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 reflect the views of the author and are not endorsed by committee members or Oklahoma State University.

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

Table Page

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 -ϵ 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 -ϵ. 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

$$y(X_1, X_2) = 1 + \frac{\rho}{2\pi} \int y(X_1, X_3) f(X_1, X_3) \times \Big[y(X_3, X_2) f(X_3, X_2) + y(X_3, X_2) - 1 \Big] dX_3 \tag{1.4}$$

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

2.1 States and Properties of Ice

Ice is cool. Ice has many forms, each with unique environments and structures that give rise to similar and unique properties.

2.1.1 Bernal-Fowler Ice Rules

Bernal-Fowler Ice Rules are the basic rules for how water molecules interact in an ice structure. DETAILS ON BF PAPER

Basically, water's tetrahedral structure allows for four interactions on each molecule. The two protons allow for a hydrogen bond with a lone pair from a neighboring oxygen atom. Similarly, the oxygen atom's two lone pairs allow for a hydrogen bond with a neighboring proton. These rules are fairly rigid in the sense that every water molecule can interact with two oxygen atoms and two protons from four surrounding water molecules. These are also free-form in the sense that each of the four attached water molecules can occupy one of three rotational microstates, allowing for 81 possible configurations (including rotational duplicates).

2.1.2 Forms of Ice

While ubiquitous in the 'I_h' form, ice water has many known forms. As of the writing of this work, there are 17 established forms of ice. These forms usually occur in cubic, hexagonal, and orthorhombic crystal structures. The relationship between external pressure and temperature are the primary defining characteristics of which form will form in a given system. Do other characteristics come into play???????

2.1.3 Ice $\mathbf{I}_h$

As the most commonly found form on earth, ice I_h is a highly desired form for computational studies involving ice systems.

2.1.4 Efforts to Generate Ice $\mathbf{I}_h$

Has anyone else published efforts to generate Ice I_h? I'm sure someone has.

2.1.5 Comparison between Ice XI and Ice $\mathbf{I}_h$

While ice I_h is known as the most common form of ice found on the planet, it is much more difficult to computationally generate than an ice XI crystal. 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.

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.

2.2 Method Design

2.2.1 Overview

The big idea is to convert an easy-to-make ice XI crystal into an ice Ih crystal. Because the key difference in structure is the proton-orderedness, it might be possible to rearrange the water molecule orientations in a pseudorandom way to create an ice Ih crystal. This section walks through the method developed to convert ice XI into ice Ih, the results of initial testing, and imperfections discovered in the design.

2.2.2 Selection of Software Tools

Python was chosen as the language of the tool due to the versatility of the language and the ease of development due to the "pseudocode" written style of the language and the availability of scientific packages including SciPy and NumPy. Python version 2.7 was specifically chosen due to familiarity with the language. Crystal files where defined and saved as Protein Data Bank (.pdb) files as this format allows for defining multiple molecules within a larger structure with a simple X, Y, Z grid position format.

2.2.3 Generation of Source Ice XI

This is Dr. Fennell's method to create an ice XI pdb file. Basically, the ice XI unit cell of eight water molecules is repeated as desired to create a sufficiently large crystal. The primarily used crystal consists of a 3 x 3 x 6 unit cell repetition totaling 432 water molecules.

2.2.4 Source Ingestion

It is important that the crystal be read and stored in an efficient method to keep relevant information about each molecule easily accessible. As the file is read in, each molecule is stored as an entry in a multidimensional array where the first index is the molecule number. Further, the second index defines the molecule number where 0 is oxygen and 1 and 2 are the protons. The third, fourth, and fifth indices define the X, Y, and Z position coordinates.

2.2.5 Identifying Neighboring Molecules

Identifying the neighboring molecules proved computationally difficult. The most effective method is to find the closest four molecules by computing a distance calculation between every two oxygen atoms. This ensures every molecule is considered, but also presents significant hurdles. First, a distance calculation utilizes an extremely computationally-inefficient square root calculation, which can be ignored by instead calculating the squared-distance between molecules and finding the lowest values.

Second, molecules on the walls and edges of the molecule will not have four neighbors in the non-periodic crystal. This is accounted for by shifting all six sides to make a pseudo-periodicity for these edge cases. Those periodically-neighboring molecules are flagged with a shifting value in the neighboring atom array by specifying a translation in the x, y, or z axis values. Unfortunately, the necessary code to implement the periodically-neighboring molecule detections requires a major rewrite of the entire tool and has not yet been implemented.

Once these closest neighboring oxygen atoms have been discovered, the appropriate interacting tetrahedral position is identified by finding the closest of the four tetrahedral positions using the same squared-distance calculation with the four defined tetrahedral positions detailed in the next subsection.

2.2.6 Defining Tetrahedral Positions

An important aspect of pseudorandom selection is the existence of a bank of options. Using the ingestion portion to calculate and store all tetrahedral possibilities proves useful. For each water molecule, the first two tetrahedral positions are known by the positions of the two hydrogen atoms. The other two positions are found by rotating one hydrogen atom 120° twice about the vector from the oxygen atom through the other hydrogen atom and storing the resulting positions as tetrahedral positions three and four.

This does not produce an exactly correct tetrahedral position of potential hydrogen atoms due to the slight acuteness of the H-O-H bond created by the variance in repulsive forces between the two lone pairs of electrons and two hydrogen atoms. Fortunately, this difference is sufficiently small for visualization programs like Avogadro to still recognize hydrogen bonds between a rotated hydrogen atom and corresponding neighboring lone pair. Currently, the method relies on a very soft annealing process by a simulation package to minimize the effect of this hydrogen bond imperfection. Future versions of this method may account for the variations.

2.2.7 Pseudorandom Rearrangement of Water Molecules

Once the tetrahedral positions have been defined, each water molecule is ready to rotate. What may seem the most crucial step in this methods ends up being the most simple. As designed, the rotation of water molecules is as simple as using a stepwise iterator to pseudorandomly select two tetrahedral positions for the hydrogen bonds and store the new positions in a new crystal array. An extremely important note is that this rearrangement does not consider the orientations of neighboring molecules and likely introduces defects of hydrogen - hydrogen and lone pair - lone pair interactions. The likelihood of a defect-free interaction lattice forming is nearly zero and is assumed to have a great deal of defects within the lattice.

2.2.8 Detecting Hydrogen Bond Defects

After all water molecules have been rearranged, defects between incorrectly-interacting hydrogen bonds must be found and corrected. Discovering the defects relies on the detection of neighboring molecules and the appropriate interacting hydrogen atom or electron lone pair. As previously discussed, the initial data ingest records and detects the nearest water molecules and determines the tetrahedral position containing the interacting space, be it electron lone pair or hydrogen atom. From that data, the detection of a valid hydrogen bond is as simple as checking both all interacting tetrahedral positions and confirming that they both do not contain or lack a hydrogen atom. Additionally, each water molecule keeps a count of how many defects are present among the four positions. This allows for contextual changes during the correction step.

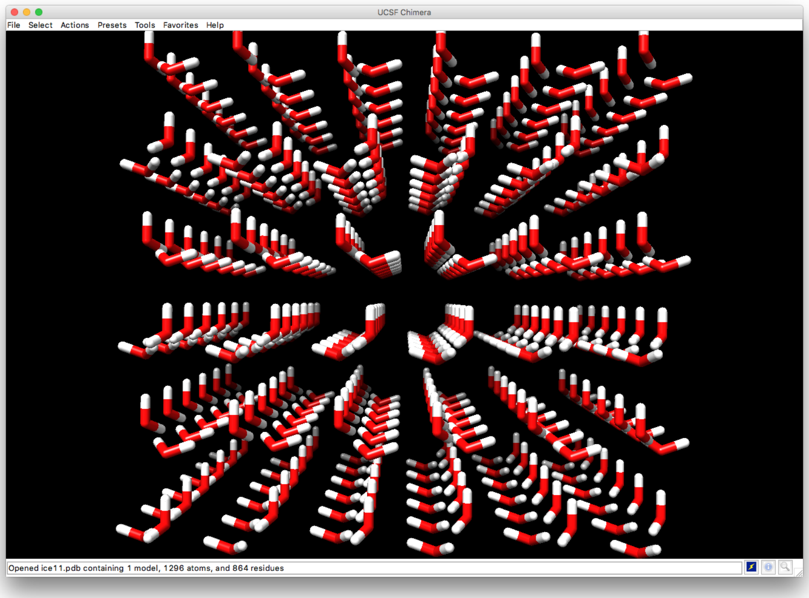
2.2.9 Correcting Hydrogen Bond Defects

Once the hydrogen bond defects have been discovered and marked, each needs to be corrected. The most direct approach to this is to sequentially walk through each defect and repeat the pseudorandom rotation until the number of defective regions is zero or a user-specified value. The current implementation sorts the defect list by the number of defects and attempts to fix the most defective molecules first. The most defective molecules may include defects impossible to solve by simple rotation, specifically when neighboring molecules have collectively directed three or four hydrogen atoms or electron lone pairs at the target water. These can only be solved by adjusting one or more of the neighboring molecules until the number of hydrogen atoms and electron lone pairs have balanced. Unfortunately, this high-defect problem can quickly escalate if the neighboring molecules contain the same problem of unbalanced hydrogen atoms and electron lone pairs. The current solution is to recursively check for and fix these impossible interactions first, but has not yet yielded a defect-free crystal in testing.

The current design of the method allows for the user to specify a threshold of defects as an average per molecule. For example, a threshold of 2.5 will allow a maximum of 3 defects on any given molecule and will continue to correct defects until the average number of defects per molecule is equal to or below 2.5. Because each of these defects will be counted twice, once for each molecule, the total number of defects in a crystal can be determined by multiplying the average defect value by the number of molecules and dividing by two. As of the current implementation, the method cannot reliably produce a crystal with a threshold below 2 as it will continue to search until the system runs out of available memory and crash without recording any new structure.

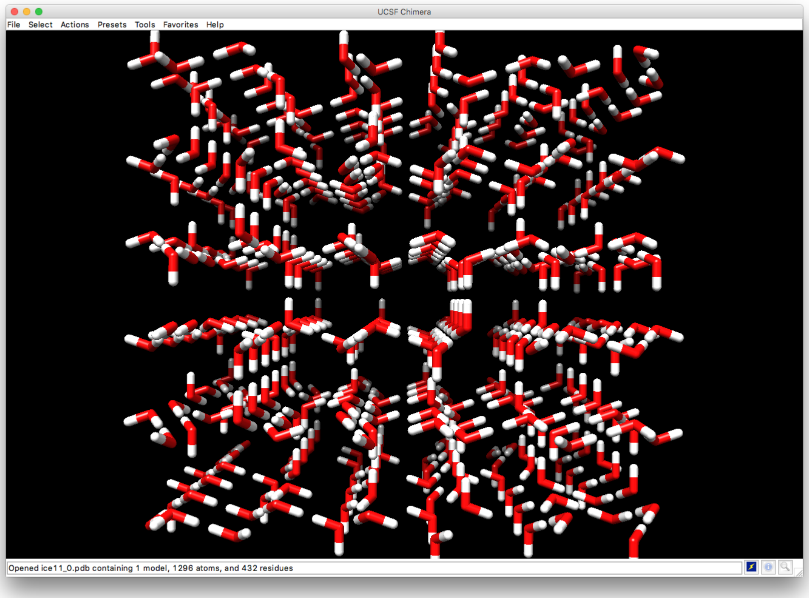
2.3 Results of Method

When supplied with an input ice XI crystal, an output structure with rotated water molecule orientations strictly consistent with ice Ih describes a success at the most basic level. An example before and after of the method is given in figures 2.1 and 2.2. The generated ice Ih has been rotated. When following the subsequent layers in the crystal, patterns emerge. Inconsistently, some rows of waters remain consistent. Some of these are a uniform rotation of both hydrogen atoms, while others are just one consistently placed hydrogen atom. Multiple trials yield internally unique results, yet all contain these strange consistencies. This may be due to some accidental pattern in the method's implementation.



Opened ice11.pdb containing 1 model, 1296 atoms, and 864 residues

Figure 2.1: "Before" image of Ice XI



Opened ice11_0.pdb containing 1 model, 1296 atoms, and 432 residues

Figure 2.2: "After" image of generated ice Ih

2.4 Comparison with Other Methods

2.4.1 benefits of own method over others

2.4.2 benefits of other methods over this

2.5 Comments on Limitations and Proposed Improvements

During the hydrogen bond defect correction step, a weakness in the design is that any clustering or regions of high defect density will not be noticed. This allows the existence of a highly-defective region within the larger structure that could potentially cause problems when the crystal is used in simulations. The prevalence and occurrence of these defects have not been studied, but seem a natural inevitability of statistics. A potential solution with partial development will score regions based on the number of defects as a weighted function expanding out from a central molecule for N connections. For example, consider a given molecule defined as level 1. The neighboring four molecules are defined as level 2, and continued onward excepting already-defined molecules out to an Nth level. The number of defects in each level can be counted and averaged. Then a depressive factor along the lines of $\frac{1}{level}$ can be used to diminish the value of defects further away from the first-level molecule. This would create a value for each molecule that shows the relative density of defects centered about that specific molecule and could even be plotted as a gradient change within the crystal. The equation would be something like:

$$Value = \sum_{l=1}^{N_{levels}} \left[\frac{1}{l} * \frac{1}{N_{molecules}} * \sum_{m=1}^{N_{molecules}} [N_{defects,m}] \right] \quad (2.1)$$

CHAPTER 3

Crystal and Liquid 2D Water at Interfaces

3.1 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.

3.1.1 Lennard-Jones Potential

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

$$V_{LJ} = 4\epsilon[(\frac{\sigma}{r})^{12} - (\frac{\sigma}{r})^{6}] \tag{3.1}$$

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 -ϵ 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 -ϵ. 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.

3.1.2 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.

3.1.2.1 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

$$c(X_1, X_2) = y(X_1, X_2)f(X_1, X_2) \tag{3.2}$$

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

$$h(X_1, X_2) = c(X_1, X_2) + \frac{\rho}{2\pi}\int c(X_1, X_3)h(X_3, X_2)dX_3 \tag{3.3}$$

to produce the overall relation

$$y(X_1, X_2) = 1+\frac{\rho}{2\pi}\int y(X_1, X_3)f(X_1, X_3)\times\Big[y(X_3, X_2)f(X_3, X_2)+y(X_3, X_2)-1\Big]dX_3 \tag{3.4}$$

3.1.2.2 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.

3.1.2.3 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.

3.1.3 Literature Review on Relevant Works

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

3.2 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.

3.3 Tools and Terms

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

3.3.1 OOPSE in 2D

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

3.3.2 Reduced Terms: 2D analogues

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

3.4 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)

3.4.1 Defining the Surface

Explain how to develop the surface in the program and how to build a custom surface

3.4.1.1 Manipulation of LJ Potential

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

3.4.1.2 Manipulation of bead spacing

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

3.5 Results

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

3.5.1 Future Work

Continued efforts to disrupt crystal growth on the surface.

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.

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	0.0000000
Cis-Trans-Trans	-15014.8246918	0.0222480	58.412124

Table 4.1: Collaborator's Hexagermane Energies by Conformation
(density functional theory, unknown basis set, energy in Hartrees and KJ/mol)

4.2 The Initial Problem: Germanium Study

During Fall 2017, Dr. Christopher Fennell was approached by Dr. Charles Weinert of OSU to continue a collaborative effort in sampling conformation energies of two germanium-based compounds of interest to Dr. Weinert's work. Seen as an opportunity to train a new graduate student in conformational calculations, this project was delegated to me. The initial focus was to create the two compounds in a 3D modeling program, save a file of each, run a conformation optimization program on a supercomputer, and read the output to report the findings. As detailed below, this work led to impossibilities, curiosities, and inconsistencies that resulted in a general solution and a discovery of a flaw in a popular computational program.

4.2.1 Parameters of Work and Previous Collaborator's Results

The two subject germanium-based compounds are very similar: a germanium backbone with terminal isopropyl groups and internal phenyl rings. One compound constituted a pentagermanium chain while the other a hexagermanium backbone. The molecular formula for both is $Pr_3^iGe(GePh_2)_nGePr_3^i$ where n equals 3 for the pentagermane or 4 for the hexagermane compounds, respectively. An example image of both compounds in their fully-trans configurations are provided in figures 4.1 and 4.2.

Dr. Weinert had worked previously with an additional collaborator who provided conformation data supplied in table 4.1. If I want to cite somebody, all I do is type in the citation for Bernal and Fowler (1933).

The approach of labeling the conformation shape of each compound, given the

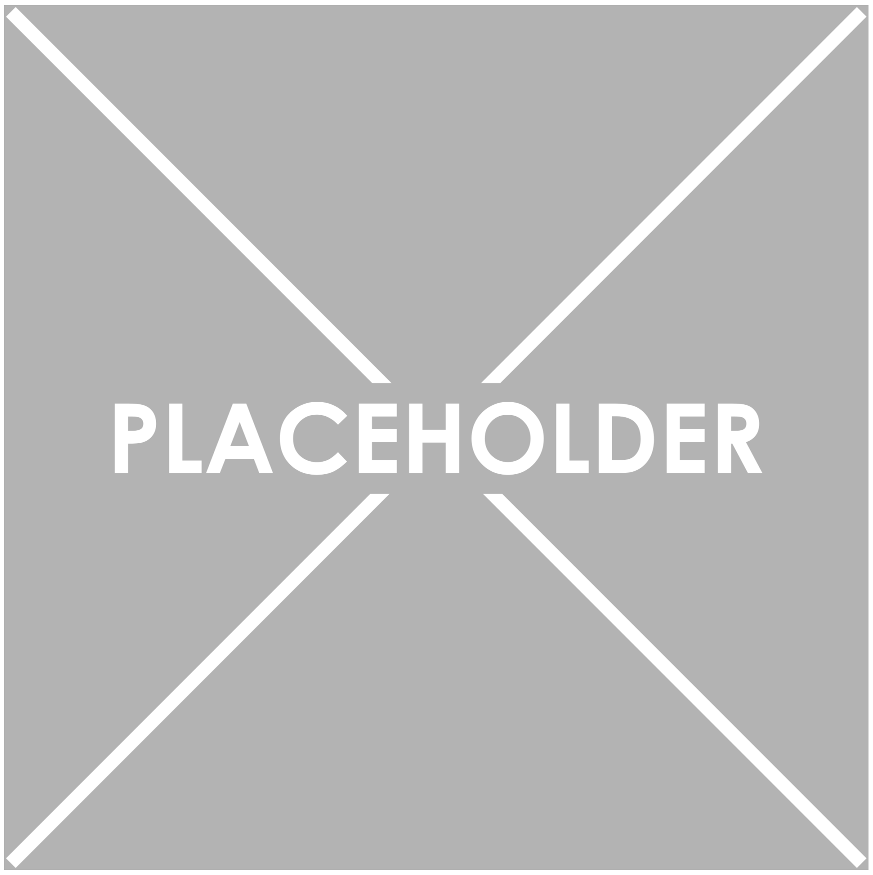


Figure 4.1: Fully trans configuration of pentagermanium-based compound.

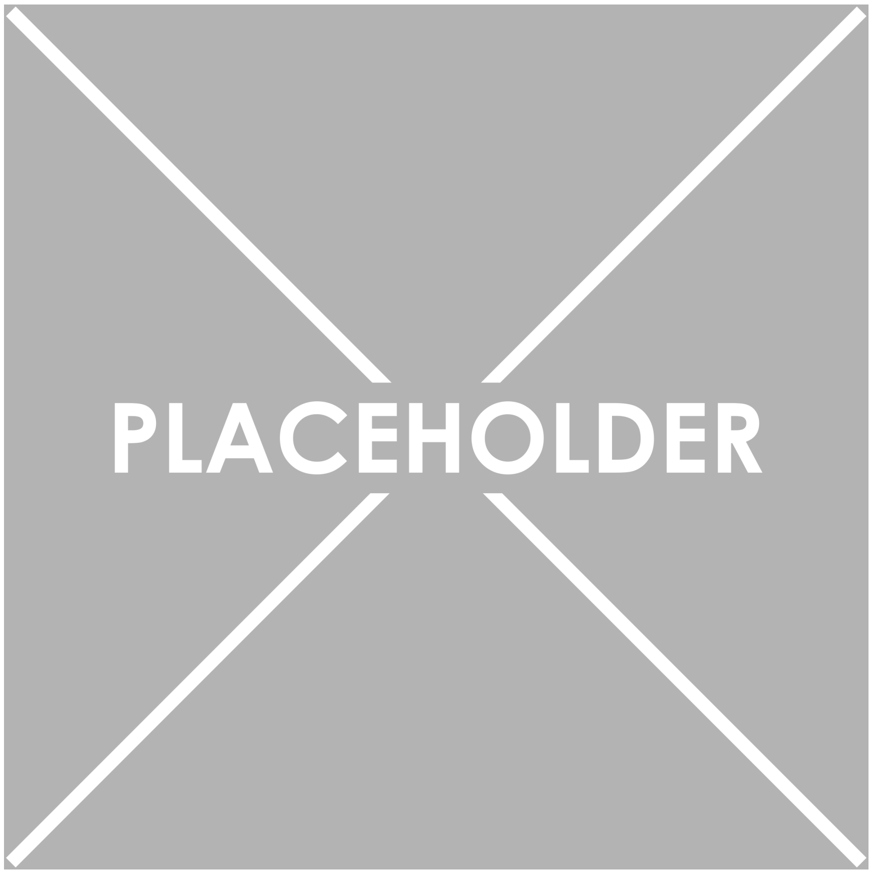


Figure 4.2: Fully trans configuration of hexagermanium-based compound.

many points of torsion, focuses on the backbone structure. As the raw data from the collaborator was not available, the general dihedral angles of cis and trans proved a vexing focus for initial efforts at conformer design. Using Newman projections like in figure REF! as a visual guide, each Ge-Ge bond was defined as cis or trans based on the relative angle produced by the two adjacent bonded Ge atoms to each subject Ge. Specifically, the bonds are marked cis if the most acute angle is 90° or fewer, and likewise trans if greater than 90° up to the maximum 180°. Effectively the cis and trans angles coincide with gauche and anti in organic structure nomenclature. Terminal germanium atoms are not considered as a part of the conformation state. This is partly due to the definition in labeling where the terminal germanium does not have an adjacent germanium for the measured relative angle, in addition to the assumed C_3 symmetry of the terminal Ge with three isopropyl groups reducing the relative effects of terminal germanium rotation. Effectively, only dihedrals formed by four consecutive Ge are given a cis or trans label.

4.2.2 Design and Approach to Solution

The initial approach involved an attempt at basic replication of the collaborative results. This design gradually became more complex as I

4.2.2.1 Design 1: Occam's Smallest Razer

With each non-terminal Ge-Ge dihedral initially labeled cis or trans for 0° or 180°, about 3 unique pentagermane and 6 unique hexagermane structures were built visually on a 3D visualization program (Avogadro). These were rotated without consideration for the phenyl rings populating the non-terminal Ge atoms. Each molecule was subjected to an energy minimzation in Gaussian 09 with the B3LYP hybrid function and STO-3G basis set as a single particle in a vacuum at otherwise default settings. See code in REF for a sample job command file.

Unsurprisingly, only the fully trans conformers successfully converged (a 22% success rate) into a stable form. These troubles were likely caused by the poor design of the initial conformers. With initial results, the conformer design was altered into a more systematic approach with some consideration for the phenyl rings.

4.2.2.2 Design 2: A Blunt Effort

In the second iteration of the conformer design process, a greater number of backbone conformers were generated. Instead of the simple 180° opposition between the cis and trans conformers, more intentional initial angles seen in Newman projections were selected. Specifically, the anti and both gauche angles were chosen for the natural local minima in a non-bulky molecule, with both gauche angles (60 and 300) labeled as cis and the anti angle (180) as trans. For initial conformer design, these backbone angles were limited to three positions: 60°, 180°, or 300°. For the hexagermane compound, these structures were sequentially labeled trans-trans-trans, trans-trans-cis, trans-cis-trans, et cetera until all major unique conformers were produced. For clarity, each conformer was identified by the dihedral angles (60-60-60, 60-60-180) in increasing order (Ge 1-2-3-4, Ge 2-3-4-5, Ge 3-4-5-6 dihedral). The phenyl rings on the non-terminal Ge atoms were left untouched from an initial steepest-descent minimization available from Avogadro ran in the fully trans conformer.

To prevent potentially strong interactions between adjacent phenyl rings, an additional steepest-descent minimization from Avogadro was initially ran with the conformer-defining Ge-Ge dihedral angles locked in place. Additionally, a visual inspection of the phenyl rings and manual adjustments were utilized on Avogadro to reduce the chance of a relatively high energy local minima conformer. The phenyl rings usually were settled in a form of pi stacking or some kind of perpendicular ring interaction, based on relative energy stability according to the immediate simple minimization available.

To further avoid backbone rotation restrictions, variations of the bulky molecules were also produced. These included versions where the phenyl rings were replaced by methyl groups and also where the isopropyl ends were additionally replaced by methyl groups. There intention in these designs were to observe the shift in relative energy between the sets of conformers to determine how significant of a role the phenyl rings and isopropyl groups played. These variations, along with the original form structures, were subject to the same calculations as in the first design: Gaussian 09, B3LYP hybrid functional, STO-3G basis set, no angle restrictions, single particle in a vacuum, otherwise default parameters. The results of these calculations are tabulated in table REFNAME

Immediately obvious in the table are the considerable number of nonconverged results. An unexpected bulkiness trend followed that a fully methylated variation of the structure was most likely to converge to a stable state, while the fully internal phenyl structures with methyl ends slightly reduced convergence and the original fully internal phenyl structures with isopropyl ends drastically reduced convergence. The common-sense expectation that the addition of the phenyl ends would reduce stability was not realized in these results. A deeper exploration into the change of stability is a promising avenue for future investigation, but was not further explored in this work. As higlighted in table REF, the lowest energy conformer for each structure form varied greatly, but never included the fully trans conformer and only once the collaborator-reported trans-cis-trans conformer as the most stable. Still, given the considerable amount of nonconverged conformers, a new design was necessary to further improve the scope of the lowest energy conformation search.

4.2.2.3 Design 3: Death by 1.59 Million Cuts

In the final version of the conformer generation effort, additional creation efforts were focused on the individual phenyl rings. The unfavorable interactions between the phenyl rings were considerable hurdle in the previous designs and a potential explanation for the large number of nonconverged structures, including the possibility that the terminal isopropyl hexagermane structures contained particularly unfavorable interactions among the phenyl rings. This third design sought to remove the uncertainty in phenyl ring bulkiness by applying the same approach as the backbone generation: create unique conformers of every backbone torsion and phenyl ring, limiting each torsion to one of three rotational positions following the Newman projection style. Unfortunately, this task proved prohibitively large.

As an explanation for the insurmountability of the problem, consider the hexagermane structure. The germanium dihedrals represent three rotatable bonds each with three initial positions. To include the phenyl rings would require the inclusion of eight new rotatable bonds each with three initial positions. Additionally, considering each terminal germanium's rotation while ignoring each isopropyl's rotatable bonds adds two initial positions each with three initial positions. Together, this creates a structure with 13 rotatable bonds each with three initial positions. The number of conformers follows as $3^{13} = 1,594,323$ initial conformers. Now we must consider the computational aspect of this many conformers. At 10 conformers rotated and generated per second and 16 KB per conformer, the initial conformers would require 44.3 hours and generate 25.49 GB of data just in the initial structures. At an average of 72 minutes per computation and 73.7 MB produced at B3LYP hybrid functional and STO-3G basis set and access to all 255 regular nodes of Oklahoma State University's Cowboy cluster running in parallel, the complete computation would generate 117.5 TB of data and require 312 days of continuous computation to determine a possible lowest energy conformer of this one molecule at a relatively low level basis set and theory. A request to utilize 100% of university supercomputer resources for nearly a year for the sake of determining the lowest energy conformer of one molecule would likely be rejected, so this task would likely require a time scale of years or even decades to produce with shared access to university resources. While conventionally considered a small molecule, the scale of conformers and computational requirements pushes this problem into the realm of Levinthal's paradox.

While this third design would have likely revealed the lowest energy conformer, or at least one considerably close the the exactly lowest energy conformer, the effort ultimate fails under its own weight. Even with efforts to truncate duplicate forms, the problem of scale remains. A reduction by 50% still requires a computation effort in the timescale of years or decades for the calculation of a single molecule. For an effective computational outlook, this system needs to be reduced by several orders of magnitude.

4.2.3 Scale Reduction Efforts

For a system with conformers on the millions scale and computations on the hour scale, a magnitude reduction in either aspect would improve the practicality of this design approach. For example, by simplifying the computational method from 72 minutes on average to 5 minutes on average, the overall computational requirement would be reduced by 92%, a full order of magnitude. Unfortunately, reducing the complexity of the method sacrifices the reliability of data. A potential solution here would be to create rounds of calculations at different complexities, where each sequential round restricts the pool of potential conformers. Ideally, the balance of the increasing computational complexity and the decreasing pool size would maintain a consistent computational requirement. For example, a new round using a higher functional theory and basis set at 5x computational requirement would ideally be paired with a reduction in conformer pool size by a factor of 5. This would produce a series of calculation sets with additive computational requirement instead of a magnitudinal expansion.

The natural next question lies within the reliability of basis sets and functional theories. It naturally follows that a less-accurate method should not be relied on while better methods exist. However, considering the scale of the conformer pool, it follows that a less accurate method would still produce energy values with a roughly similar internal consistency. For example, a 180°-0°-180° angle form of the hexagermane compound with parallel phenyl rings as modeled in figure REF will have intense syn interactions between some phenyl rings and will likely not yield a desirable energy value at any level of calculation while a fully trans form with perfect pi stacking phenyl rings will likely have a lower energy value at all levels of calculation. Therefore, at lower levels of accuracy, the extremely high energy conformers can be pruned from the pool early and drastically reduce overall computational requirements. A generic effort at producing a method in this style is detailed in chapter 2, while the remainder of this chapter concludes the efforts of calculating these germanium compounds.

4.2.4 junk section

4.2.5 Attempts at Simplification

One potential avenue of simplifying the process is removing all non-backbone groups.

4.2.6 Discovery of Program Flaw

This is a junk sentence.

CHAPTER 5

Sampling Conformation Landscapes by Rotatable Bond Degrees of Freedom

5.1 Introduction to Topic

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.

5.1.1 A Brief History on Conformation Landscapes

5.1.1.1 Levinthal's Paradox

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

Next: Levinthal golf courses by Ken Dill.

5.1.2 Computational Modeling

Introduce importance and impact. Bigly important.

5.1.2.1 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).

5.1.2.2 Hardware: Oklahoma State University's Cowboy Cluster

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

5.1.2.3 Software

Software to mention:

VisualizationAvogadro, UCSF Chimera

ComputingGAMESS, Gaussian, NWChem, Octopus

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

5.1.2.4 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).

5.1.3 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?

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 in figure 5.1.

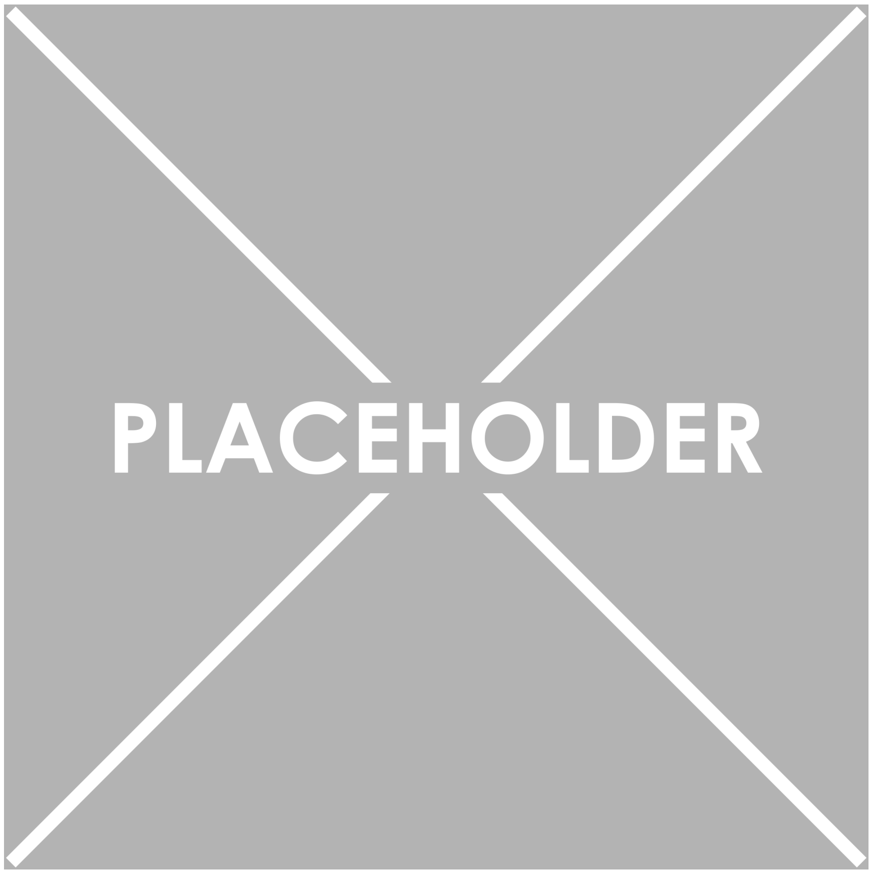


Figure 5.1: example VR chart (exploding like molecule/cell/fiber/muscle)

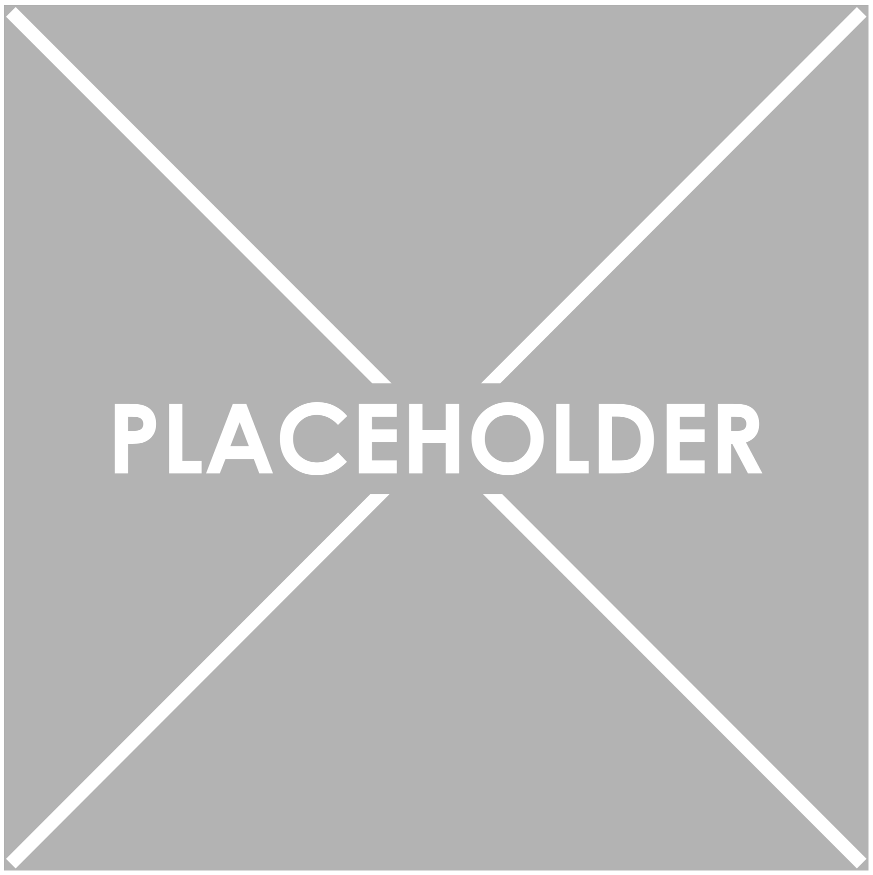


Figure 5.2: Flow of method design for variable resolution conformation landscape search.

5.2 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 figure 5.2.

This design, with implementation being a current work in progress, but should$^{\text{TM}}$ work as a cascade toward the lowest energy conformer in each case.

5.2.1 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?

5.2.2 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.

5.2.3 Inherent Complications

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

5.3 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.

5.3.1 Problems

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

5.3.2 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?)

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. This code is functional in a Python 2.7 environment with NumPy and SciPy packages included.

A.1 Code: PDBDisorganize.py

```

#!/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
        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):
                    #data[molecule][atom][X/Y/Z]
                    dataTable[iter0][iter1 % 3][i] = newData[i]
            elif data[0] == "HETATM":
                dataTable[iter0][iter1 % 3] = getDataHETATM(data)
            if iter1 == 2:
                iter0 += 1
                iter1 = 0
            elif iter1 != 2:
                iter1 += 1
    # print "DataTable: ", dataTable
    print "File read"
    return dataTable, pdbType


    # Split by index
    # if having a problem with reading data, check .pdb to see if data
    has a space between each value

# reads XYZ coordinate data from ATOM-type pdb
def getDataATOM(strLine):
    # print "Getting ATOM Data..."
    dataLine = strLine[5:8]
    # print "dataline: ", dataLine
    i = 0
    while i < 3:
        # print "dataline[", i, "]: ", dataLine[i]
        dataLine[i] = float(dataLine[i])
        # print "dataline[", i, "] type: ", type(dataLine[i])
        i += 1
    return dataLine


# reads XYZ coordinate data from HETATM-type pdb
def getDataHETATM(strLine):
    # print "Getting HETATM Data..."
    dataLine = strLine[5:8]
    # print "dataline: ", dataLine
    i = 0
    while i < 3:
        # print "dataline[", i, "]: ", dataLine[i]
        dataLine[i] = float(dataLine[i])
        # print "dataline[", i, "] type: ", type(dataLine[i])
        i += 1
    return dataLine


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

```python
    returnInt2 = rotateMolecule(newMol[1], newMol[2], (-1 * theta) )
    return [returnInt1, returnInt2]


# randomly selects new orientation, returns two unique ints, from 0 to 3
    inclusively
def newRandOrientation( positions ):
    # print "Changing orientation"
    randVal1 = random.randint(0,3)
    randVal2 = random.randint(0,3)
    while randVal1 == randVal2:
        randVal2 = random.randint(0,3)
    newMol = [ [ 0, 0, 0 ],
                positions[ randVal1 ] ,
                positions[ randVal2 ]  ]
    return newMol

# selects new orientation from list. Reduces computational overhead in
   re-orientation option traversal
def newSetOrientation( positions, pos1, pos2 ):
    newMol = [ [ 0, 0, 0 ],
               positions[ pos1 ],
               positions[ pos2 ] ]
    return newMol


# sets molecule coordinates so that oxygen is the origin
def zeroOrientation(source):
    # print "Zeroing Molecule..."

    oxy = source[0]
    hyd1 = source[1]
    hyd2 = source[2]

    # print "Oxygen pos: ", oxy
    # print "Hydrogen 1: ", hyd1
    # print "Hydrogen 2: ", hyd2

    zeroedOrigin = [0, 0, 0]
    zeroedHyd1 = [0, 0, 0]
    zeroedHyd2 = [0, 0, 0]
    for i in range(3):
        zeroedHyd1[i] = hyd1[i] - oxy[i]
        zeroedHyd2[i] = hyd2[i] - oxy[i]

    # print "Zeroed Hydrogen 1: ", zeroedHyd1
    # print "Zeroed Hydrogen 2: ", zeroedHyd2

    # return new molecule position
    newMol = [zeroedOrigin, zeroedHyd1, zeroedHyd2]
    return newMol

# resets the zeroed molecule to the original oxygen position
def resetOrientation(oxygenPos, molecule):
    # print "Resetting molecule..."
    rO = oxygenPos
    rH1 =[0,0,0]
    rH2 =[0,0,0]
    newMol = []
    for i in range(3):
        rH1[i] = molecule[1][i] + rO[i]
        rH2[i] = molecule[2][i] + rO[i]
        newMol = [rO, rH1, rH2]
    # print "Rebuilt Molecule: ", newMol
    return newMol

# rotates vector about axis for theta degrees
# Handler for rotationMatrix function below
def rotateMolecule(vector, axis, theta):
    rotMatx = rotationMatrix(axis, theta)
    return np.dot(rotMatx, vector)


# Creates Rotation matrix for a given axis and theta
# from stackoverflow user unutbu
# page: http://stackoverflow.com/questions/6802577/python-rotation-of-3d
    -vector
def rotationMatrix(axis, theta):
    """

    :type axis: list
    :type theta: union
    """
    axis = np.asarray(axis)
    theta = np.asarray(theta)
    axis /= math.sqrt(np.dot(axis, axis))
    a = math.cos( theta/2.0 )
    b, c, d = -axis*math.sin(theta/2.0)
    aa, bb, cc, dd = (a * a), (b * b), (c * c), (d * d)
    bc, ad, ac, ab, bd, cd = (b * c), (a * d), (a * c), (a * b), (b * d)
    , (c * d)
    return np.array( [ [ (aa + bb - cc - dd), ( 2 * ( bc + ad ) ), ( 2 *
    ( bd - ac ) ) ],
                        [ ( 2 * ( bc - ad ) ), (aa + cc - bb - dd), ( 2 *
    ( cd + ab ) ) ],
                        [ ( 2 * ( bd + ac ) ), ( 2 * ( cd - ab ) ), (aa +
    dd - bb - cc) ] ] )


# gets results from rotateAboutAxis plus two Hydrogens to get the
    tetrahedron positions
def getTetrahedronPositions(molecule):
    positions = [ [ 0 for i in range(3) ] for j in range(4) ]
    newMol = zeroOrientation(molecule)  # zero molecule
    positions[0] = newMol[1]
    positions[1] = newMol[2]
    newPos = getOrientations(molecule)  # get final two positions
    positions[2] = list(newPos[0])
    positions[3] = list(newPos[1])
    return positions                    # return all four positions


# checks distance of new positions from zero
def checkDist(posArray):
    distance = [0 for i in range(len(posArray))]
    for i in range(len(posArray)):
        distance[i] =  ( (posArray[i][0] * posArray[i][0]) +
                         (posArray[i][1] * posArray[i][1]) +
                         (posArray[i][2] * posArray[i][2]) )
        # print "Distance", i, ": ", distance[i]
    avg = 0
    for i in range(len(posArray)):
        avg += distance[i]
    averageDistance = ( avg / len(posArray) )
    # print "Average Distance: ", averageDistance
    return averageDistance


# prints data given a 3D table of water molecules
def printData(data):
    print "Data: "
    strData = [" O", "H1", "H2"]
    dimData = ["X", "Y", "Z"]
    bigAvg = 0
    numAtoms = 0
    for mol in range(len(data)):
        for atom in range(len(data[mol])):
            printStr = str(mol) + ": " + strData[atom] + ": "
            for dimension in range(3):
                printStr += dimData[atom] + ":" + "{:7.3f}".format(data[
    mol][atom][dimension]) + "\t"
            print printStr
        bigAvg += checkDist(zeroOrientation(data[mol])[1:])
        numAtoms += 1
        print ""
    print "total average distance: ", bigAvg / numAtoms


# checks validity of molecule
def isDefectiveCheck(err, neighborData, posData, index):
    # find nearby molecules (avg oxygen distance???)
    print "checking for defects at index", index, "..."
    print "neighbor indices: ", neighborData[index]
    returnBool = False
    neighbors = 4
    for i in range(4):  # count real neighbors
        if neighborData[index][1][i] == -1:
            neighbors -= 1
    if neighbors <= err:    # de facto good if num(neighbors) <
    maxErrAllowed
        # print "Fewer neighbors than allowed errors. de facto Good
    Orientation"
```

```
        returnBool = True
    elif neighbors > err:   # enough neighbors to require check
        # print "More neighbors than error threshold"
        defectCount = 0
        for neighbor in range(4):   # check each neighbor
            if neighborData[index][1][neighbor] != -1:  # skip over non-existant neighbors
                molA = posData[index]
                molB = posData[ neighborData[index][1][neighbor] ]
                oxyDist = getDistBetweenAtoms(molA[0], molB[0])

                if minHydrogenDistance(molA, molB) > oxyDist: # check for facing lone pairs
                    print "Double Lone Pair defect"
                    defectCount += 1
                    break
                else:   # check for facing protons
                    smallerHydrogenDistanceCount = 0
                    isDefective = False
                    for first in range(2):
                        if not isDefective:
                            for second in range(2):
                                newDist = getDistBetweenAtoms(molA[first + 1], molB[second + 1])
                                if newDist < oxyDist:
                                    smallerHydrogenDistanceCount += 1
                            if smallerHydrogenDistanceCount > 1:
                                print "Double Hydrogen defect"
                                defectCount += 1
                                isDefective = True
        # print "Defects found:", defectCount
        if defectCount > 4:
            print "IMPOSSIBLE AMOUNT OF DEFECTS DETECTED!!!! AHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHH"
        if defectCount > err:
            # print "Found a bad molecule!"
            returnBool = False
        else:
            # print "Molecule is within parameters."
            returnBool = True

    return returnBool


# randomly re-reorients molecule and neighbors, rechecks all
def rerunMolAndNeighbors(err, neighborData, posData, index):
    # print "Re-reordering molecule at", index
    # err - max errors allowed
    # neighborData - int[4] of neighbor indices
    # posData - array of all molecule position vectors
    # index - location of focus molecule in posData
    isGood = False
    timeCount = 0
    while not isGood:
        # re-rotate molecule through all positions (iterated through all orientations)
        positions = getTetrahedronPositions(posData[index])
        zeroedMol = newRandOrientation(positions)
        # print "isGood CHECK", isGood
        isGood, posData = iterThroughRotations(err, neighborData, posData, index)
        posData[index] = resetOrientation(posData[index][0], zeroedMol)
        if timeCount >= 13: # { (1 - 1/6)^n < 0.05 } says n = 17
            # BROKEN - need to rebuild
            # 0. evaluated molecule has too many defects
            # 1. reorient molecule statistically probable amount of times to cover all orientations
            # 2. Repeat 1. with neighbor 1
            # 2a repeat 1. with original molecule
            # 3. Repeat 2. with neighbor 2, 3, 4, as/if necessary
            for neighborIndex in range(4):
                if neighborData[index][1][neighborIndex] != -1:
                    positions = getTetrahedronPositions(posData[neighborIndex])
                    zeroedMol = newRandOrientation(positions)
                    posData[neighborIndex] = resetOrientation(posData[neighborIndex][0], zeroedMol)
                    # isGood = isDefectiveCheck(err, neighborData, posData, neighborIndex)
            isGood = isDefectiveCheck(err, neighborData, posData, index)
            if not isGood:
                isGood, posData = rerunMolAndNeighbors(err, neighborData, posData, neighborData[index][1][neighborIndex])
    finalData = posData
    return True, finalData

# iterates molecule through all possible rotations
def iterThroughRotations(err, neighborData, posData, index):
    isGood = False
    pos1 = 0  # tetrahedral position for H1
    pos2 = 0  # tetrahedral position for H2
    while not isGood or (pos1 != 3 and pos2 != 3):  # iterates through all orientations, stops if good orientation
        if pos1 != pos2:
            posData[index] = newSetOrientation(posData[index][0], pos1, pos2)
            isGood = isDefectiveCheck(err, neighborData, posData, index)
        if pos2 < 3:
            pos2 += 1
        elif pos2 == 3:
            if pos1 < 3:
                pos1 += 1
                pos2 = 0
    return isGood, posData
# determines minimum hydrogen distance between two atoms
def minHydrogenDistance(mol1, mol2):
    minDist = 100
    for first in range(2):
        for second in range(2):
            newDist = getDistBetweenAtoms(mol1[first+1], mol2[second+1])
            if newDist < minDist:
                minDist = newDist
    return minDist





# finds neighboring molecules of each molecule
def getNeighbors(data):
    returnData = [ [ [ 0 for i in range(4) ] for j in range(2) ] for k
    in range(len(data)) ] # data[molecule][distance,index][four values]
    for mol1 in range(len(data)):
        minDist = [100, 100, 100, 100]
        minIndex = [0, 0, 0, 0]
        for mol2 in range(len(data)):
            if mol1 != mol2:
                newMin = getDistBetweenAtoms(data[mol1][0], data[mol2
    ][0])

                bigIndex = indexOfBiggest(minDist)
                if newMin < minDist[bigIndex]:
                    minDist[bigIndex] = newMin
                    minIndex[bigIndex] = mol2
        for i in range(4):
            if minDist[i] >= 9:
                minDist[i] = -1
                minIndex[i] = -1
        # print "Four smallest Distances of", mol1, ": ", minDist
        # print "Four smallest Indices of", mol1, ": ", minIndex
        returnData[mol1] = [minDist, minIndex]
    return returnData


# finds distance between oxygen atoms
def getDistBetweenAtoms( mol1, mol2 ):
    distance = ( ( ( mol1[0] - mol2[0] ) * ( mol1[0] - mol2[0] ) ) +
                 ( ( mol1[1] - mol2[1] ) * ( mol1[1] - mol2[1] ) ) +
                 ( ( mol1[2] - mol2[2] ) * ( mol1[2] - mol2[2] ) ) )
    return distance

# gets index of largest item from a list
def indexOfBiggest(check):
    bigIndex = 0
    for i in range(len(check)):
        if check[i] > check[bigIndex]:
            bigIndex = i
    return bigIndex


# writes data to PDB file
def writeDataPDB(data, pdbType):
```

```python
    print "Writing Data to", str(pdbOUT)
    fileName = str(pdbOUT)
    output = open(fileName, 'w')
    if pdbType == 0:
        writeDataPDBATOM(data, output)
    elif pdbType == 1:
        writeDataPDBHETATM(data, output)
    output.close()


# Writes data to PDB file style = ATOM
def writeDataPDBATOM(data, inFile):
    iterator = 0
    for molecule in range(len(data)):
        for atom in range(3):
            iterator += 1
            outStr = "ATOM  "
            outStr += str(iterator)
            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)


# Writes data to PDB file style = HETATOM
def writeDataPDBHETATM(data, inFile):
    iterator = 0
    for molecule in range(len(data)):
        for atom in range(3):
            iterator += 1
            outStr = "HETATM"
            outStr += str(iterator)
            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:
            # print "fixing defects"
            while not isFine:
                # print "RerunMol"
                isFine, finalData = rerunMolAndNeighbors(err, connectedMolecules, finalData, i)
                # print "rerunDone"
    writeDataPDB(finalData, pdbType)
    # printData(newData)


badArgs = checkArgs(pdbIN, maxErr, pdbOUT) # stop in case of bad argument

# check input args
if not badArgs: # stop in case of bad argument
    print "Good Arguments, Initializing Reorientiation with", maxErr, " maximum defects"
    # testRun(pdbIN, maxErr, pdbOUT)
    runPgm(pdbIN, maxErr)
elif badArgs:
    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

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

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

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

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

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

```
HETATM 126 C 1 2.179 1.455 −2.458 0.00 0.00 C
HETATM 127 H 1 1.444 1.239 −0.432 0.00 0.00 H
HETATM 128 C 1 3.179 0.461 1.679 0.00 0.00 C
HETATM 129 C 1 2.005 0.227 4.186 0.00 0.00 C
HETATM 130 H 1 0.661 −1.197 3.265 0.00 0.00 H
HETATM 131 C 1 2.940 −0.568 −3.524 0.00 0.00 C
HETATM 132 H 1 2.840 −2.370 −2.334 0.00 0.00 H
HETATM 133 C 1 2.710 0.804 −3.567 0.00 0.00 C
HETATM 134 H 1 1.989 2.533 −2.491 0.00 0.00 H
HETATM 135 H 1 3.358 −1.081 −4.396 0.00 0.00 H
HETATM 136 H 1 2.944 1.370 −4.475 0.00 0.00 H
HETATM 137 C 1 3.067 1.105 3.998 0.00 0.00 C
HETATM 138 H 1 1.534 0.140 5.170 0.00 0.00 H
HETATM 139 C 1 3.650 1.229 2.740 0.00 0.00 C
HETATM 140 H 1 3.633 0.553 0.682 0.00 0.00 H
HETATM 141 H 1 4.480 1.926 2.585 0.00 0.00 H
HETATM 142 H 1 3.439 1.703 4.836 0.00 0.00 H
HETATM 143 C 1 −2.039 0.838 −1.804 0.00 0.00 C
HETATM 144 C 1 −1.525 0.195 −2.916 0.00 0.00 C
HETATM 145 C 1 −2.655 2.077 −1.927 0.00 0.00 C
HETATM 146 C 1 −1.618 0.802 −4.168 0.00 0.00 C
HETATM 147 H 1 −1.048 −0.789 −2.818 0.00 0.00 H
HETATM 148 C 1 −2.746 2.686 −3.175 0.00 0.00 C
HETATM 149 H 1 −3.084 2.566 −1.036 0.00 0.00 H
HETATM 150 C 1 −2.223 2.047 −4.296 0.00 0.00 C
HETATM 151 H 1 −1.210 0.296 −5.049 0.00 0.00 H
HETATM 152 H 1 −3.229 3.663 −3.275 0.00 0.00 H
HETATM  153  H             1       -2.292   2.524  -5.279  0.00  0.00
        H
HETATM  154  H             1       -2.539   1.081   2.827  0.00  0.00
        H
CONECT    3    1   20   21   22
CONECT    4    1   29   30   31
CONECT    5    1   65   66   67
CONECT    6   74   75    2
CONECT    8   83   87    7
CONECT   10   84   90    7
CONECT   11  124  128    9
CONECT   12  123  125    9
CONECT   14  103  108   13
CONECT   16  104  105   13
CONECT   17   15   47   48   49
CONECT   18   56   57   58   15
CONECT   19   15   38   39   40
CONECT   20    3   26   27   28
CONECT   21    3   23   24   25
CONECT   30    4   34   32   33
CONECT   31    4   35   36   37
CONECT   38   19   44   45   46
CONECT   39   19   41   42   43
CONECT   47   54   55   17   53
CONECT   48   17   50   51   52
CONECT   56   62   63   64   18
CONECT   58   59   60   61   18
CONECT   66   71   72   73    5
CONECT   67   68   69   70    5
CONECT   74   78  154    6
CONECT   75   76   77    6
CONECT   77   75   79   81
CONECT   78   74   79   80
CONECT   79   77   78   82
CONECT   83   85   86    8
CONECT   84   88   89   10
CONECT   85   83   91   92
CONECT   87   93   94    8
CONECT   88   84   97   98
CONECT   90   95   96   10
CONECT   91   85   93  101
CONECT   93   87   91  102
CONECT   95   90   97   99
CONECT   97   88   95  100
CONECT  103  109  110   14
CONECT  104  106  107   16
CONECT  105  117  118   16
CONECT  107  104  115  116
CONECT  108  113  114   14
CONECT  109  103  111  112
CONECT  111  109  113  121
CONECT  113  108  111  122
CONECT  115  107  117  120
CONECT  117  105  115  119
CONECT  123  126  127   12
CONECT  124  129  130   11
CONECT  125  131  132   12
CONECT  126  123  133  134
CONECT  128  139  140   11
CONECT  129  124  137  138
CONECT  131  125  133  135
CONECT  133  126  131  136
CONECT  137  129  139  142
CONECT  139  128  137  141
CONECT  143  144  145    2
CONECT  144  143  146  147
CONECT  145  143  148  149
CONECT  146  144  150  151
CONECT  148  145  150  152
CONECT  150  146  148  153
CONECT    1    2    3    4    5
CONECT    2    1  143    6    7
CONECT    7    2    8    9   10
CONECT    9    7   11   12   13
CONECT   13    9   14   15   16
CONECT   15   13   17   18   19
CONECT   22    3
CONECT   23   21
CONECT   24   21
CONECT   25   21
CONECT   26   20
CONECT   27   20
CONECT   28   20
CONECT   29    4
CONECT   32   30
CONECT   33   30
CONECT   34   30
CONECT   35   31
CONECT   36   31
CONECT   37   31
CONECT   40   19
CONECT   41   39
CONECT   42   39
CONECT   43   39
CONECT   44   38
CONECT   45   38
CONECT   46   38
CONECT   49   17
CONECT   50   48
CONECT   51   48
CONECT   52   48
CONECT   53   47
CONECT   54   47
CONECT   55   47
CONECT   57   18
CONECT   59   58
CONECT   60   58
CONECT   61   58
CONECT   62   56
CONECT   63   56
CONECT   64   56
CONECT   65    5
CONECT   68   67
CONECT   69   67
CONECT   70   67
CONECT   71   66
CONECT   72   66
CONECT   73   66
CONECT   76   75
CONECT   80   78
CONECT   81   77
CONECT   82   79
CONECT   86   83
CONECT   89   84
CONECT   92   85
CONECT   94   87
CONECT   96   90
CONECT   98   88
CONECT   99   95
CONECT  100   97
CONECT  101   91
CONECT  102   93
CONECT  106  104
CONECT  110  103
CONECT  112  109
CONECT  114  108
CONECT  116  107
CONECT  118  105
CONECT  119  117
CONECT  120  115
CONECT  121  111
CONECT  122  113
CONECT  127  123
CONECT  130  124
CONECT  132  125
CONECT  134  126
CONECT  135  131
CONECT  136  133
CONECT  138  129
CONECT  140  128
CONECT  141  139
CONECT  142  137
CONECT  147  144
CONECT  149  145
CONECT  151  146
CONECT  152  148
CONECT  153  150
CONECT  154   74
END
```

The above molecule contains 154 atoms and 153 bonds, making it extremely computationally expensive for regular 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

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

```
MASTER                0       0       0       0       0       0       0       0     14        0     14        0
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