrete values (usually ϵ and ∞)as the potential shifts from $r \geq \sigma$ to $r < \sigma$ (maybe include image?). The Lennard-Jones potential is a "soft-sphere" model, which blurs the line and replaces the vertical line with a functional representation. This breaks with reality as the particles become "squishy" and the potential ramps up toward infinity as r decreases. The benefit to the soft-sphere model is that modeling programs can more-easily account for overlaps in particles during time steps than with hard-sphere models. For example, a hard-sphere model of two particles interacting will likely not have a position where $r = \sigma$ and will potentially overlap. At this overlap, the potential is infinity and will introduce a nearly infinite force at that instant of time. Computer systems do not like having infinitely large repulsions suddenly introduced into a simulation.

5.1.2 Modeling Water in Two Dimensions

Modeling in two dimensions sacrifice the "realism" of models in three dimensions, but reduce the computational load significantly. This allows researchers (scientists, chemists, digital magicians?) to test more simple designs in two dimensions as well as a higher volume of simulations at the same time/computational cost.

5.1.2.1 Mercedes-Benz Model

The "Mercedes-Benz" BN2D model of water first proposed by Ben-Naim (1971) as "waterlike particles" are a popular two-dimensional representation of water. ROUGH: details of shape of MB water

The mathematical model used in the BN2D model is generated from the Percus-Yevick equation by substituting the approximation

$$c(X_1, X_2) = y(X_1, X_2)f(X_1, X_2) \tag{5.2}$$

into the Percus-Yevick equation obtained from the Ornstein-Zernike relation

$$h(X_1, X_2) = c(X_1, X_2) + \frac{\rho}{2\pi} \int c(X_1, X_3)h(X_3, X_2)dX_3 \tag{5.3}$$

to produce the overall relation

$$y(X_1, X_2) = 1 + \frac{\rho}{2\pi} \int y(X_1, X_3)f(X_1, X_3) \times \Big[y(X_3, X_2)f(X_3, X_2) + y(X_3, X_2) - 1\Big]dX_3 \tag{5.4}$$

5.1.2.2   Rose Potential Model

The rose potential is another model first introduced by Williamson et al. (2017). This model, while similar to the three-pronged BN2D, is notably different in that the rose potential model simplifies the model by use of a radial sinusoidal plot to make the three "prongs" of the particle.

5.1.2.3   Two-Dimensional Modeling

Something other than OOPSE? (not seeing obvious answer other than "custom code modified/forked from existing 3D tools")

The Object Oriented Parallel Simulation Engine (OOPSE) was introduced by A. et al. (n.d.) as a relatively light-weight molecular dynamics simulation package focused on "efficiently integrating equations of motion for atom types with orientational degrees of freedom" (from abstract). While OOPSE was further developed and renamed OpenMD, a fork of OOPSE was developed specifically to model water in two dimensions.

5.1.3 Literature Review on Relevant Works

intro: Paragraph refreshing current work. Emphasize computational efficiency of Rose-Potential over Mercedez-Benz model.

5.2 Goal of Project

The objective of this work was to model two-dimensional water with a surface designed to discourage crystal growth at freezing temperatures. The design of the surface was the primary focus. Successfully designing a surface capable of discouraging water ice formation at freezing temperatures would provide valuable information in designing a three-dimensional model of the same type at a reduced computational cost.

Idea: adjust freezing point depression to be freezing point modification.

5.3 Tools and Terms

Either a refresh from intro or a detailed explanation of OOPSE and the reduced terms.

5.3.1 OOPSE in 2D

Detail differences in computation system that Dr. Fennell developed to allow 2D MD(?) on a 3D program.

5.3.2 Reduced Terms: 2D analogues

Detail differences in dimensionality and define the reduced dimensions. Still working on the understanding/equations.

5.4 Designing System

Include: ensemble, thermodynamic variables, box attributes (size, pressure, temp, etc), number of waters, surface (size, spacing between beads of surface, charges, LJ values, etc)

5.4.1 Defining the Surface

Explain how to develop the surface in the program and how to build a custom surface (walk through surface builder here)

5.4.1.1 Manipulation of LJ Potential

Manipulate σ and ϵ values to effectively adjust the radius and interaction strength of surface beads.

5.4.1.2 Manipulation of bead spacing

Detail design of optimizing bead spacing for freezing encouragement or disruption.

5.5 Results

Success of freezing point elevation, pending results for freezing point depression

5.5.1 Future Work

Continued efforts to disrupt crystal growth on the surface.

CHAPTER 6

Conclusion

6.1   Ice I$_h$ Generation

Review:

6.1.1   Purpose

6.1.2   Results

6.1.3   Flaws/Improvements

6.1.4   Practical Implications

Easier to generate crystals for modeling research

6.2   Conformation Landscapes

Review:

6.2.1   Purpose

6.2.2   Results

6.2.3   Flaws/Improvements

6.2.4   Practical Implications

Collected Program for general use

6.3 Modeling Germanium Compounds

Review:

6.3.1 Purpose

6.3.2 Results

6.3.3 Flaws/Improvements

6.3.4 Practical Implications

Discovered mistake to be fixed

6.4 Two-Dimensional Water

Review:

6.4.1 Purpose

6.4.2 Results

6.4.3 Flaws/Improvements

6.4.4 Practical Implications

super-duper anti-freeze? also freezing point elevation.

6.5 Super Duper Final Conclusion

References

A., M. M., F., V. C., Teng, L., J., F. C. and Daniel, G. J. (n.d.), 'Oopse: An object-oriented parallel simulation engine for molecular dynamics', *Journal of Computational Chemistry* **26**(3), 252–271.
    URL: *https://onlinelibrary.wiley.com/doi/abs/10.1002/jcc.20161*

Ben-Naim, A. (1971), 'Statistical mechanics of "waterlike" particles in two dimensions. i. physical model and application of the percus-yevick equation', *The Journal of Chemical Physics* **54**(9), 3682–3695.
    URL: *https://doi.org/10.1063/1.1675414*

Bernal, J. D. and Fowler, R. H. (1933), 'A theory of water and ionic solution, with particular reference to hydrogen and hydroxyl ions', *The Journal of Chemical Physics* **1**(8), 515–548.
    URL: *https://doi.org/10.1063/1.1749327*

Pauling, L. (1935), 'The structure and entropy of ice and of other crystals with some randomness of atomic arrangement', *Journal of the American Chemical Society* **57**(12), 2680–2684.
    URL: *https://doi.org/10.1021/ja01315a102*

Williamson, C. H., Hall, J. R. and Fennell, C. J. (2017), 'Two-dimensional molecular simulations using rose potentials', *Journal of Molecular Liquids* **228**(Complete), 11–18.

APPENDIX A

Ice Ih to Ice XI Conversion

Listed below is the source code utilized in the conversion of a PDB Ice Ih structure into an Ice XI structure.

A.1   Code: PDBDisorganize.py

```python
#!/usr/bin/python

# Author = Gentry Smith
# Copyright 2016, all rights reserved

# this reads in a .PDB file, takes an argument for deformities per
#     molecules, and randomly organizes the crystal
# structure into a disordered proton formation

# import sample: python PDBDisorganize.py arg1 arg2 arg3
# where:
# arg1 = source pdb file to be read (ex: acetone.pdb or acetone)
# arg2 = number of defects per molecule (in H20, num of non-hydrogen-
#     bonds. from 0 to 4)
# arg3 = desired output pdb file name

import sys
print sys.path
import string
import numpy as np
import math
import random

sys.setrecursionlimit(10000000) # maximum recursive depth. Set to
#     (10,000,000) as under maximum


pdbIN = file(sys.argv[1])        # source PDB file
maxErr = int(sys.argv[2])      # max errors allowed
pdbOUT = str(sys.argv[3])      # output file name
finalData = [ [ [ 0 for i in range(3) ] for j in range(3) ] for k in
     range(300) ]

# looks at args validity
def checkArgs(arg1, arg2, arg3):
    returnBool = False
    if type(arg1) != file: # check arg1
```

```python
            print"Bad arg", arg1, " must be a file "
            returnBool = True
        if type(arg3) != str: # check arg3
            print"Bad arg", arg3, ", must be a file name"
            checkPDBSuffix(arg3)
            print arg3
            returnBool = True
        if type(arg2) != int:  # check arg2 type
            print "Bad arg2: ", arg2, " is not an int."
            returnBool = True
        elif type(arg2) == int:
            if arg2 < 0 or arg2 > 4:  # check arg2 range
                print "arg2 is not in a valid range 0 <= arg2 <= 4"
                returnBool = True
        return returnBool

def checkPDBSuffix(pdbFile):
    if string.find(pdbFile, '.pdb', 0, len(pdbFile)) == -1:
        print("did not find 'pdb' in ", pdbFile, ". Appending...")
        pdbFile += '.pdb'



# reads in file,
def readFile(fileName):
    print "Reading file..."
    # gets number of atoms
    atoms = 0
    for line in fileName:
        data = line.split()
        if len(data) > 0:
            if data[0] != "CONECT" and data[0] != "END":
                atoms += 1
    # print "atoms: ", atoms
    numMol = atoms / 3  # assumes 3-atom water molecule
    dataTable = [ [ [ 0 for i in range(3) ] for j in range(3) ] for k in
     range(numMol) ]
    fileName.seek(0)
    iter0 = 0
    iter1 = 0
    pdbType = -1
    for line in fileName:
        data = line.split()
        if pdbType == -1:
            if data[0] == "ATOM":
                pdbType = 0
            elif data[0] == "HETATM":
                pdbType = 1
        # print "LineTuple= ", data
        if len(data) > 1 and ( data[0] == "ATOM" or data[0] == "HETATM"
    ):
            if data[0] == "ATOM":
                newData = getDataATOM(data)
                for i in range(3):
```

```
87                          #data[molecule][atom][X/Y/Z]
88                          dataTable[iter0][iter1 % 3][i] = newData[i]
89                   elif data[0] == "HETATM":
90                       dataTable[iter0][iter1 % 3] = getDataHETATM(data)
91                   if iter1 == 2:
92                       iter0 += 1
93                       iter1 = 0
94                   elif iter1 != 2:
95                       iter1 += 1
96      # print "DataTable: ", dataTable
97       print "File read"
98       return dataTable, pdbType
99

100
101      # Split by index
102      # if having a problem with reading data, check .pdb to see if data
     has a space between each value
103
104 # reads XYZ coordinate data from ATOM-type pdb
105 def getDataATOM(strLine):
106      # print "Getting ATOM Data..."
107      dataLine = strLine[5:8]
108      # print "dataline: ", dataLine
109      i = 0
110      while i < 3:
111          # print "dataline[", i, "]: ", dataLine[i]
112          dataLine[i] = float(dataLine[i])
113          # print "dataline[", i, "] type: ", type(dataLine[i])
114          i += 1
115      return dataLine
116

117
118 # reads XYZ coordinate data from HETATM-type pdb
119 def getDataHETATM(strLine):
120      # print "Getting HETATM Data..."
121      dataLine = strLine[5:8]
122      # print "dataline: ", dataLine
123      i = 0
124      while i < 3:
125          # print "dataline[", i, "]: ", dataLine[i]
126          dataLine[i] = float(dataLine[i])
127          # print "dataline[", i, "] type: ", type(dataLine[i])
128          i += 1
129      return dataLine
130

131
132 # gets all four position vectors of hydrogen/lone pair as offset of
     oxygen molecule
133 def getOrientations( molecule ):
134      # 120 degrees = ( 2 * pi ) / 3 radians
135      theta = ( ( 2 * math.pi ) / 3 )
136      newMol = zeroOrientation(molecule)
137      returnInt1 = rotateMolecule(newMol[1], newMol[2], theta)
138      returnInt2 = rotateMolecule(newMol[1], newMol[2], (-1 * theta) )
```

```python
139        return [returnInt1, returnInt2]
140
141
142 # randomly selects new orientation, returns two unique ints, from 0 to 3
         inclusively
143 def newRandOrientation( positions ):
144     # print "Changing orientation"
145     randVal1 = random.randint(0,3)
146     randVal2 = random.randint(0,3)
147     while randVal1 == randVal2:
148         randVal2 = random.randint(0,3)
149     newMol = [ [ 0, 0, 0 ],
150                 positions[ randVal1 ] ,
151                 positions[ randVal2 ]   ]
152     return newMol
153
154 # selects new orientation from list. Reduces computational overhead in
       re-orientation option traversal
155 def newSetOrientation( positions, pos1, pos2 ):
156     newMol = [ [ 0, 0, 0 ],
157                 positions[ pos1 ],
158                 positions[ pos2 ] ]
159     return newMol
160
161
162 # sets molecule coordinates so that oxygen is the origin
163 def zeroOrientation(source):
164     # print "Zeroing Molecule..."
165
166     oxy = source[0]
167     hyd1 = source[1]
168     hyd2 = source[2]
169
170     # print "Oxygen pos: ", oxy
171     # print "Hydrogen 1: ", hyd1
172     # print "Hydrogen 2: ", hyd2
173
174     zeroedOrigin = [0, 0, 0]
175     zeroedHyd1 = [0, 0, 0]
176     zeroedHyd2 = [0, 0, 0]
177     for i in range(3):
178         zeroedHyd1[i] = hyd1[i] - oxy[i]
179         zeroedHyd2[i] = hyd2[i] - oxy[i]
180
181     # print "Zeroed Hydrogen 1: ", zeroedHyd1
182     # print "Zeroed Hydrogen 2: ", zeroedHyd2
183
184     # return new molecule position
185     newMol = [zeroedOrigin, zeroedHyd1, zeroedHyd2]
186     return newMol
187
188 # resets the zeroed molecule to the original oxygen position
189 def resetOrientation(oxygenPos, molecule):
190     # print "Resetting molecule..."
```

```
191      rO = oxygenPos
192      rH1 = [0,0,0]
193      rH2 = [0,0,0]
194      newMol = []
195      for i in range(3):
196          rH1[i] = molecule[1][i] + rO[i]
197          rH2[i] = molecule[2][i] + rO[i]
198          newMol = [rO, rH1, rH2]
199      # print "Rebuilt Molecule: ", newMol
200      return newMol
201
202 # rotates vector about axis for theta degrees
203 # Handler for rotationMatrix function below
204 def rotateMolecule(vector, axis, theta):
205      rotMatx = rotationMatrix(axis, theta)
206      return np.dot(rotMatx, vector)
207
208
209 # Creates Rotation matrix for a given axis and theta
210 # from stackoverflow user unutbu
211 # page: http://stackoverflow.com/questions/6802577/python-rotation-of-3d
         -vector
212 def rotationMatrix(axis, theta):
213      """
214
215      :type axis: list
216      :type theta: union
217      """
218      axis = np.asarray(axis)
219      theta = np.asarray(theta)
220      axis /= math.sqrt(np.dot(axis, axis))
221      a = math.cos( theta/2.0 )
222      b, c, d = -axis*math.sin(theta/2.0)
223      aa, bb, cc, dd = (a * a), (b * b), (c * c), (d * d)
224      bc, ad, ac, ab, bd, cd = (b * c), (a * d), (a * c), (a * b), (b * d)
         , (c * d)
225      return np.array( [ [ (aa + bb - cc - dd), ( 2 * ( bc + ad ) ), ( 2 *
         ( bd - ac ) ) ],
226                              [ ( 2 * ( bc - ad ) ), (aa + cc - bb - dd), ( 2 *
         ( cd + ab ) ) ],
227                              [ ( 2 * ( bd + ac ) ), ( 2 * ( cd - ab ) ), (aa +
         dd - bb - cc) ] ] )
228
229
230 # gets results from rotateAboutAxis plus two Hydrogens to get the
         tetrahedron positions
231 def getTetrahedronPositions(molecule):
232      positions = [ [ 0 for i in range(3) ] for j in range(4) ]
233      newMol = zeroOrientation(molecule)  # zero molecule
234      positions[0] = newMol[1]
235      positions[1] = newMol[2]
236      newPos = getOrientations(molecule)  # get final two positions
237      positions[2] = list(newPos[0])
238      positions[3] = list(newPos[1])
```

```python
239     return positions                              # return all four positions
240
241
242 # checks distance of new positions from zero
243 def checkDist(posArray):
244     distance = [0 for i in range(len(posArray))]
245     for i in range(len(posArray)):
246         distance[i] = (  (posArray[i][0] * posArray[i][0]) +
247                          (posArray[i][1] * posArray[i][1]) +
248                          (posArray[i][2] * posArray[i][2])  )
249         # print "Distance", i, ": ", distance[i]
250     avg = 0
251     for i in range(len(posArray)):
252         avg += distance[i]
253     averageDistance = ( avg / len(posArray) )
254     # print "Average Distance: ", averageDistance
255     return averageDistance
256
257
258 # prints data given a 3D table of water molecules
259 def printData(data):
260     print "Data: "
261     strData = [" O", "H1", "H2"]
262     dimData = ["X", "Y", "Z"]
263     bigAvg = 0
264     numAtoms = 0
265     for mol in range(len(data)):
266         for atom in range(len(data[mol])):
267             printStr = str(mol) + ": " + strData[atom] + ": "
268             for dimension in range(3):
269                 printStr += dimData[atom] + ":" + "{:7.3f}".format(data[
     mol][atom][dimension]) + "\t"
270             print printStr
271         bigAvg += checkDist(zeroOrientation(data[mol])[1:])
272         numAtoms += 1
273         print ""
274     print "total average distance: ", bigAvg / numAtoms
275
276
277 # checks validity of molecule
278 def isDefectiveCheck(err, neighborData, posData, index):
279     # find nearby molecules (avg oxygen distance???)
280     print "checking for defects at index", index, "..."
281     print "neighbor indices: ", neighborData[index]
282     returnBool = False
283     neighbors = 4
284     for i in range(4):  # count real neighbors
285         if neighborData[index][1][i] == -1:
286             neighbors -= 1
287     if neighbors <= err:      # de facto good if num(neighbors) <
     maxErrAllowed
288         # print "Fewer neighbors than allowed errors. de facto Good
     Orientation"
289         returnBool = True
```

39

```python
290        elif neighbors > err:    # enough neighbors to require check
291            # print "More neighbors than error threshold"
292            defectCount = 0
293            for neighbor in range(4):    # check each neighbor
294                    if neighborData[index][1][neighbor] != -1: # skip over non-
    existant neighbors
295                        molA = posData[index]
296                        molB = posData[ neighborData[index][1][neighbor] ]
297                        oxyDist = getDistBetweenAtoms(molA[0], molB[0])
298
299                        if minHydrogenDistance(molA, molB) > oxyDist: # check
    for facing lone pairs
300                            print "Double Lone Pair defect"
301                            defectCount += 1
302                            break
303                    else:    # check for facing protons
304                        smallerHydrogenDistanceCount = 0
305                        isDefective = False
306                        for first in range(2):
307                            if not isDefective:
308                                for second in range(2):
309                                    newDist = getDistBetweenAtoms(molA[first
    + 1], molB[second + 1])
310                                    if newDist < oxyDist:
311                                        smallerHydrogenDistanceCount += 1
312                                if smallerHydrogenDistanceCount > 1:
313                                    print "Double Hydrogen defect"
314                                    defectCount += 1
315                                    isDefective = True
316        # print "Defects found:", defectCount
317        if defectCount > 4:
318            print "IMPOSSIBLE AMOUNT OF DEFECTS DETECTED!!!!
    AHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHH"
319        if defectCount > err:
320            # print "Found a bad molecule!"
321            returnBool = False
322        else:
323            # print "Molecule is within parameters."
324            returnBool = True
325
326    return returnBool
327
328
329 # randomly re-reorients molecule and neighbors, rechecks all
330 def rerunMolAndNeighbors(err, neighborData, posData, index):
331    # print "Re-reordering molecule at", index
332    # err - max errors allowed
333    # neighborData - int[4] of neighbor indices
334    # posData - array of all molecule position vectors
335    # index - location of focus molecule in posData
336    isGood = False
337    timeCount = 0
338    while not isGood:
```

```python
339             # re-rotate molecule through all positions (iterated through all
          orientations)
340             positions = getTetrahedronPositions(posData[index])
341             zeroedMol = newRandOrientation(positions)
342             # print "isGood CHECK", isGood
343             isGood, posData = iterThroughRotations(err, neighborData,
        posData, index)
344             posData[index] = resetOrientation(posData[index][0], zeroedMol)
345             if timeCount >= 13: # { (1 - 1/6)^n < 0.05 } says n = 17
346                 # BROKEN - need to rebuild
347                 # 0. evaluated molecule has too many defects
348                 # 1. reorient molecule statistically probable amount of
        times to cover all orientations
349                 # 2. Repeat 1. with neighbor 1
350                 # 2a repeat 1. with original molecule
351                 # 3. Repeat 2. with neighbor 2, 3, 4, as/if necessary
352                 for neighborIndex in range(4):
353                     if neighborData[index][1][neighborIndex] != -1:
354                         positions = getTetrahedronPositions(posData[
        neighborIndex])
355                         zeroedMol = newRandOrientation(positions)
356                         posData[neighborIndex] = resetOrientation(posData[
        neighborIndex][0], zeroedMol)
357                         # isGood = isDefectiveCheck(err, neighborData,
        posData, neighborIndex)
358                 isGood = isDefectiveCheck(err, neighborData, posData, index)
359                 if not isGood:
360                     isGood, posData = rerunMolAndNeighbors(err, neighborData
        , posData, neighborData[index][1][neighborIndex])
361     finalData = posData
362     return True, finalData
363
364 # iterates molecule through all possible rotations
365 def iterThroughRotations(err, neighborData, posData, index):
366     isGood = False
367     pos1 = 0  # tetrahedral position for H1
368     pos2 = 0  # tetrahedral position for H2
369     while not isGood or (pos1 != 3 and pos2 != 3):  # iterates through
        all orientations, stops if good orientation
370         if pos1 != pos2:
371             posData[index] = newSetOrientation(posData[index][0], pos1,
        pos2)
372             isGood = isDefectiveCheck(err, neighborData, posData, index)
373         if pos2 < 3:
374             pos2 += 1
375         elif pos2 == 3:
376             if pos1 < 3:
377                 pos1 += 1
378                 pos2 = 0
379     return isGood, posData
380 # determines minimum hydrogen distance between two atoms
381 def minHydrogenDistance(mol1, mol2):
382     minDist = 100
383     for first in range(2):
```

```python
384                for second in range(2):
385                    newDist = getDistBetweenAtoms(mol1[first+1], mol2[second+1])
386                    if newDist < minDist:
387                        minDist = newDist
388        return minDist
389
390
391
392
393
394 # finds neighboring molecules of each molecule
395 def getNeighbors(data):
396     returnData = [ [ [ 0 for i in range(4) ] for j in range(2) ] for k
       in range(len(data)) ] # data[molecule][distance,index][four values]
397     for mol1 in range(len(data)):
398         minDist = [100, 100, 100, 100]
399         minIndex = [0, 0, 0, 0]
400         for mol2 in range(len(data)):
401             if mol1 != mol2:
402                 newMin = getDistBetweenAtoms(data[mol1][0], data[mol2
       ][0])
403
404                 bigIndex = indexOfBiggest(minDist)
405                 if newMin < minDist[bigIndex]:
406                     minDist[bigIndex] = newMin
407                     minIndex[bigIndex] = mol2
408         for i in range(4):
409             if minDist[i] >= 9:
410                 minDist[i] = -1
411                 minIndex[i] = -1
412         # print "Four smallest Distances of", mol1, ": ", minDist
413         # print "Four smallest Indices of", mol1, ": ", minIndex
414         returnData[mol1] = [minDist, minIndex]
415     return returnData
416
417
418 # finds distance between oxygen atoms
419 def getDistBetweenAtoms( mol1, mol2 ):
420     distance = ( ( ( mol1[0] - mol2[0] ) * ( mol1[0] - mol2[0] ) ) +
421                  ( ( mol1[1] - mol2[1] ) * ( mol1[1] - mol2[1] ) ) +
422                  ( ( mol1[2] - mol2[2] ) * ( mol1[2] - mol2[2] ) ) )
423     return distance
424
425 # gets index of largest item from a list
426 def indexOfBiggest(check):
427     bigIndex = 0
428     for i in range(len(check)):
429         if check[i] > check[bigIndex]:
430             bigIndex = i
431     return bigIndex
432
433
434 # writes data to PDB file
435 def writeDataPDB(data, pdbType):
```

```python
436        print "Writing Data to", str(pdbOUT)
437        fileName = str(pdbOUT)
438        output = open(fileName, 'w')
439        if pdbType == 0:
440            writeDataPDBATOM(data, output)
441        elif pdbType == 1:
442            writeDataPDBHETATM(data, output)
443        output.close()
444
445
446 # Writes data to PDB file style = ATOM
447 def writeDataPDBATOM(data, inFile):
448        iterator = 0
449        for molecule in range(len(data)):
450            for atom in range(3):
451                iterator += 1
452                outStr = "ATOM  "
453                outStr += str(iterator)
454                while len(outStr) < 11:
455                    outStr = outStr[:6] + " " + outStr[6:]
456                outStr += " "
457                if atom == 0:
458                    outStr += " O  " + " WAT"
459                elif atom == 1:
460                    outStr += " H1 " + " WAT"
461                elif atom == 2:
462                    outStr += " H2 " + " WAT"
463                outStr += str(molecule)
464                while len(outStr) < 26:
465                    outStr = outStr[:20] + " " + outStr[20:]
466                outStr += "    "
467                outStr += "{:8.3f}".format(data[molecule][atom][0])
468                outStr += "{:8.3f}".format(data[molecule][atom][1])
469                outStr += "{:8.3f}".format(data[molecule][atom][2])
470                outStr += "  1.00" + "  0.00"
471                outStr += "          "
472                if atom == 0:
473                    outStr += " O  "
474                elif atom == 1:
475                    outStr += " H  "
476                elif atom == 2:
477                    outStr += " H  "
478                outStr += "\n"
479                inFile.write(outStr)
480
481
482 # Writes data to PDB file style = HETATOM
483 def writeDataPDBHETATM(data, inFile):
484        iterator = 0
485        for molecule in range(len(data)):
486            for atom in range(3):
487                iterator += 1
488                outStr = "HETATM"
489                outStr += str(iterator)
```

```python
                while len(outStr) < 11:
                    outStr = outStr[:6] + " " + outStr[6:]
                outStr += " "
                if atom == 0:
                    outStr += " O  " + " WAT"
                elif atom == 1:
                    outStr += " H1 " + " WAT"
                elif atom == 2:
                    outStr += " H2 " + " WAT"
                outStr += str(molecule)
                while len(outStr) < 26:
                    outStr = outStr[:20] + " " + outStr[20:]
                outStr += "      "
                outStr += "{:8.3f}".format(data[molecule][atom][0])
                outStr += "{:8.3f}".format(data[molecule][atom][1])
                outStr += "{:8.3f}".format(data[molecule][atom][2])
                outStr += "  1.00" + "  0.00"
                outStr += "              "
                if atom == 0:
                    outStr += " O  "
                elif atom == 1:
                    outStr += " H  "
                elif atom == 2:
                    outStr += " H  "
                outStr += "\n"
                inFile.write(outStr)


# runs program
def testRun(inFile, err, outFile):
    print "Running Test Version of Program..."


# this is the parent runner for the program
def runPgm(inFile, err):
    print "Running Program..."
    data, pdbType = readFile(inFile)
    newData = [ [ [ 0 for i in range(3) ] for j in range(3) ] for k in
        range(len(data)) ]
    print "Reordering Molecules..."
    for i in range(len(data)):
        positions = getTetrahedronPositions(data[i])
        zeroedMol = newRandOrientation(positions)
        newMol = resetOrientation( data[i][0], zeroedMol )
        newData[i] = newMol
    print "Molecules Reordered"
    connectedMolecules = getNeighbors(newData)   # -1 index = not
        neighboring
    finalData = newData
    for i in range(len(connectedMolecules)):
        # print "check defects"
        isFine = isDefectiveCheck(err, connectedMolecules, finalData, i)
        # print "isFINE CHECK", isFine
        if not isFine:
```

```python
542                # print "fixing defects"
543                 while not isFine:
544                     # print "RerunMol"
545                     isFine, finalData = rerunMolAndNeighbors(err,
        connectedMolecules, finalData, i)
546                     # print "rerunDone"
547        writeDataPDB(finalData, pdbType)
548        # printData(newData)
549
550
551 badArgs = checkArgs(pdbIN, maxErr, pdbOUT) # stop in case of bad
        argument
552
553 # check input args
554 if not badArgs: # stop in case of bad argument
555        print "Good Arguments, Initializing Reorientiation with", maxErr, "
        maximum defects"
556        # testRun(pdbIN, maxErr, pdbOUT)
557        runPgm(pdbIN, maxErr)
558 elif badArgs:
559        print "Bad Arguments, Quitting..."
```

APPENDIX B

Conformation Landscapes

APPENDIX C

Germanium Landscape

Listed below are two example Germanium PDB files. The first is for the end-goal hexagermane in the trans-trans-trans conformation with isopropyl groups on the terminal Ge atoms. The second is for the simplified butagermane with fully protonated Germanium atoms.

C.1   Code: hexagermane-transall.pdb

```
1  HEADER
2  REMARK  Title: hexagermane_transall system
3  HETATM     1 Ge            1     -4.399    0.008    0.355   0.00   0.00
          Ge
4  HETATM     2 Ge            1     -1.965    0.138   -0.022   0.00   0.00
          Ge
5  HETATM     3  C            1     -4.822    1.886    0.961   0.00   0.00
           C
6  HETATM     4  C            1     -5.008   -1.297    1.715   0.00   0.00
           C
7  HETATM     5  C            1     -5.256   -0.261   -1.445   0.00   0.00
           C
8  HETATM     6  C            1     -1.213    1.435    1.157   0.00   0.00
           C
9  HETATM     7 Ge            1     -0.756   -1.988    0.223   0.00   0.00
          Ge
10 HETATM     8  C            1     -1.297   -2.917    1.805   0.00   0.00
           C
11 HETATM     9 Ge            1      1.647   -1.496    0.371   0.00   0.00
          Ge
12 HETATM    10  C            1     -1.182   -3.010   -1.339   0.00   0.00
           C
13 HETATM    11  C            1      2.131   -0.425    1.877   0.00   0.00
           C
14 HETATM    12  C            1      2.111   -0.634   -1.269   0.00   0.00
           C
15 HETATM    13 Ge            1      2.889   -3.585    0.738   0.00   0.00
          Ge
16 HETATM    14  C            1      2.287   -4.358    2.378   0.00   0.00
           C
17 HETATM    15 Ge            1      5.327   -3.386    1.080   0.00   0.00
          Ge
18 HETATM    16  C            1      2.766   -4.685   -0.813   0.00   0.00
           C
19 HETATM    17  C            1      5.688   -2.615    2.887   0.00   0.00
           C
```

47

20	HETATM	18	C	C	1	6.239	−2.415	−0.417	0.00	0.00
21	HETATM	19	C	C	1	5.893	−5.324	0.888	0.00	0.00
22	HETATM	20	C	C	1	−3.527	2.543	1.328	0.00	0.00
23	HETATM	21	C	C	1	−5.754	1.844	2.133	0.00	0.00
24	HETATM	22	H	H	1	−5.303	2.355	0.072	0.00	0.00
25	HETATM	23	H	H	1	−5.269	1.358	2.999	0.00	0.00
26	HETATM	24	H	H	1	−6.679	1.287	1.913	0.00	0.00
27	HETATM	25	H	H	1	−6.047	2.856	2.449	0.00	0.00
28	HETATM	26	H	H	1	−3.043	2.019	2.171	0.00	0.00
29	HETATM	27	H	H	1	−3.683	3.585	1.642	0.00	0.00
30	HETATM	28	H	H	1	−2.818	2.559	0.490	0.00	0.00
31	HETATM	29	H	H	1	−4.336	−1.167	2.589	0.00	0.00
32	HETATM	30	C	C	1	−4.907	−2.680	1.154	0.00	0.00
33	HETATM	31	C	C	1	−6.417	−0.909	2.051	0.00	0.00
34	HETATM	32	H	H	1	−3.858	−2.985	0.982	0.00	0.00
35	HETATM	33	H	H	1	−5.433	−2.773	0.187	0.00	0.00
36	HETATM	34	H	H	1	−5.349	−3.420	1.836	0.00	0.00
37	HETATM	35	H	H	1	−6.488	0.167	2.302	0.00	0.00
38	HETATM	36	H	H	1	−6.802	−1.477	2.909	0.00	0.00
39	HETATM	37	H	H	1	−7.103	−1.094	1.205	0.00	0.00
40	HETATM	38	C	C	1	5.200	−6.127	1.944	0.00	0.00
41	HETATM	39	C	C	1	7.384	−5.453	0.966	0.00	0.00
42	HETATM	40	H	H	1	5.523	−5.590	−0.126	0.00	0.00
43	HETATM	41	H	H	1	7.790	−4.974	1.874	0.00	0.00
44	HETATM	42	H	H	1	7.885	−4.994	0.099	0.00	0.00
45	HETATM	43	H	H	1	7.691	−6.509	0.992	0.00	0.00
46	HETATM	44	H	H	1	5.502	−5.821	2.960	0.00	0.00

47	HETATM	45	H	1	5.436	−7.197	1.849	0.00	0.00
			H						
48	HETATM	46	H	1	4.106	−6.027	1.879	0.00	0.00
			H						
49	HETATM	47	C	1	6.243	−1.232	2.746	0.00	0.00
			C						
50	HETATM	48	C	1	6.612	−3.524	3.636	0.00	0.00
			C						
51	HETATM	49	H	1	4.684	−2.582	3.376	0.00	0.00
			H						
52	HETATM	50	H	1	7.535	−3.731	3.068	0.00	0.00
			H						
53	HETATM	51	H	1	6.139	−4.497	3.853	0.00	0.00
			H						
54	HETATM	52	H	1	6.913	−3.088	4.599	0.00	0.00
			H						
55	HETATM	53	H	1	7.243	−1.234	2.279	0.00	0.00
			H						
56	HETATM	54	H	1	6.347	−0.742	3.725	0.00	0.00
			H						
57	HETATM	55	H	1	5.589	−0.589	2.128	0.00	0.00
			H						
58	HETATM	56	C	1	5.630	−1.055	−0.555	0.00	0.00
			C						
59	HETATM	57	H	1	6.024	−3.039	−1.315	0.00	0.00
			H						
60	HETATM	58	C	1	7.712	−2.342	−0.145	0.00	0.00
			C						
61	HETATM	59	H	1	7.923	−1.890	0.839	0.00	0.00
			H						
62	HETATM	60	H	1	8.227	−1.728	−0.898	0.00	0.00
			H						
63	HETATM	61	H	1	8.188	−3.335	−0.163	0.00	0.00
			H						
64	HETATM	62	H	1	4.573	−1.106	−0.861	0.00	0.00
			H						
65	HETATM	63	H	1	6.155	−0.455	−1.314	0.00	0.00
			H						
66	HETATM	64	H	1	5.675	−0.486	0.391	0.00	0.00
			H						
67	HETATM	65	H	1	−5.890	−1.163	−1.302	0.00	0.00
			H						
68	HETATM	66	C	1	−4.220	−0.487	−2.505	0.00	0.00
			C						
69	HETATM	67	C	1	−6.093	0.945	−1.729	0.00	0.00
			C						
70	HETATM	68	H	1	−6.841	1.122	−0.939	0.00	0.00
			H						
71	HETATM	69	H	1	−6.644	0.838	−2.676	0.00	0.00
			H						
72	HETATM	70	H	1	−5.478	1.858	−1.818	0.00	0.00
			H						
73	HETATM	71	H	1	−3.754	−1.481	−2.414	0.00	0.00
			H						

74	HETATM	72	H H	1	−3.411	0.262	−2.459	0.00	0.00
75	HETATM	73	H H	1	−4.659	−0.429	−3.512	0.00	0.00
76	HETATM	74	C C	1	−1.706	1.681	2.429	0.00	0.00
77	HETATM	75	C C	1	−0.128	2.155	0.679	0.00	0.00
78	HETATM	76	H H	1	0.268	1.941	−0.323	0.00	0.00
79	HETATM	77	C C	1	0.451	3.147	1.465	0.00	0.00
80	HETATM	78	C C	1	−1.134	2.678	3.216	0.00	0.00
81	HETATM	79	C C	1	−0.058	3.415	2.731	0.00	0.00
82	HETATM	80	H H	1	−1.525	2.873	4.219	0.00	0.00
83	HETATM	81	H H	1	1.306	3.716	1.086	0.00	0.00
84	HETATM	82	H H	1	0.391	4.199	3.349	0.00	0.00
85	HETATM	83	C C	1	−1.557	−4.274	1.694	0.00	0.00
86	HETATM	84	C C	1	−0.365	−3.088	−2.455	0.00	0.00
87	HETATM	85	C C	1	−2.027	−4.985	2.796	0.00	0.00
88	HETATM	86	H H	1	−1.378	−4.792	0.742	0.00	0.00
89	HETATM	87	C C	1	−1.446	−2.277	3.025	0.00	0.00
90	HETATM	88	C C	1	−0.752	−3.866	−3.544	0.00	0.00
91	HETATM	89	H H	1	0.592	−2.548	−2.482	0.00	0.00
92	HETATM	90	C C	1	−2.386	−3.699	−1.304	0.00	0.00
93	HETATM	91	C C	1	−2.219	−4.336	4.011	0.00	0.00
94	HETATM	92	H H	1	−2.237	−6.056	2.707	0.00	0.00
95	HETATM	93	C C	1	−1.915	−2.983	4.130	0.00	0.00
96	HETATM	94	H H	1	−1.159	−1.217	3.132	0.00	0.00
97	HETATM	95	C C	1	−2.771	−4.484	−2.388	0.00	0.00
98	HETATM	96	H H	1	−3.043	−3.610	−0.422	0.00	0.00
99	HETATM	97	C C	1	−1.952	−4.568	−3.509	0.00	0.00
100	HETATM	98	H H	1	−0.105	−3.928	−4.425	0.00	0.00

101	HETATM	99	H	H	1	−3.721	−5.027	−2.358	0.00	0.00
102	HETATM	100	H	H	1	−2.253	−5.182	−4.364	0.00	0.00
103	HETATM	101	H	H	1	−2.596	−4.891	4.876	0.00	0.00
104	HETATM	102	H	H	1	−2.041	−2.474	5.091	0.00	0.00
105	HETATM	103	C	C	1	2.487	−3.679	3.571	0.00	0.00
106	HETATM	104	C	C	1	1.701	−5.563	−0.935	0.00	0.00
107	HETATM	105	C	C	1	3.733	−4.618	−1.807	0.00	0.00
108	HETATM	106	H	H	1	0.940	−5.615	−0.140	0.00	0.00
109	HETATM	107	C	C	1	1.598	−6.382	−2.057	0.00	0.00
110	HETATM	108	C	C	1	1.690	−5.609	2.382	0.00	0.00
111	HETATM	109	C	C	1	2.102	−4.259	4.776	0.00	0.00
112	HETATM	110	H	H	1	2.956	−2.680	3.567	0.00	0.00
113	HETATM	111	C	C	1	1.520	−5.523	4.784	0.00	0.00
114	HETATM	112	H	H	1	2.260	−3.721	5.716	0.00	0.00
115	HETATM	113	C	C	1	1.311	−6.197	3.585	0.00	0.00
116	HETATM	114	H	H	1	1.504	−6.131	1.431	0.00	0.00
117	HETATM	115	C	C	1	2.562	−6.313	−3.057	0.00	0.00
118	HETATM	116	H	H	1	0.754	−7.074	−2.153	0.00	0.00
119	HETATM	117	C	C	1	3.630	−5.430	−2.933	0.00	0.00
120	HETATM	118	H	H	1	4.590	−3.931	−1.700	0.00	0.00
121	HETATM	119	H	H	1	4.391	−5.376	−3.718	0.00	0.00
122	HETATM	120	H	H	1	2.481	−6.954	−3.941	0.00	0.00
123	HETATM	121	H	H	1	1.223	−5.984	5.731	0.00	0.00
124	HETATM	122	H	H	1	0.844	−7.187	3.587	0.00	0.00
125	HETATM	123	C	C	1	1.878	0.732	−1.306	0.00	0.00
126	HETATM	124	C	C	1	1.530	−0.534	3.120	0.00	0.00
127	HETATM	125	C	C	1	2.642	−1.289	−2.370	0.00	0.00

128	HETATM	126	C	1	2.179	1.455	−2.458	0.00	0.00
			C						
129	HETATM	127	H	1	1.444	1.239	−0.432	0.00	0.00
			H						
130	HETATM	128	C	1	3.179	0.461	1.679	0.00	0.00
			C						
131	HETATM	129	C	1	2.005	0.227	4.186	0.00	0.00
			C						
132	HETATM	130	H	1	0.661	−1.197	3.265	0.00	0.00
			H						
133	HETATM	131	C	1	2.940	−0.568	−3.524	0.00	0.00
			C						
134	HETATM	132	H	1	2.840	−2.370	−2.334	0.00	0.00
			H						
135	HETATM	133	C	1	2.710	0.804	−3.567	0.00	0.00
			C						
136	HETATM	134	H	1	1.989	2.533	−2.491	0.00	0.00
			H						
137	HETATM	135	H	1	3.358	−1.081	−4.396	0.00	0.00
			H						
138	HETATM	136	H	1	2.944	1.370	−4.475	0.00	0.00
			H						
139	HETATM	137	C	1	3.067	1.105	3.998	0.00	0.00
			C						
140	HETATM	138	H	1	1.534	0.140	5.170	0.00	0.00
			H						
141	HETATM	139	C	1	3.650	1.229	2.740	0.00	0.00
			C						
142	HETATM	140	H	1	3.633	0.553	0.682	0.00	0.00
			H						
143	HETATM	141	H	1	4.480	1.926	2.585	0.00	0.00
			H						
144	HETATM	142	H	1	3.439	1.703	4.836	0.00	0.00
			H						
145	HETATM	143	C	1	−2.039	0.838	−1.804	0.00	0.00
			C						
146	HETATM	144	C	1	−1.525	0.195	−2.916	0.00	0.00
			C						
147	HETATM	145	C	1	−2.655	2.077	−1.927	0.00	0.00
			C						
148	HETATM	146	C	1	−1.618	0.802	−4.168	0.00	0.00
			C						
149	HETATM	147	H	1	−1.048	−0.789	−2.818	0.00	0.00
			H						
150	HETATM	148	C	1	−2.746	2.686	−3.175	0.00	0.00
			C						
151	HETATM	149	H	1	−3.084	2.566	−1.036	0.00	0.00
			H						
152	HETATM	150	C	1	−2.223	2.047	−4.296	0.00	0.00
			C						
153	HETATM	151	H	1	−1.210	0.296	−5.049	0.00	0.00
			H						
154	HETATM	152	H	1	−3.229	3.663	−3.275	0.00	0.00
			H						

155	HETATM	153	H H		1	−2.292	2.524	−5.279	0.00	0.00
156	HETATM	154	H H		1	−2.539	1.081	2.827	0.00	0.00
157	CONECT	3	1	20	21	22				
158	CONECT	4	1	29	30	31				
159	CONECT	5	1	65	66	67				
160	CONECT	6	74	75	2					
161	CONECT	8	83	87	7					
162	CONECT	10	84	90	7					
163	CONECT	11	124	128	9					
164	CONECT	12	123	125	9					
165	CONECT	14	103	108	13					
166	CONECT	16	104	105	13					
167	CONECT	17	15	47	48	49				
168	CONECT	18	56	57	58	15				
169	CONECT	19	15	38	39	40				
170	CONECT	20	3	26	27	28				
171	CONECT	21	3	23	24	25				
172	CONECT	30	4	34	32	33				
173	CONECT	31	4	35	36	37				
174	CONECT	38	19	44	45	46				
175	CONECT	39	19	41	42	43				
176	CONECT	47	54	55	17	53				
177	CONECT	48	17	50	51	52				
178	CONECT	56	62	63	64	18				
179	CONECT	58	59	60	61	18				
180	CONECT	66	71	72	73	5				
181	CONECT	67	68	69	70	5				
182	CONECT	74	78	154	6					
183	CONECT	75	76	77	6					
184	CONECT	77	75	79	81					
185	CONECT	78	74	79	80					
186	CONECT	79	77	78	82					
187	CONECT	83	85	86	8					
188	CONECT	84	88	89	10					
189	CONECT	85	83	91	92					
190	CONECT	87	93	94	8					
191	CONECT	88	84	97	98					
192	CONECT	90	95	96	10					
193	CONECT	91	85	93	101					
194	CONECT	93	87	91	102					
195	CONECT	95	90	97	99					
196	CONECT	97	88	95	100					
197	CONECT	103	109	110	14					
198	CONECT	104	106	107	16					
199	CONECT	105	117	118	16					
200	CONECT	107	104	115	116					
201	CONECT	108	113	114	14					
202	CONECT	109	103	111	112					
203	CONECT	111	109	113	121					
204	CONECT	113	108	111	122					
205	CONECT	115	107	117	120					
206	CONECT	117	105	115	119					

207	CONECT	123	126	127	12	
208	CONECT	124	129	130	11	
209	CONECT	125	131	132	12	
210	CONECT	126	123	133	134	
211	CONECT	128	139	140	11	
212	CONECT	129	124	137	138	
213	CONECT	131	125	133	135	
214	CONECT	133	126	131	136	
215	CONECT	137	129	139	142	
216	CONECT	139	128	137	141	
217	CONECT	143	144	145	2	
218	CONECT	144	143	146	147	
219	CONECT	145	143	148	149	
220	CONECT	146	144	150	151	
221	CONECT	148	145	150	152	
222	CONECT	150	146	148	153	
223	CONECT	1	2	3	4	5
224	CONECT	2	1	143	6	7
225	CONECT	7	2	8	9	10
226	CONECT	9	7	11	12	13
227	CONECT	13	9	14	15	16
228	CONECT	15	13	17	18	19
229	CONECT	22	3			
230	CONECT	23	21			
231	CONECT	24	21			
232	CONECT	25	21			
233	CONECT	26	20			
234	CONECT	27	20			
235	CONECT	28	20			
236	CONECT	29	4			
237	CONECT	32	30			
238	CONECT	33	30			
239	CONECT	34	30			
240	CONECT	35	31			
241	CONECT	36	31			
242	CONECT	37	31			
243	CONECT	40	19			
244	CONECT	41	39			
245	CONECT	42	39			
246	CONECT	43	39			
247	CONECT	44	38			
248	CONECT	45	38			
249	CONECT	46	38			
250	CONECT	49	17			
251	CONECT	50	48			
252	CONECT	51	48			
253	CONECT	52	48			
254	CONECT	53	47			
255	CONECT	54	47			
256	CONECT	55	47			
257	CONECT	57	18			
258	CONECT	59	58			
259	CONECT	60	58			
260	CONECT	61	58			

```
261 CONECT     62    56
262 CONECT     63    56
263 CONECT     64    56
264 CONECT     65     5
265 CONECT     68    67
266 CONECT     69    67
267 CONECT     70    67
268 CONECT     71    66
269 CONECT     72    66
270 CONECT     73    66
271 CONECT     76    75
272 CONECT     80    78
273 CONECT     81    77
274 CONECT     82    79
275 CONECT     86    83
276 CONECT     89    84
277 CONECT     92    85
278 CONECT     94    87
279 CONECT     96    90
280 CONECT     98    88
281 CONECT     99    95
282 CONECT    100    97
283 CONECT    101    91
284 CONECT    102    93
285 CONECT    106   104
286 CONECT    110   103
287 CONECT    112   109
288 CONECT    114   108
289 CONECT    116   107
290 CONECT    118   105
291 CONECT    119   117
292 CONECT    120   115
293 CONECT    121   111
294 CONECT    122   113
295 CONECT    127   123
296 CONECT    130   124
297 CONECT    132   125
298 CONECT    134   126
299 CONECT    135   131
300 CONECT    136   133
301 CONECT    138   129
302 CONECT    140   128
303 CONECT    141   139
304 CONECT    142   137
305 CONECT    147   144
306 CONECT    149   145
307 CONECT    151   146
308 CONECT    152   148
309 CONECT    153   150
310 CONECT    154    74
311 END
```

The above molecule contains 154 atoms and 153 bonds, making it extremely computationally expensive for regular(WHICH) QM calculations. This made utilizing

the large molecule as a trial system unreasonable due to the prohibitively long computation time for each conformation, assuming the conformation calculation would complete at all.

The below PDB file is the simplified butagermane with fully protonated Germanium atoms. As a significantly smaller system with only 14 atoms and 13 bonds, the relatively short computation time allowed the trial system to move with relative ease.

C.2   Code: ge4h.pdb

```
 1 COMPND       UNNAMED
 2 AUTHOR       GENERATED BY OPEN BABEL  2.3.90
 3 HETATM      1 GE    UNL     1       −3.520    1.842   −0.078   1.00   0.00
          Ge3−
 4 HETATM      2 GE    UNL     1       −1.368    2.888   −0.034   1.00   0.00
          Ge2−
 5 HETATM      3 GE    UNL     1        0.324    1.200    0.059   1.00   0.00
          Ge3−
 6 HETATM      4 GE    UNL     1        2.475    2.248    0.099   1.00   0.00
          Ge
 7 HETATM      5 H     UNL     1       −4.622    2.930   −0.135   1.00   0.00
          H
 8 HETATM      6 H     UNL     1       −3.699    0.985    1.202   1.00   0.00
          H
 9 HETATM      7 H     UNL     1       −3.621    0.932   −1.328   1.00   0.00
          H
10 HETATM      8 H     UNL     1       −1.258    3.797    1.217   1.00   0.00
          H
11 HETATM      9 H     UNL     1       −1.178    3.740   −1.314   1.00   0.00
          H
12 HETATM     10 H     UNL     1        0.213    0.288   −1.189   1.00   0.00
          H
13 HETATM     11 H     UNL     1        0.135    0.352    1.342   1.00   0.00
          H
14 HETATM     12 H     UNL     1        2.655    3.095   −1.186   1.00   0.00
          H
15 HETATM     13 H     UNL     1        3.578    1.161    0.165   1.00   0.00
          H
16 HETATM     14 H     UNL     1        2.574    3.167    1.343   1.00   0.00
          H
17 CONECT      1      2      5      6      7
18 CONECT      2      1      3      8      9
19 CONECT      3      2      4     10     11
20 CONECT      4      3     12     13     14
21 CONECT      5      1
22 CONECT      6      1
23 CONECT      7      1
24 CONECT      8      2
25 CONECT      9      2
26 CONECT     10      3
27 CONECT     11      3
28 CONECT     12      4
29 CONECT     13      4
30 CONECT     14      4
```

| 31 | MASTER | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 14 | 0 | 14 | 0 |
| 32 | END | | | | | | | | | | | | |

APPENDIX D

Two-Dimensional Rose-Potential Water

words

VITA

Gentry H. Smith

Candidate for the Degree of

Master of Science

Thesis: EXPLORING CRITICAL CONFORMATIONS

Major Field: Chemistry

Biographical:

Personal Data: Born in Olathe, KS in November 1993.

Education:
Received a Bachelors of Science in Chemistry at Southern Nazarene University in May 2016.

Completed the requirements for the degree of Master of Science with a major in Chemistry at Oklahoma State University in May 2018.

Experience:

Professional Affiliations:
American Chemical Society

Awards
Colonel Andre Whitely Scholarship in Chemistry