2012

An ab initio investigation of boron bonding

Michael Paul Ver Haag
Iowa State University

An ab initio investigation of boron bonding

by

Michael Paul Ver Haag

A dissertation submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Major: Physical Chemistry

Program of Study Committee:
Thomas Holme, Major Professor
Mark Gordon
Theresa Windus
Nicola Pohl
Gordon Miller

Iowa State University

Ames, Iowa

2012

Table of Contents

CHAPTER 1. GENERAL INTRODUCTION

Overview

Chemists find certain types of bonding interesting and useful. Dative bonds are such a category. Boron is the only non-metal with significant dative bond chemistry. The focus of this dissertation will be exploring the dative bond formation of boron-containing compounds and electron-pair donating species. From a hybrid orbital theoretical framework, neutral boron-containing compounds exhibit an empty p-orbital, ideal for accepting a lone pair thus acting as a Lewis acid.

Boron's utility has been discovered in a wide variety of fields of study. In this chapter I will discuss a few of the most widely investigated experimental behaviors of boron containing molecules. Among these areas of inquiry boron has been shown to exhibit, physiological activity, the ability to act as binding mechanism for sugar sensing, the primary component of bacterial communication molecules, and as an important plant nutrient. I will describe some of the computational investigation methods that are utilized to study boron systems. This chapter will identify some of the limitations of those methods and provide a basis for justifying method selection for the studies described herein.

Common studies of boron

Boron Neutron Capture Therapy

One environment in which boron has been studied since the 1950 is boron neutron capture therapy[1–4](BNCT), a cancer treatment method. BNCT is a binary system, meaning that the two components of the treatment are not lethal independently but lethal when combined. This binary system allows for greater selectivity in the attack on the cancerous

cells. BNCT intervenes in the progress of cancer when ^{10}B is bombarded and absorbs thermal neutrons. The resulting ^{11}B undergoes decay to a high kinetic energy alpha particle and lithium ion (combined kinetic energy of 2.31 MeV) with a small portion of γ decay. (0.48 MeV) (Equation 1.1)[1] MeV energies completely ionize the product species.

$$^{10}B + {}^{1}n \rightarrow {}^{4}He^{2+} + {}^{7}Li^{3+} + 5e^{-} + \gamma \qquad \textbf{1.1}$$

BNCT is an effective treatment therapy because boron-10 has a very large effective cross section for capture of neutrons to undergo radioactive decay and boron readily absorbs neutrons where most other organic components do not. In addition the generated components have relatively low ionization tracks of only about 0.01 mm, or about the diameter of a cell. This means that nearby tissues are relatively unaffected by this decay, it is highly localized. Also it is very effective at causing apoptosis, cell death; both properties are highly desired for cancer treatments.

In a binary therapy both components need to be delivered to the treatment site. Thermal neutrons are an excellent source of neutrons, particularly for surface tumors. However, thermal neutrons penetrate poorly as they are scattered by hydrogen nuclei; therefore for deeper tumors epithermal neutrons must be used, which penetrate well.

Boron has a very low natural abundance in healthy cells. The isotopic abundance of boron is also of concern as well ^{10}B's natural abundance is only 19.8% ^{11}B does not absorb neutrons and therefore is inactive for this therapy, Delivery of boron atoms selectively to target sites is therefore challenge associated with this therapy. A process of enrichment to cancerous cells must occur. Boron can be delivered in many ways to several different portions of a cell. Some of the locations that are possible are the extracellular volume, the cell wall, and intracellular volume.[2] Incorporation into the internal portion of the nucleus is

the most difficult; however, inter-nuclear incorporation is the most effective for apoptosis.[5] Cancerous cells have a disproportionate nutrient uptake therefore have some preference in boron loading and selectivity.[6]

Many boron substituted amino acids have been synthesized for the incorporation method.[7–13] In searching for more effective uptake mechanisms, other surprising physiological activities of boron containing compounds were discovered. Small four-coordinate boron containing molecules were found to have anti-hyperlipidemic,[14–19] anti-Neoplastic,[20–24] anti-inflammatory,[25–30] anti-osteoporotic[31] and anti-obesity activity.[22] Low LD50 (lethal dosage in 50% of the population) values are common with boron containing complexes. The toxicity and low boron concentration of these compounds has caused other compounds to be used more frequently in modern BNCT; in spite of this, the unexpected activities are still of interest.

It has been hypothesized that competitive Lewis Acid/Base chemistry may be the responsible mechanism for the nature of the activity. A correlation between, anti-hyperlipidemic and anti-neoplastic activities, and boron-nitrogen Lewis acid/base dative bond strength in small molecules has been shown.[32] In chapter 2 the proposed mechanism for this activity will be described. The reaction pathway will be confirmed at a higher level of theory then previous works. The described pathway, an S_N2 mechanism, will also be shown to be lower in energy then an alternate mechanism, the water assisted transfer mechanism.

Quorum Sensing

Quorum sensing (QS) is a bacterial communication method. Bacteria excrete small molecules known as signaling molecules into their environment. When large enough

quantities of bacteria exist in the same area the concentration of those communicator molecules rises above a threshold concentration. Once that threshold has been reached this induces group activities by groups of bacteria. Quorum sensing has been shown to relate to biofilm formation, bioluminescence,[33] virulence,[34,35] conjugation,[36] sporulation,[37] and swarming motility.[38]

The signaling molecules can be species dependant or occur between diverse species. Among the few signaling molecules that have been identified is auto-inducer-2 (AI-2; Figure 1.1) this furanosyl borate diester is utilized by a large array of gram negative and gram positive bacteria.[39] AI-2 has been referred to as the universal signal due to the ubiquity of luxS. LuxS is the gene responsible for the creation of AI-2. It is not clear that AI-2 is always used as a signaling molecule despite its ubiquity.[40]

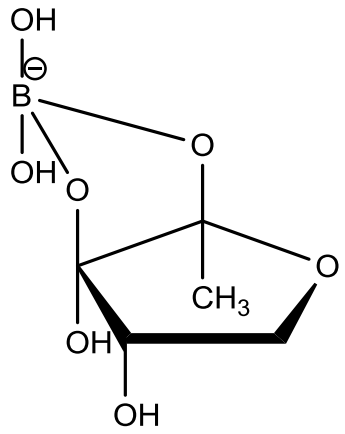


Figure 1.1 Auto-Inducer-2 (AI-2)

Much research has gone into identifying ways to inhibit quorum sensing molecules to help mitigate the undesirable effects of QS. The universal nature of the AI-2 signaling molecule is a prime target for study. The precursor product of AI-2 is call Pro-AI-2 and is an

erythro-furanose, and has much similarity to saccharide crosslinking that will be discussed in the next section. Quorum sensing has been shown to be inhibited by boronic acids.[41] The inhibition may occur due to a borate diester forming with large functional groups that can preclude the auto-inducer from binding to is active site. It consequently becomes of interest to study the importance of boron in signaling.

Saccharide binding, sensing, and recognition

Boronic acids and borates readily form esters and diesters via a dehydration reaction to diols. This binding is tight and reversible. In the last 20 years, an enormous amount of research has gone into using boronic acid functional groups for sensing and carbohydrate recognition.

"Not all boronic acids bind with the diol moiety with the same affinity and not all diols bind to a boronic acid with the same affinity"[42] The differentiation property makes boronic acids ideal for sensing and recognition. To utilize this property it is important to completely understand the intrinsic driving factors associated with borate binding.

Boronic acids and boronates have been utilized in a wide array of novel fashions, from attaching florescent components, to developing a non invasive glucose detector for diabetics, to stimuli responsive controlled release systems, to boron mediated drug delivery.[43–46] The wide array of research lines and intense current investigations indicates that this will be an important field for many years to come.

Studying saccharides poses significant computational challenges. Their variability makes configurational searching problematic. The complex nature of modeling of saccharides will be described in chapter 3. I will discuss how a tool was created to help

characterize saccharides, and how it can be implemented. This tool will be useful in future saccharide inquiries.

Rhamnogalacturonan-II

Boron has been known to exist in plants since the mid 1800's.[47] At the turn of the century it was known to be ubiquitous in the plant kingdom.[48] By the 1920s boron had been determined to be an essential micronutrient to plants.[49] Boron content has been found to vary between plants of different species and types; from low boron requirements in graminaceous monocot plants, such as barley to moderate boron need in dicots like, peas, beets and lettuce, to high boron needs in latex forming plants, such as dandelion and poppy.[50] The variability in boron uptake requirements has caused much research to go into identifying the optimal boron content for individual plant types.

The boron content of soil is hard to maintain. Boric acid is easily leached from soil with too much water while too little water causes boron to crystallize and become intransigent. Boron uptake has been determined to be a passive uptake mechanism of boric acid by root tips.[51] Soil pH also effects boric acid availability. As pH increases boric acid binds to soil material and consequently is less available to plants.

Deficiencies in boron can lead to a wide variety of atypical plant growth patterns. Boron deficient plants exhibit stunted growth, misshaped leaves and discoloration. Brittle stems and leaves are produced, and fewer reproductive organs (i.e. buds and flowers) form. Stems can also form hollow. Boron requirements are greater for reproductive vegetative parts and for developing and growing parts of plants.[52,53]

Boron plays a critical role in plant cell wall structure. The plant cell wall is large rigid network of hemicellulose strings embedded in a pectic framework. The pectic fraction is composed of three covalently linked polysaccharides responsible for pectic infrastructure; homogalacturonan (HG), rhamnogalacturonan I (RG-I), and rhamnogalacturonan II(RG-II).[54,55] Homogalacturonan is composed of α-(1-4)-linked D-galacturonic acids to form long chains. RG-I and RG-II are functional groups attached to HG chains in a short length of the chain. Individual RG-II monomer units (mRG-II) are capable of using a boron atom to dimerize to form dRG-II-B.[56]

The RG-II polysaccharide is plays a unique role in the structural rigidity of the plan cell wall.[57,58] The pectic infrastructure is constrained (i.e. becomes rigid) when boron has bridged two RG-II monomer units. The mechanism for modulating rigidity is simply a change in pH.[56] At low pH's the crosslink is disrupted and mRG-II is formed. This allows the cell wall's porosity to change and the cell to grow or change as needed. When the pH returns to neutral ranges d-RG-II is formed and rigidity returns. The pH change does need to be significant. RG-II remains in its dimer form until beyond a pH of 4 in *in vitro* studies.[56]

RG-II structure is a highly conserved polysaccharide in plants. [59] Considerable effort has gone into identifying the glycosyl residues of RG-II **(**Figure 1.2).[60–72] The RG-II backbone contains at least 8, α-(1-4)-linked D-galacturonic acids. There are four side chains labeled A-D **(**Figure 1.2**)**. Side chains A and B have Api*f* linkages to the back bone, however the boron cross link only occurs at side chain A (Figure 1.3). The crosslinking furanosyl apiose is highlighted in red. Side Chain B is not consistently structurally conserved.[59]

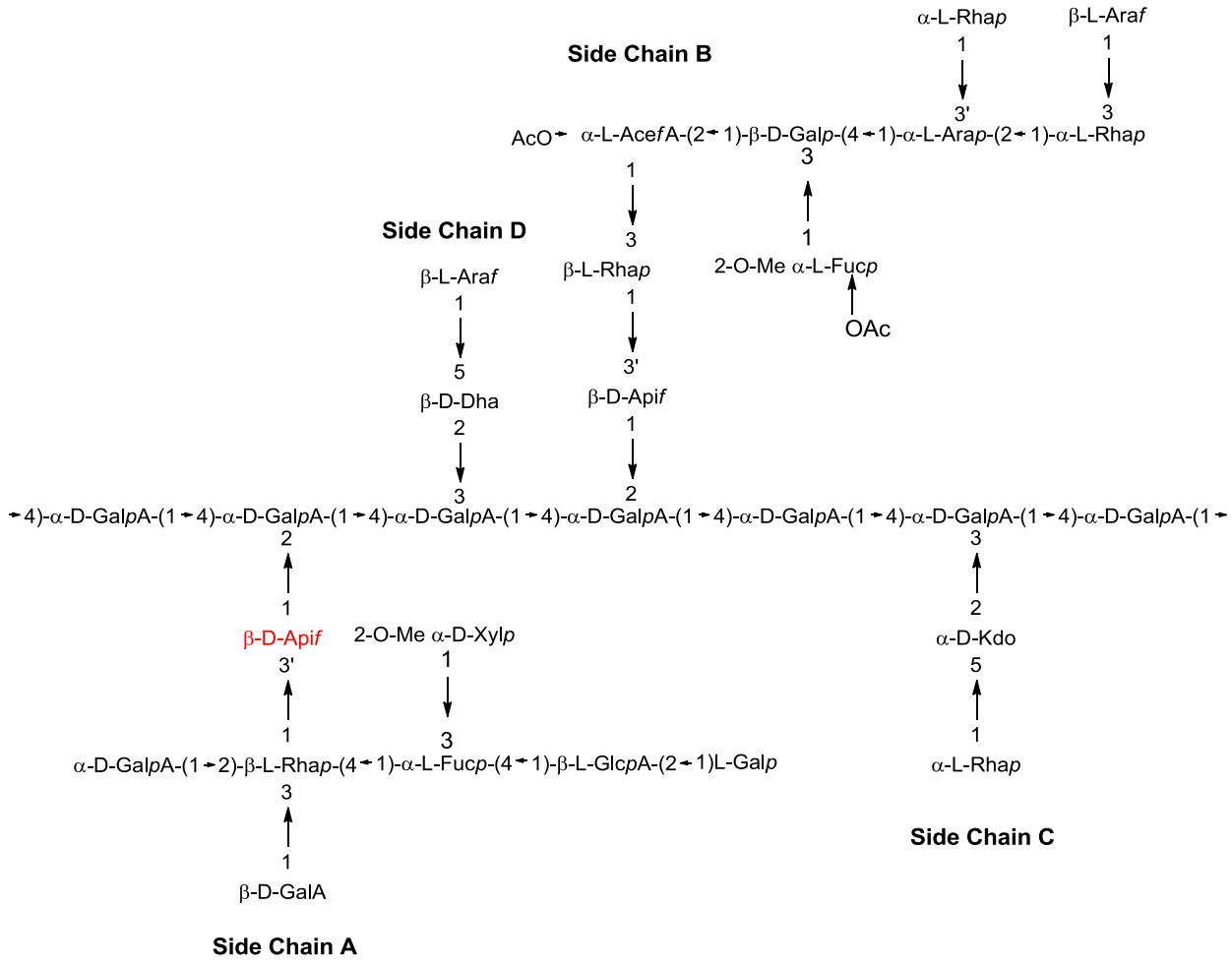


Figure 1.2 RG-II, residue structure, crosslink occurs at β-D-Api*f* (red)

The ubiquity of the rhamnogalacturonan II crosslink mechanism implies that it developed early in plant evolution and it is likely that it is fundamental to plant cell wall structure. β-D-erythro-furanoses readily form borate esters.[73] β-D-Api*f* is the only component of RG-II with this configuration. The crosslink always occurs at the side chain A apiosyl residue.

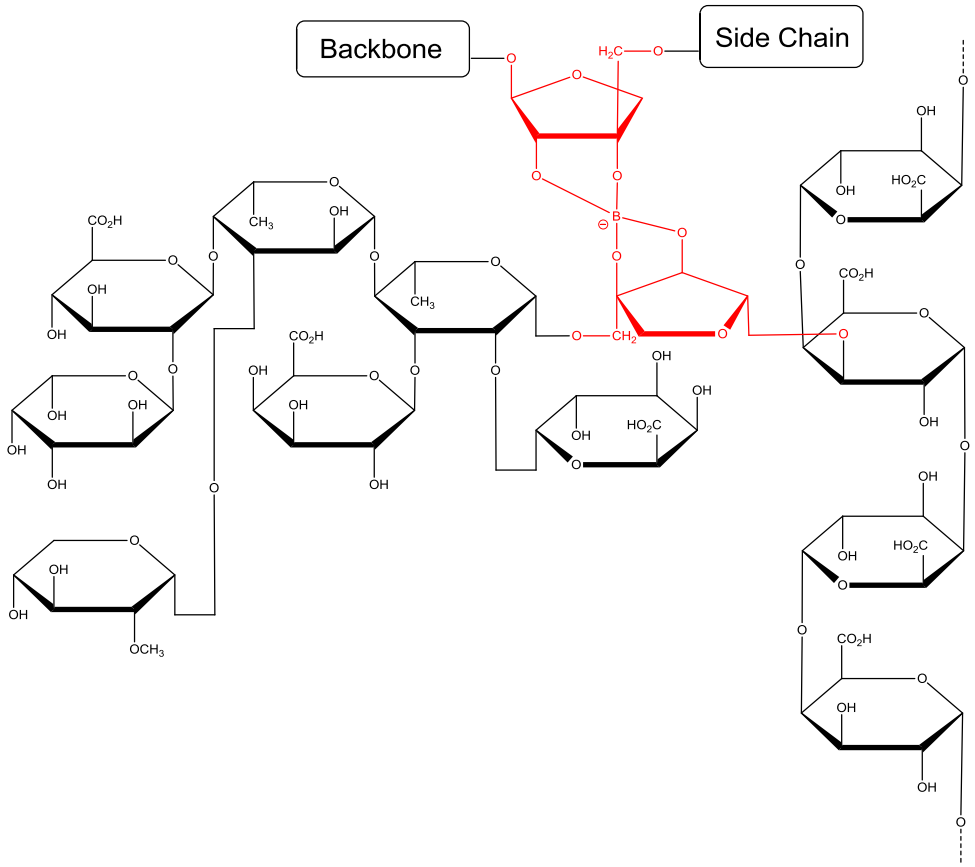


Figure 1.3 Rhamnogalacturonan II, detail side chain A

The boron of the borate diester is a chiral center. There are two possible configurations for the crosslink: R and S. Currently the orientation of the crosslink is un assignable however the two configurations appear to be in equal distribution[74] (Figure 1.4).

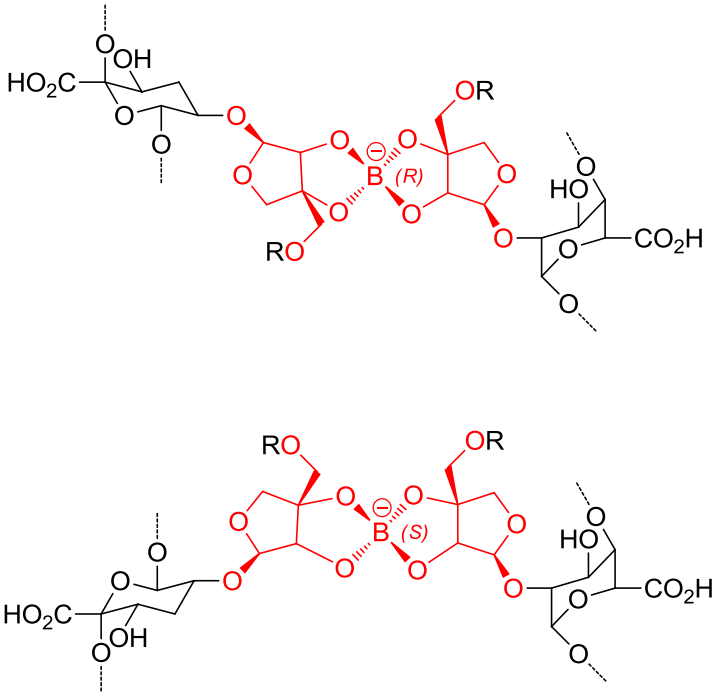


Figure 1.4 R (Top) and S (Bottom) configurations of borate crosslink of apiose in RG-II (Shown in wedge dash projection to emphasize stereochemistry)

The cross link is stabilized by large atomic radius divalent cations.[56,75] The role of the cations is poorly understood.. Pb^{2+}, Ba^{2+} and Sr^{2+} significantly improve the formation of dRG-II-B. Ca^{2+} at higher concentrations, as well as other cations La^{3+}, Eu^{3+} Ce^{3+}, Pr^{3+}, or Nd^{3+} have been shown to stabilize as well.[76,77] The binding site of the divalent cation is not known.

RG-II is a major polysaccharide component of wine. This RG-II component comes from the cell wall by microbial glycanase during fermentation. This has become the preferred mechanism for extraction of RG-II. Much work continues in the biological realm concerning RG-II particularly with respect to enzymatic activity and structural activity.

In Chapter 4 a model of the apiose crosslink will be presented. I will discuss the potential energy surface along the crosslink pathway from products to reactants. An

assignment of R and S configurations will be proposed based on calculated thermodynamic values of each of the two configurations.

Computational investigation methods, description, selection criteria, and justification

Introduction

There are many methods that could be used to attempt to study these boron containing systems. Some of the most common methods are molecular mechanics, molecular dynamics, density functional theory, and ab inito methods. Each varying level of theory has certain strengths and drawbacks in accuracy and compute time as well as other constraints. I will discuss some of these components to elucidate the manner of method selection used in the studies reported here. In the next section I will begin with methods that were not chosen and the reasons why followed by the methods that have been selected.

Method selection is dependent on the framework that we are attempting to study within. The accuracy achievable by computational chemistry models is constrained by available computer resources. It therefore is important to evaluate methodology before attempting any calculation. The large systems of interest to us, if fully modeled, generally exceed the resources we have to investigate them. We must therefore, find an alternate means to understand the large systems of interest. For example, if we can thoroughly understand boron's role in small systems, then we may be able to scale our understanding to larger systems and elucidate the role that boron plays there. In using this research motif, our method for understanding large systems is predicated on understanding derived from smaller systems.

Considered Methods

Molecular Mechanics

The molecular mechanics method is often employed for modeling large systems due to its cost effectiveness in computational time. This method renders large systems including proteins, enzymes and polysaccharides more manageable computationally. The speed of calculation can come at a significant cost to accuracy depending on the system and how closely it matches parameterized values.

Molecular mechanics (MM) is a method that uses classical mechanics to model molecular systems. It evaluates potential energy using a force field approximation[78]. This essentially sums a series of potential energy terms. Each potential energy function maintains a classical energy form based on adjustable parameters. The parameters are adjusted to match experimental or computed results. The parameterization must be internally consistent. MM performs well for systems that are well parameterized. If a system is of study is not near parameterized values or has not been sufficiently parameterized in MM its results may be poor. Relatively little parameterization has been done on boron dative bond, particularly with respect to boron bonding to sugars. Molecular mechanics must also conserve bonds, as this is the method by which energy is conserved in the model.

Molecular Dynamics

Molecular Dynamics (MD) simulates the motion of atoms over time by calculating trajectories in time evolved steps. MD calculations may be used with any level of theory to determine forces and potential energies[79]. Often MD utilizes molecular mechanics force fields to calculate its trajectories. In this case MD offers code that has the same constraints as

MM but is more expensive computationally. MD allows us to get thermodynamic data and to study a larger sampling of our system. It can also provide dynamics for systems. This method essentially has the same force field problems as MM and was not pursued as a potential research avenue.

Density Functional Theory

Density functional theory is an *ab initio* method. DFT obtains electronic structures for the electron density by solving the Kohn-Sham equation (electrons interacting with a static potential) rather than approximating the electronic wave function. DFT is commonly computationally cheaper than other more traditional *ab initio* methods. DFT works well with extended systems. Density functional theory derives orbitals from the electron density instead of starting with derived canonical orbitals. Considerable effort is expended to optimize functionals for DFT[80]. Four-coordinate boron is somewhat exotic so the accuracy of DFT methods is not as well established for systems that contain this motif. Choosing functionals for DFT can also be difficult because improvements to functionals are hard to compare. Finally, DFT has been shown to have challenges describing boron containing compounds.[81–83]

Selected methods

Introduction

There are many *ab initio* methods and several will be described and compared in this section. In general *ab initio* methods are used for their accuracy and versatility in

computations. For these reasons we use ab initio methods, to better understand structural information in small-scale models.

Hartree-Fock

The Hartree-Fock (HF) method is the starting point for most *ab initio* methods. All the systems examined herein are closed shell systems. The most efficient type HF calculation for closed shell systems is Restricted Hartree-Fock (RHF)[84]. Several approximations are made in HF. The approximation of most frequent concern is that it does not account for electron correlation. The coulombic interaction between elections is a significant, though consistent component of the electronic potential energy.

One key quality of HF methods is that they are variational[84]. The variational principle indicates that the calculated expectation value of the exact Hamiltonian operator on any approximate normalized wave function will be greater than the true ground state energy. i.e. anything that it is possible to do to the wave function that reduces the energy is closer to the correct energy because no energy that is calculated will ever be lower than the real value. In addition to being variational Hartree-Fock is also comparably fast; HF calculations scale as N^4 where N is the number of orbital functions[85]. In the works reported here the RHF level of theory was used for preliminary optimization and geometry searching.

Møller-Plesset Perturbation Theory

Møller-Plesset perturbation theory provides an estimate of electron correlation relative to the HF method[86,87]. The electron correlation perturbation causes this theory to be no longer variational; however, it typically obtains better approximations of the energy expectation value than HF. Møller-Plesset perturbation is a Taylor expansion. Expanding to

the second order in the power series (MP2) is frequently used. As with any Taylor expansion the first term is the predominant term and therefore the largest term in recovering electron correlation is in the MP2 term. Expanding to the third order term (MP3) in the power series typically and inconsistent over estimates electron correlation and improperly identifies geometries[88,89]. Expanding to the fourth term (mp4) is computationally expensive. Due to relative cost efficiency as a means for recovering some of the correlation effects, MP2 was chosen as the primary method for this work for electron correlated level of theory for its balance between accuracy and cost efficiency. In this work all structures of concern are optimized to at least the MP2 level of theory. MP2 scales as N^5 [85].

Coupled Cluster

Coupled cluster (CC) is non-variational method that is capable of chemical accuracy, the accuracy required to make realistic chemical predictions. This method converts the Slater determinant into a linear combination of all possible excited Slater determinants, thereby allowing electrons to avoid each other, and therefore adding electron correlation[90,78]. For our purposes we chose CCSD(T). This means that all the single and double excitations of electrons are treated fully while triple excitations are treated as a perturbation. This method is a computationally expensive method (scales as N^7)[85], and consequently was only used in a limited fashion.

Property and Confirmation calculations

Hessian Matrix

When identifying potential energy minima, the way to determine if a structure with a stationary point is actually a minimum is to compute the gradient. The second derivative with respect to energy square matrix is an example of a Hessian matrix or the force-constant matrix. This matrix is used in many ways. The Hessian Matrix is approximated in the quasi-Newtonian optimization methodology as a means to identifying minima and transitions states. It also helps approximate the curvature during the search portion of optimization, this is why it's updated at each optimization step. The Hessian allows us to compute vibrational frequencies and determine if the curvature is positive definite. Positive definite functions indicate that the potential energy surface gradient is positive or has a positive curvature in all directions indicating it is a genuine minimum. A Hessian matrix is used to confirm that saddle point calculations have only one negative eigenvalue. Finally, Hessian matrices provide key data to generate thermodynamic data.

Thermodynamics

By calculating the second order derivatives it is possible to calculate the vibrational frequencies and thereby obtain the vibrational partition function. Rotation, translation, and electronic partition functions are readily obtained from information that is easily derived from electronic structure calculation.

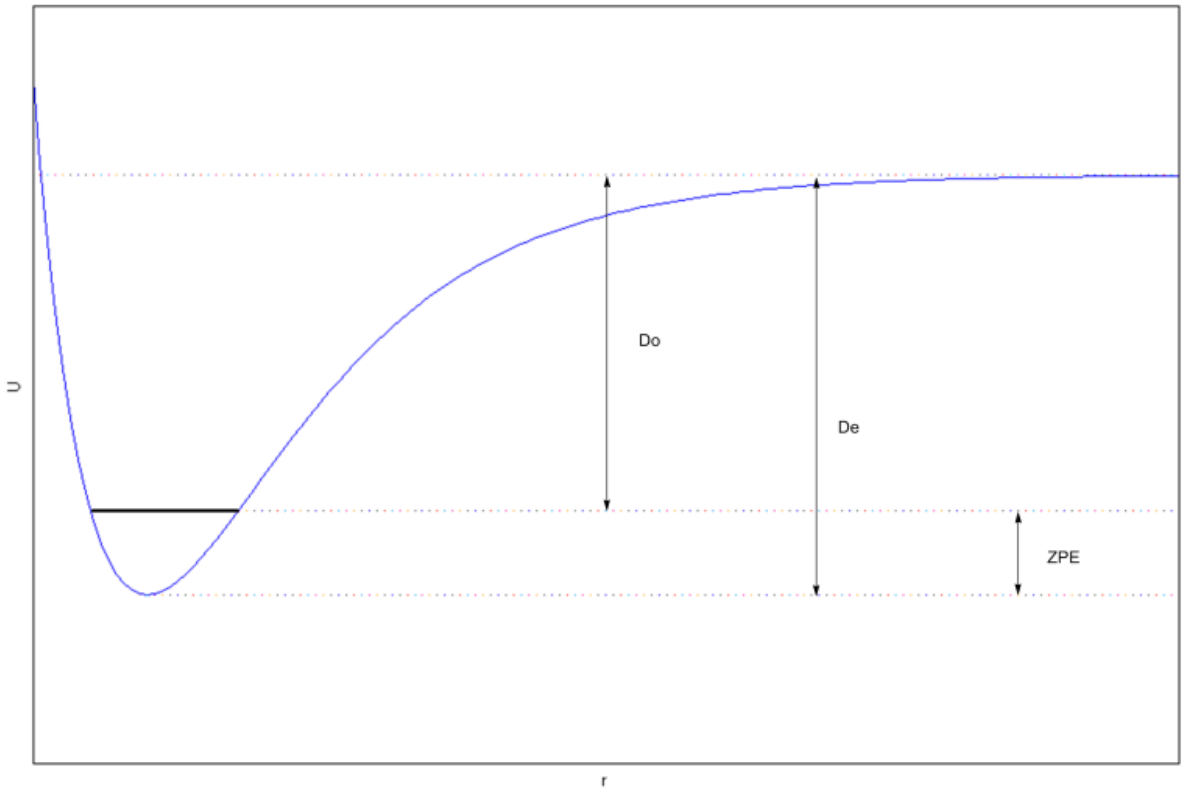


Figure 1.5 Electronic Energy

In general the absolute energies calculated create a framework for the potential energy surface. U_e is the absolute energy calculated by GAMESS (Figure 1.5). U_e is also called D_e or the dissociation energy at the equilibrium radius. $D_0 = D_e - E_{ZPE}$ The internal energy at T=0 is then $U_0^0 = N_{A*} D_0$

ΔU_e is the difference in energy between products to reactants. At T= 0, $\Delta U = \Delta E = \Delta H = \Delta G$ These values can be thermally corrected to any temperature, however the correction is small particularly in reference to U_e and D_0 and even ΔU_e and ΔD_0.

$$\Delta E^{298} = \Delta E_e^{\,0} + \Delta(\Delta E_e)^{298} + \Delta E_v^{\,0}\ \Delta(\Delta E_v)_{298} + \Delta E_r^{\,298} + \Delta E_t^{\,298} \quad \textbf{1.2}$$

$$\Delta H^{298} = \Delta E^{298\,+}\ \Delta(PV) \quad \textbf{1.3}$$

$$\Delta H^{298} = \Delta E^{298} + \Delta E_T + \Delta nRT \quad \textbf{1.4}$$

$\Delta E_e^{\,0} = \Delta U_e$ is the energy difference between products and reactants at 0K

$\Delta(\Delta E_e)^{298}$ is the change in electronic energy difference, with the exception of some rare cases of low lying excited states this should be zero.

$\Delta E_v^{\,0}$ is the difference in zero point energies between reactants and products.

$\Delta(\Delta E_v)_{298}$ is the change in the thermal vibrational correction at 298K

$\Delta E_r^{\,298}$ is the rotational energy difference between products and reactants

$\Delta E_t^{\,298}$ is the translational energy difference between products and reactants

ΔE_T is simply a collection of thermal correction terms.

GAMESS simplifies these calculations. GAMESS prints thermochemistry values that are formatted in such a way that $\Delta U_e + \Delta H$ from products and reactants (which does account for ZPE) = ΔH^{298} for the systems.

Basis Set

At each level of theory, a set of mathematical functions needs to be used in order to model the molecular orbitals. These functions are called the basis set. When considering a basis set, one must consider the orbital products ($\Psi^*\Psi$). The more functions or the more complex a basis set, the more computationally expensive the calculation becomes. The number of functions used to model a wave function, is scaled exponentially in *ab initio*

calculations. Basis sets are categorized into several different types including minimal, extended, split-valence, polarized and diffuse.

Slater-type orbitals (STO) are used to approximate atomic orbitals, however they are computationally expensive to use. They have the general form of

$$S_{nlm}^{\zeta}(r,\vartheta,\psi) = N\, r^{n-1} e^{-\zeta \mathrm{r}}\, Y_l^m(\vartheta,\psi)$$

where N is the normalization constant and Y is the spherical harmonics. STO's are very similar in functional from to hydrogen atom eigenfunctions. While STOs are capable of describing atomic orbitals well, the required integrals are often need to be solved numerically and therefore are computationally expensive.

$$G_{nlm}^{\zeta}(r,\vartheta,\psi) = N\, r^{n-1} e^{-\zeta \mathrm{r}^2}\, Y_l^m(\vartheta,\psi)$$

It is more computationally compact to calculate integrals with Gaussian basis functions. Gaussian-type orbitals, or GTOs as these are called typically, are easier mathematically to use than STOs as the squares also produce Gaussians thus two GTO on different centers can be written as a single GTO centered at a position in between the original two centers. Gaussians have the wrong behavior near the nuclear where they don't have a cusp and far from the nucleus where they taper off too soon. Therefore several Gaussian primitives are used to approximately model the STO. Gaussian primitive coefficients are typically optimized with *ab initio* (usually HF though some correlation methods have been used) atom calculations.

Figure 1.6 shows how one Gaussian would represent the 1s wave function. It is readily apparent that this method does not appropriately represent the system it is trying to model. We therefore use combinations as described above.

Figure 1.7 demonstrates how using an increasing number of Gaussians can better approximate the Slater type orbitals.

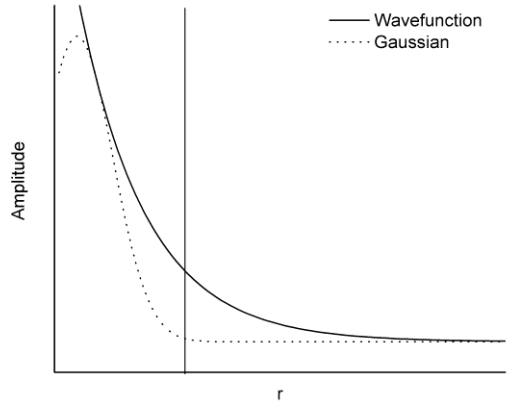


Figure 1.6 Wave function vs. 1 Gaussian

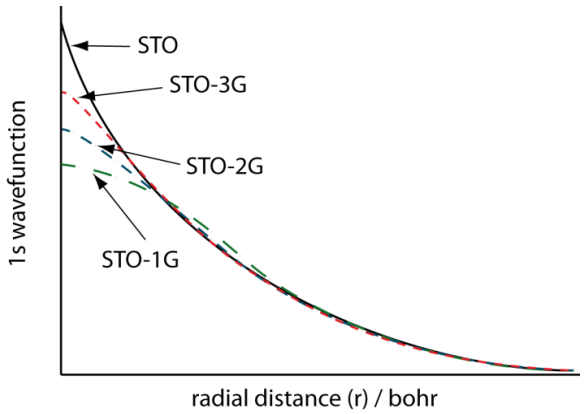


Figure 1.7 Wave Function vs. STO

Minimal

A minimum basis uses a single basis function is used for each orbital (or more appropriately STO) on an atom. The most common of these minimal basis set functions are referred to as STO-NG where N is denotes the number of Gaussians primitives used to create the orbital that will represent the STO. Minimal basis sets are the simplest expansions of orbitals possible. Minimal basis sets are usually chosen for cursory investigations or preliminary structure determination.

Extended

A single STO is often not flexible enough to properly represent the nature of the atomic orbitals it is trying to represent. In an expanded basis set, orbitals are approximated

using a linear combination of two (in the case of double zeta) STOs. Zeta is the term that varies the diffuseness of an orbital. A double zeta form is shown below. The orbital is expressed as a sum of two STOs with a constant to determine contribution of each STO

$$\phi(r) = \phi(r, \zeta_1) + d\,\phi(r, \zeta_2)$$

Triple and quadruple zeta basis sets have a similar form only including more STOs. Expanded basis sets are computationally expensive; however they do provide better approximations of the wave function.

Split Valence

Pople basis sets

Calculating full extended orbitals for every orbital on every atom can be very costly computationally. Core electrons participate to a lesser extent than valence electrons in chemistry. Split-valence basis sets were created to account for this and consequently reduce the cost of a calculation. X-YZ(WV)g is the typical form of a split-valence basis set. As the inner shell electrons play a smaller roll in any calculation, core electrons are approximated by X number of Gaussians to represent a single STO. The Y and Z (W V) term indicates the number of Gaussians used to represent each STO of the valence electrons. The number of terms included therefore implies the zeta. (i.e. 6-31G is a double zeta quality basis set) The combination of Gaussians significantly improves the accuracy of the basis set. Pople basis set is the most commonly used technique for computations as core electrons play relatively small rolls in problems of typical concern. This method models the wave function well while performing significantly faster than extended basis sets.

Basis set expansions

Most basis sets are optimized in relative to atomic calculations which have no directional characteristics. It is not feasible to optimize a basis set based on individual molecules as those optimized orbitals would not be optimal for other molecules. It therefore becomes necessary to account for the molecular environment. This is done through the addition of functions to add polarization and diffuseness. The polarization functions allow for more directional character in orbitals, while the diffuse functions allow for a further extension of the functions. These functions are added to improve the representative nature of the molecular orbitals.

A polarization function adds one node to the orbital is operating on, an example of such a case is an s orbital having some p character assigned to it. This essentially allows the orbital to be more asymmetric about the nucleus. This is important in molecular bonding as a molecule removes spherical environment. Polarization effects are usually noticed when atoms are brought close together and are important in chemical bonding and where electron correlation is important.

Diffuse basis sets take into account the tail feature of the wave functions. Gaussians typically do not represent the wave function at large radius as they taper off sooner. Diffuse functions do a better job of approximating the wave functions at greater radii. Using diffuse functions is very important when dealing with longer range effects, negative charges or with loose binding.

Polarized basis sets usually designated with a * or (d) to indicate that polarization functions are in place on all “heavy” (not hydrogen and helium) atoms. The notation ** typically indicated that in addition to d orbitals on heavy atoms p orbitals have been added to

hydrogens. Diffuse basis sets are denoted by +. The notation ++ typically refers to a diffuse shell being added to both heavy atoms and to hydrogen atoms.

Correlation Consistent

Correlation consistent (cc) basis sets are more recently created then Pople basis sets and have become widely used. The cc basis set was designed to converge systematically to the complete basis set limit. Dunning basis sets[91] (as they are sometime called for their creator) are described by the form cc-pVNZ where cc = correlation consistent, p = polarized, V= valence only, N = zeta level (D, T, Q… for double, triple and quadruple zeta respectively) cc basis sets include polarization and diffuse functions by default with additional functions available to further augment the representation of the electronic orbitals.

Solvent models

Explicit solvents

Most experiments are done in solution, so models of the experiments need a solvent to be more representative and accurate. There are two ways to approximate a solvent. An explicit solvent can be used or a continuum model. An explicit solvent can be done using full *ab initio* solvent molecules or a reduced form. *ab initio* waters are prohibitively expensive and are often not used unless they are participating in chemistry. Explicit solvent can be computed by effective fragment potential method (EFP)[92–94]. EFP creates modeled solvent molecules that interact non-covalently with solute molecules. EFP calculations often require a large quantity of solvent molecules to appropriately model a solvent typically 1 to 2 solvent

shells. This means that both solvent models are still computationally too expensive to use in conjunction with our large solute molecules.

Continuum Solvents

Two continuum models are readily available, are used frequently and are easy to use in GAMESS, self consistent reaction field (SCRF)[95,96] and polarized continuum model (PCM)[97–100] calculations. Both models approximate solvent by creating a dielectric continuum outside a bubble surrounding the solute. SCRF creates the bubble at a fixed radius about the molecule. PCM is a more computationally expensive, though more accurate calculation type that is run for solvent calculations. This method approximates a solvent shell slightly larger than the van der Waals radius around each of the atoms in a molecule. The solvent shell is a dielectric continuum using the dielectric constant of a given solvent. PCM is less susceptible to error than SCRF to error, due to SCRF's dependence on user defines solvent shell size.

Computational Resources

Throughout the process of computational chemistry the available resources are a primary concern in evaluating what calculations can and will be computed. Perspective on the available resources available can help inform the understanding of the choices that have been made.

The cluster, used for the calculations reported here, consists of 32 Dell PowerEdge M610 blade nodes in two Dell M1000e enclosures. Each node utilizes 2 Intel Xeon L5520 quad core processors, with 24GB (6x4GB) of DDR3 RAM. Each node has two 73GB 15K

rpm SCSI drives in RAID 0 configuration with the majority of space dedicated to scratch space. The individual nodes are interlinked with two network infrastructures. A GigE network is used for slow network traffic while a Mellanox quad data rate (QDR) infiniband network is used for low latency network traffic. A head node controls the cluster and manages the Rocks Cluster distribution that is the operating system of the cluster. It utilizes a Redhat Linux backbone. GAMESS is installed on each node and it utilizes the Intel compilers, MKL and MPI.

Conclusion

Boron has been shown to be useful in many surprising ways. It is often neglected in the development of computational methodologies and consequently many methods for understanding large scale systems perform poorly when they contain boron. To rectify this lack, high level computations of boron containing molecules have been done to improve our understanding of small systems so that the gleaned information can be utilized to develop better large systems studying methodologies.

In the following chapters I will discuss the behavior of boron dative bonding to nitrogen and oxygen. Chapter Two will discuss boron nitrogen dative bonding direct transfers and boron oxygen bonding intermediates. This information will be used to shed light on a proposed mechanism for physiological activities. Chapter 3 focuses on the development of a tool called CREPES and demonstrates its utility in searching the potential energy surface of saccharides. Chapter 4 describes the potential energy surface for the crosslinking of RG-II and will provide a theoretical framework for assigning the R and S configurations of the apiose crosslink. Chapter 5 will summarize the contents and describe

how the tools developed here can help study in the developing fields of borate sugar binding and quorum sensing.

References

(1) Barth, R. F.; Soloway, A. H.; Fairchild, R. G.; Brugger, R. M. *Cancer* **1992**, *70*, 2995–3007.

(2) Hawthorne, M. F. *Angewandte Chemie-International Edition in English* **1993**, *32*, 950–984.

(3) Slatkin, D. N. *Brain* **1991**, *114*, 1609–1629.

(4) Barth, R. F.; Yang, W.; Soloway, A. H. *Cancer Res.* **1997**, *57*, 1129.

(5) Barth, R.; Soloway, A.; Brugger, R. *Cancer Invest.* **1996**, *14*, 534–550.

(6) Dagrosa, M. A.; Viaggi, M.; Rebagliati, R. J.; Batistoni, D.; Kahl, S. B.; Juvenal, G. J.; Pisarev, M. A. *Mol. Pharmaceutics* **2005**, *2*, 151–156.

(7) Kabalka, G. W.; Wu, Z.; Yao, M. *Applied Organometallic Chemistry* **2008**, *22*, 516–522.

(8) Kabalka, G.; Yao, M. *Synthesis* **2003**, 2890–2893.

(9) Diaz, S.; Gonzalez, A.; de Riancho, S.; Rodriguez, A. *J. Organomet. Chem.* **2000**, *610*, 25–30.

(10) Lindstrom, P.; Naeslund, C.; Sjoberg, S. *Tetrahedron Lett.* **2000**, *41*, 751–754.

(11) Nakamura, H.; Fujiwara, M.; Yamamoto, Y. *J. Org. Chem.* **1998**, *63*, 7529–7530.

(12) Malan, C.; Morin, C. *J. Org. Chem.* **1998**, *63*, 8019–8020.

(13) Kahl, S.; Kasar, R. *J. Am. Chem. Soc.* **1996**, *118*, 1223–1224.

(14) Burnham, B. S.; Chen, S. Y.; Sood, A.; Spielvogel, B. F.; Hall, I. H. *Met.-Based Drugs* **1995**, *2*, 221.

(15) Hall, I. H.; Burnham, B. S.; Rajendran, K. G.; Chen, S. Y.; Sood, A.; Spielvogel, B. F.; Shaw, B. R. *Biomed. Pharmacother.* **1993**, *47*, 79.

(16) Hall, I. H.; Spielvogel, B. F.; Griffin, T. S.; Docks, E. L.; Brotherton, R. J. *Research Communications in Chemical Pathology and Pharmacology* **1989**, *65*, 297–317.

(17) Hall, I. H.; Das, M. K.; Harchelroad, F.; Jr, null; McPhail, A. T.; Spielvogel, B. F. *J. Pharm. Sci.* **1981**, *70*, 339.

(18) Sood, A.; Sood, C. K.; Spielvogel, B. F.; Hall, I. H.; Wong, O. T.; Mittakanti, M.; Morse, K. *Archiv Der Pharmazie* **1991**, *324*, 423–432.

(19) Eisen, E. J.; Jones, E. E.; Rajendran, K. G.; Hall, I. H. *Growth Dev. Aging* **1996**, *60*, 7.

(20) Hall, I. H.; Wong, O. T.; Sood, A.; Sood, C. K.; Spielvogel, B. F.; Shrewsbury, R. P.; Morse, K. W. *Pharmacol. Res.* **1992**, *25*, 259.

(21) Das, M. K.; Maiti, P. K.; Roy, S.; Mittakanti, M.; Morse, K. W.; Hall, I. H. *Arch. Pharm. (Weinheim, Ger.)* **1992**, *325*, 267.

(22) Hall, I. H.; Chen, S. Y.; Rajendran, K. G.; Sood, A.; Spielvogel, B. F.; Shih, J. *Environ. Health Perspect.* **1994**, *102*, 21.

(23) Sood, A.; Spielvogel, B. F.; Shaw, B. R.; Carlton, L. D.; Burnham, B. S.; Hall, E. S.; Hall, I. H. *Anticancer Research* **1992**, *12*, 335–343.

(24) Sood, C. K.; Sood, A.; Spielvogel, B. F.; Yousef, J. A.; Burnham, B.; Hall, I. H. *J. Pharm. Sci.* **1991**, *80*, 1133.

(25) Hall, I. H.; Burnham, B. S.; Elkins, A.; Sood, A.; Powell, W.; Tomasz, J.; Spielvogel, B. F. *Met.-Based Drugs* **1996**, *3*, 155.

(26) Hall, I. H.; Burnham, B. S.; Chen, S. Y.; Sood, A.; Spielvogel, B. F.; Morse, K. M. *Met.-Based Drugs* **1995**, *2*, 1.

(27) Hall, I. H.; Rajendran, K. G.; Chen, S. Y.; Wong, O. T.; Sood, A.; Spielvogel, B. F. *Archiv Der Pharmazie* **1995**, *328*, 39–44.

(28) Hall, I. H.; Starnes, C. O.; Mcphail, A. T.; Wisianneilson, P.; Das, M. K.; Harchelroad, F.; Spielvogel, B. F. *Journal of Pharmaceutical Sciences* **1980**, *69*, 1025–1029.

(29) Rajendran, K. G.; Burnham, B. S.; Chen, S. Y.; Sood, A.; Spielvogel, B. F.; Shaw, B. R.; Hall, I. H. *J. Pharm. Sci.* **1994**, *83*, 1391.

(30) Sood, A.; Sood, C. K.; Spielvogel, B. F.; Hall, I. H.; Wong, O. T. *J. Pharm. Sci.* **1992**, *81*, 458.

(31) Rajendran, K. G.; Chen, S. Y.; Sood, A.; Spielvogel, B. F.; Hall, I. H. *Biomed. Pharmacother.* **1995**, *49*, 131.

(32) Fisher, L. A Computational Investigation of Boron Containing Complexes and their Putative Physiological Activity, UW-Milwaukee: Milwaukee, 1999.

(33) Miyamoto, C. M.; Lin, Y. H.; Meighen, E. A. *Molecular Microbiology* **2000**, *36*, 594–607.

(34) Joly-Guillou, M. L. *Presse Med* **1998**, *27 Suppl 5*, 31–33.

(35) Lyon, G. J.; Muir, T. W. *Chemistry & Biology* **2003**, *10*, 1007–1021.

(36) Dunny, G. M.; Brown, B. L.; Clewell, D. B. *Proceedings of the National Academy of Sciences* **1978**, *75*, 3479 –3483.

(37) Veening, J.; Hamoen, L. W.; Kuipers, O. P. *Molecular Microbiology* **2005**, *56*, 1481–1494.

(38) Lindum, P. W.; Anthoni, U.; Christophersen, C.; Eberl, L.; Molin, S.; Givskov, M. *Journal of Bacteriology* **1998**, *180*, 6384 –6388.

(39) Chen, X.; Schauder, S.; Potier, N.; van Dorsselaer, A.; Pelczer, I.; Bassler, B. L.; Hughson, F. M. *Nature* **2002**, *415*, 545.

(40) Diggle, S. P.; Gardner, A.; West, S. A.; Griffin, A. S. *Philos. Trans. R. Soc. B-Biol. Sci.* **2007**, *362*, 1241–1249.

(41) Ni, N. T.; Choudhary, G.; Peng, H. J.; Li, M. Y.; Chou, H. T.; Lu, C. D.; Gilbert, E. S.; Wang, B. H. *Chemical Biology & Drug Design* **2009**, *74*, 51–56.

(42) Jin, S.; Cheng, Y.; Reid, S.; Li, M.; Wang, B. *Med. Res. Rev.* **2010**, *30*, 171–257.

(43) Steiner, M.-S.; Duerkop, A.; Wolfbeis, O. S. *Chem. Soc. Rev.* **2011**, *40*, 4805–4839.

(44) Fang, H.; Kaur, G.; Wang, B. H. *Journal of Fluorescence* **2004**, *14*, 481–489.

(45) DI, L.; WANG, C.; WU, J.; WAN, L.-S.; XU, Z.-K. *Chinese Journal of Analytical Chemistry* **2011**, *39*, 592–598.

(46) Ellis, G. A.; Palte, M. J.; Raines, R. T. *J. Am. Chem. Soc.* **2012**.

(47) Wittstein, A.; Apioger, F. *Ann. Chem. Pharm.* **1857**, *103*, 362–364.

(48) Jay, H. *Compt. Rend. Acad. Sci.* **1895**, *121*, 896.899.

(49) Sommer, A.; Lipman, C. *Plant Physiol.* **1926**, *1*, 231–249.

(50) Mengel, K.; Kirkby, E. *Principles of Plant Nutrition*; 5th ed.; Kluwer Acedemic Publishers, 2001.

(51) Hu, H.; Brown, P. *Plant Soil* **1997**, *193*, 49–58.

(52) *Albion Plant Nutrition* **2003**.

(53) Foroughi, M.; Marschner, H.; Döting, H. -W *Zeitschrift für Pflanzenernährung und Bodenkunde* **1973**, *136*, 220–228.

(54) O'Neill, M. A. *Methods in plant biochemistry*; Academic Press: London ; San Diego, 1990; Vol. 2.

(55) Ridley, B.; O'Neill, M.; Mohnen, D. *Phytochemistry* **2001**, *57*, 929–967.

(56) ONeill, M.; Warrenfeltz, D.; Kates, K.; Pellerin, P.; Doco, T.; Darvill, A.; Albersheim, P. *J. Biol. Chem.* **1996**, *271*, 22923–22930.

(57) CARPITA, N.; GIBEAUT, D. *Plant J.* **1993**, *3*, 1–30.

(58) MCCANN, M.; ROBERTS, K. *J. Exp. Bot.* **1994**, *45*, 1683–1691.

(59) O'Neill, M. A.; Ishii, T.; Albersheim, P.; Darvill, A. G. *Annual Review of Plant Biology* **2004**, *55*, 109–139.

(60) WHITCOMBE, A.; ONEILL, M.; STEFFAN, W.; ALBERSHEIM, P.; DARVILL, A. *Carbohydr. Res.* **1995**, *271*, 15–29.

(61) MELTON, L.; MCNEIL, M.; DARVILL, A.; ALBERSHEIM, P.; DELL, A. *Carbohydr. Res.* **1986**, *146*, 279–305.

(62) SPELLMAN, M.; MCNEIL, M.; DARVILL, A.; ALBERSHEIM, P.; DELL, A. *Carbohydr. Res.* **1983**, *122*, 131–153.

(63) PUVANESARAJAH, V.; DARVILL, A.; ALBERSHEIM, P. *Carbohydr. Res.* **1991**, *218*, 211–222.

(64) Vidal, S.; Doco, T.; Williams, P.; Pellerin, P.; York, W.; O'Neill, M.; Glushka, J.; Darvill, A.; Albersheim, P. *Carbohydr. Res.* **2000**, *326*, 277–294.

(65) Glushka, J.; Terrell, M.; York, W.; O'Neill, M.; Gucwa, A.; Darvill, A.; Albersheim, P.; Prestegard, J. *Carbohydr. Res.* **2003**, *338*, 341–352.

(66) STEVENSON, T.; DARVILL, A.; ALBERSHEIM, P. *Carbohydr. Res.* **1988**, *182*, 207–226.

(67) Reuhs, B. L.; Glenn, J.; Stephens, S. B.; Kim, J. S.; Christie, D. B.; Glushka, J. G.; Zablackis, E.; Albersheim, P.; Darvill, A. G.; O'Neill, M. A. *Planta* **2004**, *219*, 147–157.

(68) STEVENSON, T.; DARVILL, A.; ALBERSHEIM, P. *Carbohydr. Res.* **1988**, *179*, 269–288.

(69) YORK, W.; DARVILL, A.; MCNEIL, M.; ALBERSHEIM, P. *Carbohydr. Res.* **1985**, *138*, 109–126.

(70) Rodriguez-Carvajal, M.; du Penhoat, C.; Mazeau, K.; Doco, T.; Perez, S. *Carbohydr. Res.* **2003**, *338*, 651–671.

(71) THOMAS, J.; DARVILL, A.; ALBERSHEIM, P. *Carbohydr. Res.* **1989**, *185*, 261–277.

(72) Perez, S.; Rodriguez-Carvajal, M.; Doco, T. *Biochimie* **2003**, *85*, 109–121.

(73) MAZUREK, M.; PERLIN, A. *Can. J. Chem.-Rev. Can. Chim.* **1963**, *41*, 2403–&.

(74) Ishii, T.; Ono, H. *Carbohydr. Res.* **1999**, *321*, 257–260.

(75) Ishii, T.; Matsunaga, T.; Pellerin, P.; O'Neill, M.; Darvill, A.; Albersheim, P. *J. Biol. Chem.* **1999**, *274*, 13098–13104.

(76) Pellerin, P.; O'Neill, M. *Analusis* **1998**, *26*, M32–M36.

(77) Kobayashi, M.; Nakagawa, H.; Asaka, T.; Matoh, T. *Plant Physiol.* **1999**, *119*, 199.

(78) Levine, I. N. *Quantum Chemistry*; 6th ed.; Prentice Hall, 2008.

(79) *Encyclopedia of computational chemistry 1, A - D.*; Wiley: Chichester; Weinheim, 1998.

(80) Density functional theory http://www.lmgtfy.com/?q=density+functional+theory (accessed Apr 13, 2012).

(81) LeTourneau, H. A.; Birsch, R. E.; Korbeck, G.; Radkiewicz-Poutsma, J. L. *Journal of Physical Chemistry A* **2005**, *109*, 12014–12019.

(82) Gilbert, T. *J. Phys. Chem. A* **2004**, *108*, 2550–2554.

(83) HOLME, T.; TRUONG, T. *Chem. Phys. Lett.* **1993**, *215*, 53–57.

(84) Szabo, A.; Ostlund, N. S. *Modern quantum chemistry.*; Dover Publications: New York, 1989.

(85) GORDON, M.; Schmidt, M. W. General Atomic and Molecular Electronic Structure System: GAMESS User's Guide http://www.msg.ameslab.gov/gamess/documentation.html (accessed Apr 12, 2012).

(86) Moller, C.; Plesset, M. *Phys. Rev.* **1934**, *46*, 0618–0622.

(87) BARTLETT, R. *Annu. Rev. Phys. Chem.* **1981**, *32*, 359–401.

(88) ALBERTS, I.; HANDY, N. *J. Chem. Phys.* **1988**, *89*, 2107–2115.

(89) HANDY, N.; KNOWLES, P.; SOMASUNDRAM, K. *Theor. Chim. Acta* **1985**, *68*, 87–100.

(90) CIZEK, J. *J. Chem. Phys.* **1966**, *45*, 4256–&.

(91) Dunning, T. H. *Journal of Chemical Physics* **1971**, *55*, 716–&.

(92) JENSEN, J.; DAY, P.; GORDON, M.; BASCH, H.; COHEN, D.; GARMER, D.; KRAUS, M.; STEVENS, W. In *Modeling the Hydrogen Bond*; Smith, D., Ed.; Amer Chemical Soc: Washington, 1994; Vol. 569, pp. 139–151.

(93) Day, P.; Jensen, J.; Gordon, M.; Webb, S.; Stevens, W.; Krauss, M.; Garmer, D.; Basch, H.; Cohen, D. *J. Chem. Phys.* **1996**, *105*, 1968–1986.

(94) Gordon, M. S.; Slipchenko, L.; Li, H.; Jensen, J. H. In *Annual Reports in Computational Chemistry*; Elsevier, 2007; Vol. Volume 3, pp. 177–193.

(95) Kirkwood, J. G. *J. Chem. Phys.* **1934**, *2*.

(96) KARELSON, M.; TAMM, T.; ZERNER, M. *J. Phys. Chem.* **1993**, *97*, 11901–11907.

(97) Li, H.; Pomelli, C.; Jensen, J. *Theor. Chem. Acc.* **2003**, *109*, 71–84.

(98) Li, H.; Jensen, J. *J. Comput. Chem.* **2004**, *25*, 1449–1462.

(99) Li, H. *J. Chem. Phys.* **2009**, *131*.

(100) Wang, Y.; Li, H. *J. Chem. Phys.* **2009**, *131*.

CHAPTER 2. THORETICAL STUDY OF COMPETATIVE LEWIS ACID/BASE CHEMISTRY VIA BORANE TRANSFER REACTIONS

Under revision for Journal of Physcial Chemistry

Abstract

Ab initio calculations have been carried out on a series of boron-nitrogen dative bond containing compounds with intriguing physiological activity. They describe the pathway for the transfer of a borane moiety between Lewis base sites. The calculation of bond dissociation energies, barrier heights, reaction energies, and free energies were used to evaluate the proposed mechanism. Transition state energies for the S_N2 reactions studied were lower than bond dissociation energies or water assisted dissociation. Reaction energies suggest that the transfer of boranes from simple amines to more physiologically relevant Lewis base sites is thermodynamically favored. Inclusion of solvation effects has an important effect on both the predicted thermodynamic and kinetic variables.

Introduction

Boron has been studied for its physiological activity for many years. Much of this research has been on boron neutron capture therapy (BNCT)[1-4] as a treatment for cancer. Many small boron-containing molecules, used as potential boron delivery mechanism for BNCT, have been found to exhibit surprising physiological activity. These boron-containing molecules include boron-nitrogen dative bonds, dipolar bonds with bond dissociation energies around 200 kJ*mol^{-1}. Some of the interesting activities of these molecules include anti-hyperlipidemic[5-10], anti-neoplastic[11-15], anti-obesity[13], anti-inflammatory[16-21], and anti-osteoporotic[22] effects. Despite an abundance of research on the physiological effects, there

have been relatively few experimental attempts to elucidate the mechanism by which these physiological activities occur.

The broad range of activities of small boron-containing molecules may be explained by the inhibition of enzymes or critical physiological pathways by the transfer of the electron deficient borane moieties to nitrogen Lewis base active sites in the body.[23] This hypothesized reaction is a competitive Lewis acid/base transfer reaction. Correlations between boron-nitrogen bond strength and the anti-neoplastic and the anti-hyperlipidemic activities has been found[24]. The present work shows evidence that Lewis base sites modeled after common active site residues create more stable B-N dative bonds than with other small amines. This evidence supports the theory that a transfer of a borane moiety to biologically relevant nitrogens can occur and may result in the described physiological activity. Among the molecules observed to have physiological activities[2-4], the structural complexity of the small molecules did not produce a trend in activity. It is important to note that clear trends in physiological activity based on structural complexity would not be expected from our hypothesis of borane transfer between Lewis base sites. The expected trend would be based on competitive Lewis acid base chemistry. The strength of the binding of the borane to the small nitrogen containing moiety as compared to the strength of the binding to the physiological Lewis base site would show the expected trend.

The exchange of boron-containing moieties from the small nitrogen containing molecule to the biologically relevant nitrogen-containing molecules can occur via an S_N1 or S_N2 mechanism. A previous study has found that the S_N1 pathway is almost equivalent in energy to complete dissociation[23]. The S_N2 mechanism, however, has a significantly lower energy barrier than both S_N1 (partial dissociation) and a two-step pathway through adduct

bond breaking and separate formation of a new adduct bond (complete dissociation). Data indicate that the most probable mechanism for the borane moiety exchange is the S_N2 mechanism. The work reported herein improves S_N2 mechanism calculations[23] and demonstrates that water assisted dissociation is not energetically competitive with the S_N2 mechanism.

The next two sections will describe the calculations and calculated molecules. The discussion section will describe the results of a study of borane moiety exchange between nitrogen containing molecules. The selected molecules have shown physiological activity. The borane moiety exchange is envisioned as an S_N2 mechanistic Lewis acid/base transfer; the proposed mechanistic pathway is hypothesized to be an active component of observed enzymatic inhibition and physiological pathway termination. Histidine was chosen as the prototypical biological nitrogen containing Lewis base. Calculations confirm that borane moiety transfer to histidine is thermodynamically favorable. The favorability of the S_N2 mechanistic pathway versus bond dissociation or water assisted bond dissociation is confirmed here. Finally, Gibbs free energy values and resultant equilibrium constants show significant transfer of boron moieties to nitrogen Lewis base sites in enzyme residues. Such a transfer may give rise to the activities that have been observed.

Methods

All structures were optimized at several levels of theory: initially, MP2/6-31G(d)[25-28] and RHF/6-31G(d) PCM[26-30]; eventually, Dunning basis sets MP2/cc-pVTZ[25,31] and, RHF/cc-pVTZ PCM.[29-31]; finally, MP2/cc-pVTZ PCM level calculations were carried out. MP2/cc-pVTZ and MP2/cc-pVTZ PCM results are presented here. Beginning and end point

stationary structures were determined to be minima by positive definite Hessian matrices. Saddle point calculations for transition states were classified by having only one negative eigenvalue of the Hessian matrix.

Selected torsional searches were done at the MP2/6-31G(d) level of theory to identify the lowest energy conformations. The geometries from the MP2/6-31G(d) level optimized structures were used as starting geometries in progressive optimizations. Solvent model calculations were done to mimic the biological reactions environment.

A polarized continuum model (PCM) with water as the solvent was used. PCM does not contain any explicit water; however, for water assisted transfer of boron moieties an ab initio water at key locations in the continuum solvent has been added. Calculations were conducted using GAMESS[32,33].

Bond dissociation energies (BDE), reaction energies and barrier heights were calculated by absolute total energy differences. Thermodynamic values were derived at 298.15 K from Hessian matrices of the optimized structures. MP2/cc-pVTZ enthalpy and free energy, values were scaled by frequency scaling factors of 0.9832[34]. Equilibrium data was computed from free energy using $\Delta G = -RT \ln K_{eq}$.

Results

The 49 structures utilized by this study are identified in Figure 2.2 - Figure 2.4, Figure 2.6 - Figure 2.11, Figure 2.13 - Figure 2.15. For convenience, the structures are broken down into categories. General structures are provided in Figure 2.1, Figure 2.5, and Figure 2.12 Reactants are defined as the borane bonded to ammonia with a free fragment of

the modeled amine Lewis base site and water. Borane bound to a modeled Lewis base site with a free ammonia molecule and water is defined as the products.

Starting Point Structures

The first category of structures includes starting geometries (Figure 2.2, (1-3)); these compounds represent the most computationally tractable set of compounds whose pharmacological activities have been studied experimentally. The general structure is described in Figure 2.1.

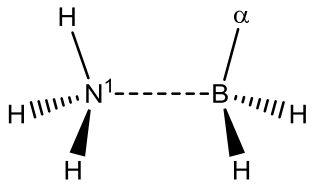
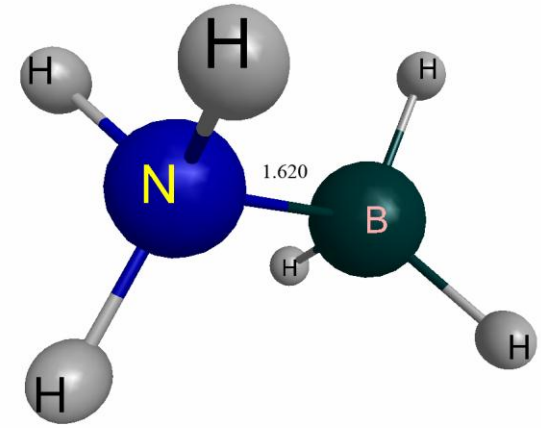


Figure 2.1 Starting Geometry

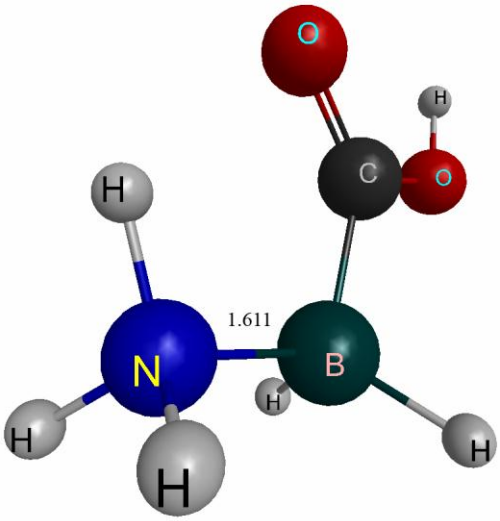
α

1) H
2) CN
3) COOH

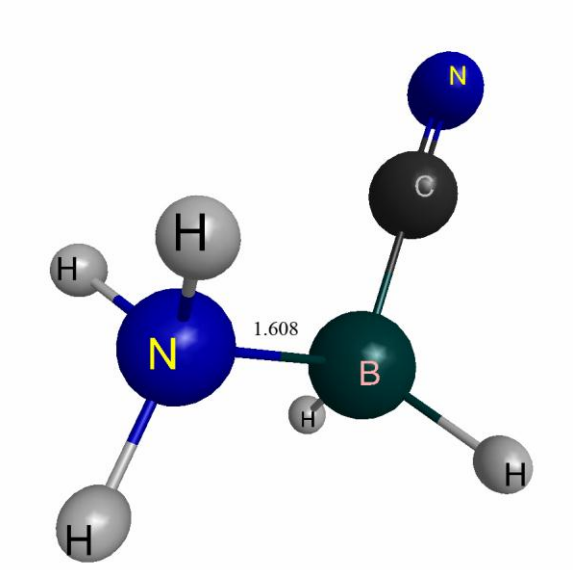
Starting geometries of the putative pharmacophores are in the staggered configuration for ammonia borane (1), and cyano ammonia borane (2), in carboxylic ammonia borane (3) the ammonia is rotated to the eclipsed position due to a non covalent inter nuclear interaction between the hydrogen on the ammonia and the carbonyl portion of the carboxylic acid



1. NH_3-BH_3



3. NH_3- BH_2COOH

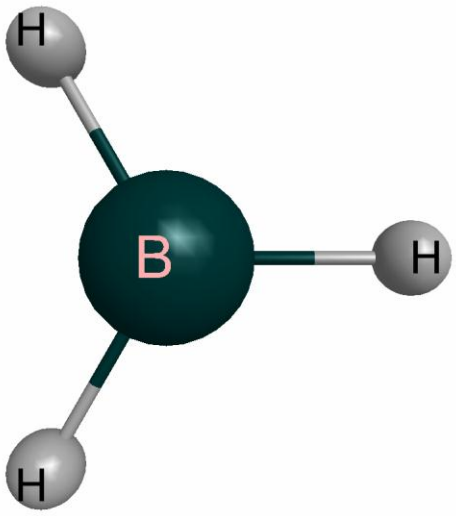


2. NH_3-BH_2CN

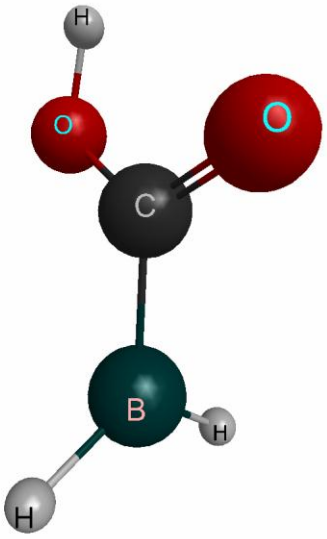
Figure 2.2 Starting point structures

Fragment Structures

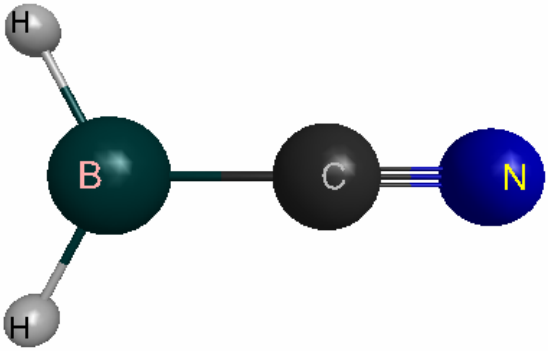
Molecular fragments are the second category of structures (Figure 2.3(4-10)). They are derived from dissociation of other molecules in the study and are required for calculating barrier heights as well as bond dissociation energies. Boranes 4 and 5 are planar while the carboxy borane is staggered. In order to have a physiologically relevant model for the histidine residue (Figure 2.3(10)), the calculations reported here had to exclude the hydrogen-bonding interactions between ring nitrogens and the N-H bond of the amine group that would be part of the peptide backbone. This interaction would not be available when histidine is a residue in an actual biological system. Backbone amines are be involved in peptide bonds and consequently hydrogen-bonding with the imidazole portion of the histidine cannot occur. The lowest energy configuration with the imidazole rotated away from the backbone was chosen as the minimum energy structure. The histidine that was calculated is the zwitterion.



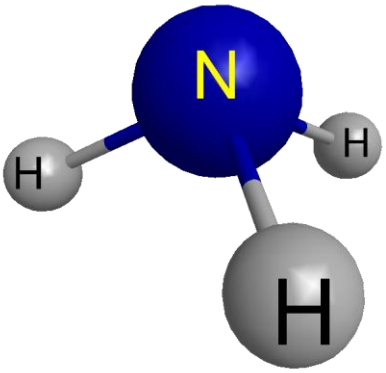
4. BH_3



6. BH_2COOH

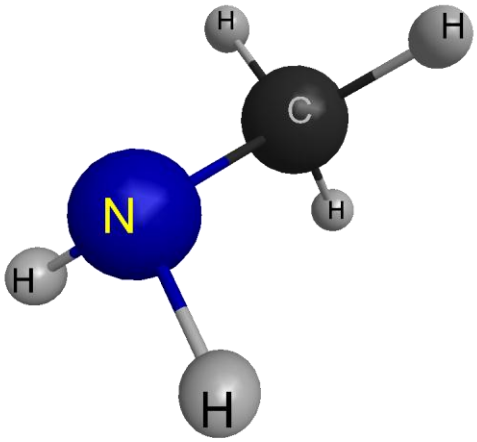


5. BH_2CN

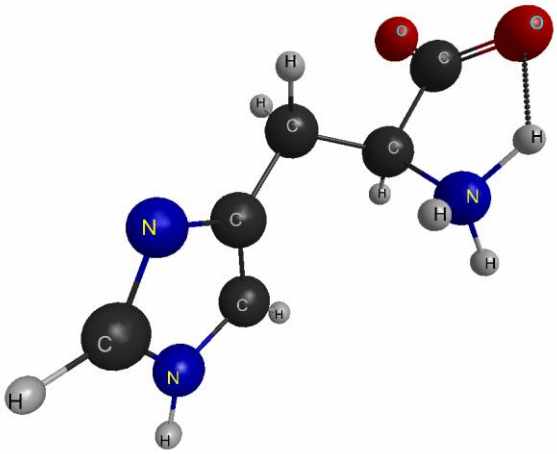


7. NH_3

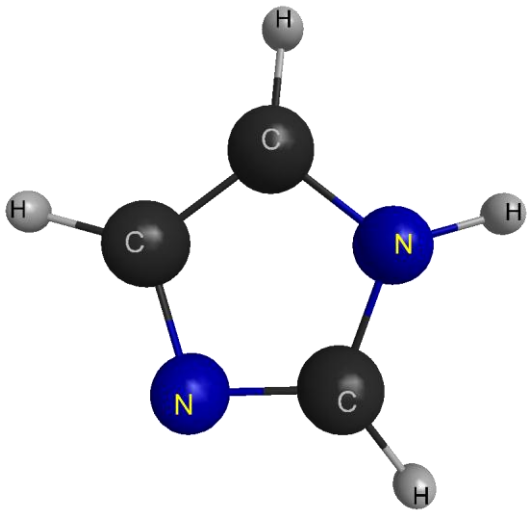
Figure 2.3 Fragments



8. NH_2CH_3



10. Histidine

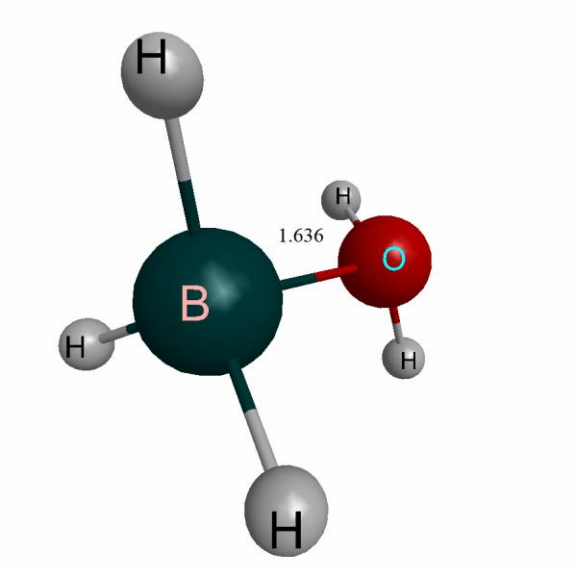


9. Imidazole

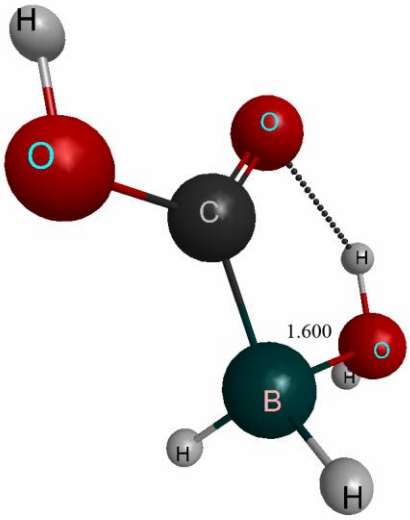
Figure 2.3 (Continued) Fragments

Water Intermediate Structures

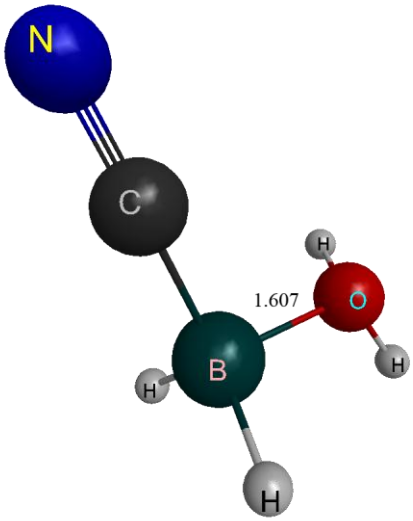
Figure 2.4 (11-13) shows the intermediates for water assisted transition; they are the third category of structures. In a water assisted mechanism the Lewis acid transfers to a water Lewis base pair to form an intermediate. This water bound intermediate then transfers the borane moiety to the Lewis base site in the body. Shown are the low energy rotamers of each structure. As with the ammonia starting point a non covalent inter nuclear interaction occurs between the carbonyl portion of the carboxylic acid and the hydrogen of the bound Lewis base.



11. BH_3-H_2O



13. BH_2COOH-H_2O



12. BH_2CN-H_2O

Figure 2.4 Water Intermediate structures

Transition State Structures

The fourth category consists of transition states. Figure 2.6 - Figure 2.8 shows the ammonia base transitions (14-25); Figure 2.10 - Figure 2.11 displays the water configurations (26-37). These structures illustrate the S_N2 transitions from putative pharmacophore to the model protein residue Lewis base site. Figure 2.5 diagrams the general structures of transitions states.

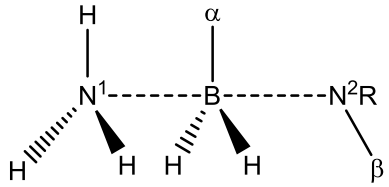


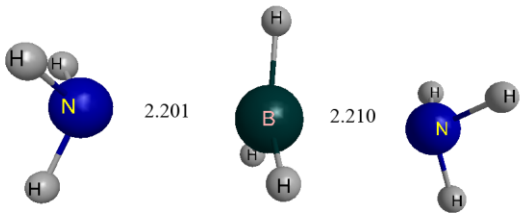
Figure 2.5 Transition State

There are 2 sets of transition states modeled, ammonia and water (ammonia pictured)

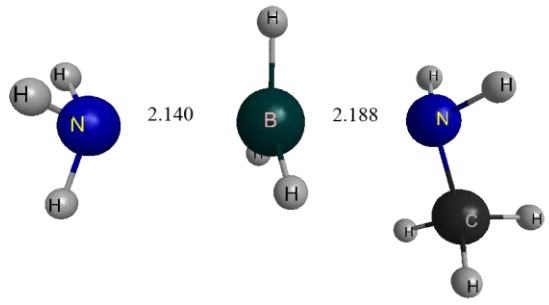
α	β	
1) H	1) H	R=2 H
2) CN	2) CH_3	R=2 H
3) COOH	3) Imidazole	R=0
	4) Histidine	R=0

In the β 3 and 4 cases the N is part of the imidazole ring, in these cases β is the carbon adjacent to the other nitrogen in the imidazole ring.

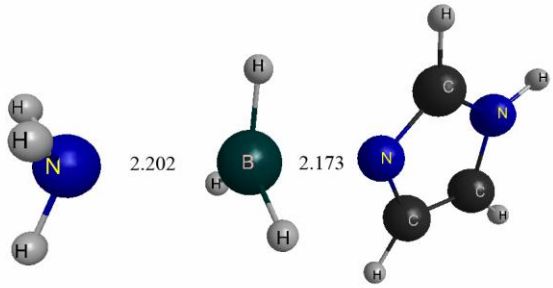
The transition states for borane, cyano borane and carboxy borane are shown in Figure 2.6 (14-17), Figure 2.7 (18-21), and Figure 2.8 (22-25) respectively. Transition states typically have staggered configurations except where a strong noncovalent interaction occurs between adjacent groups attached to the boron and nitrogen. This is particularly common with carbonyl groups of the carboxylic acid borane.



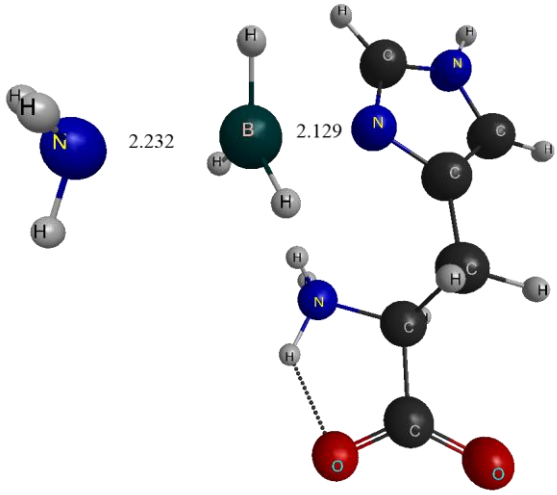
14. NH_3-BH_3-NH_3



15. NH_3-BH_3-$NH2CH_3$

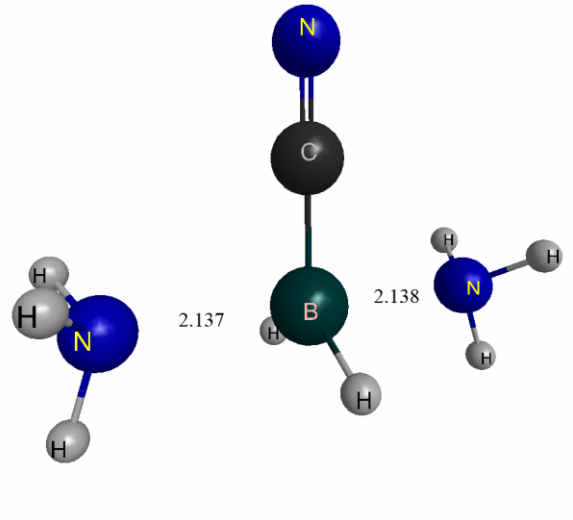


16. NH_3-BH_3-Imidazole

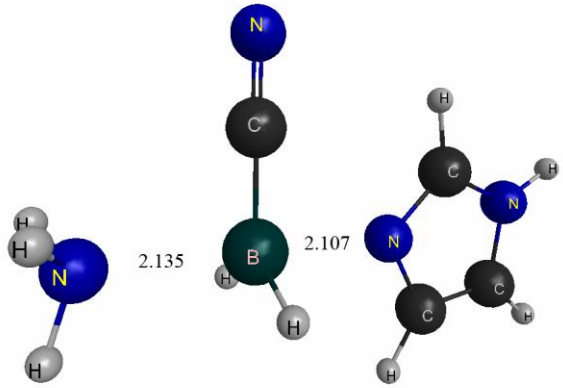


17. NH_3-BH_3-Histidine

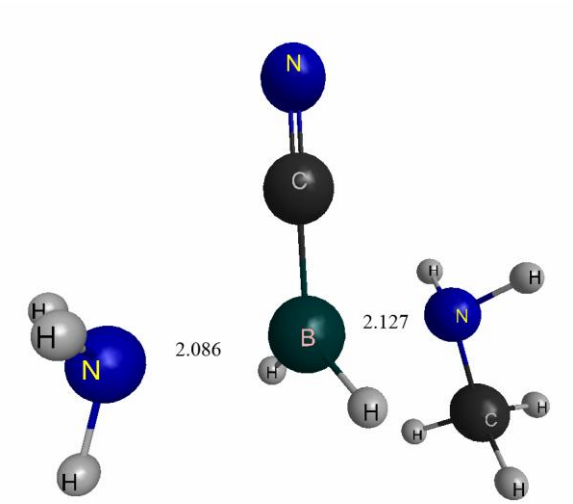
Figure 2.6 Transition states structures (ammonia), Boranes (14-17)



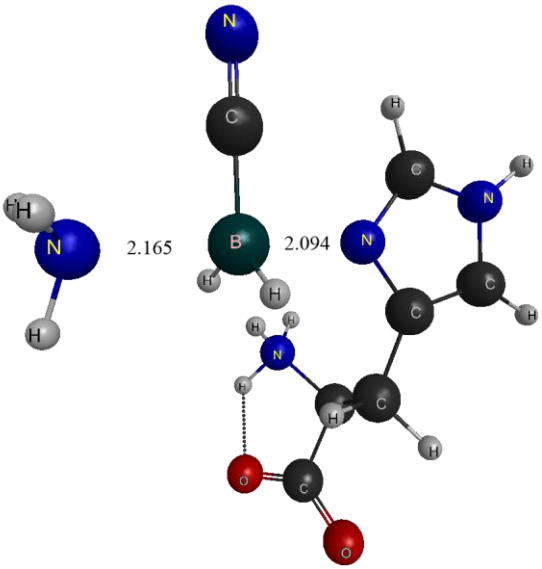
18. NH_3-BH_2CN-NH_3



20. NH_3-BH_2CN-Imidazole

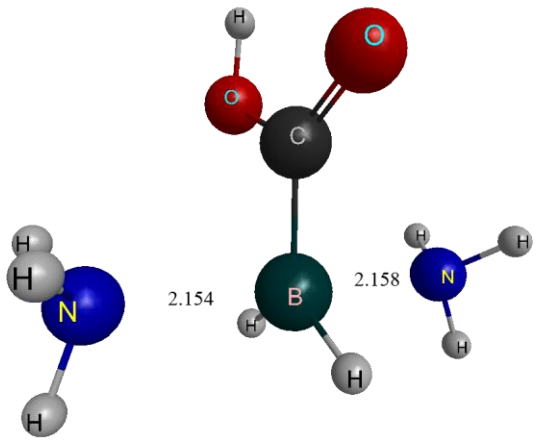


19. NH_3-BH_2CN-NH_2CH_3

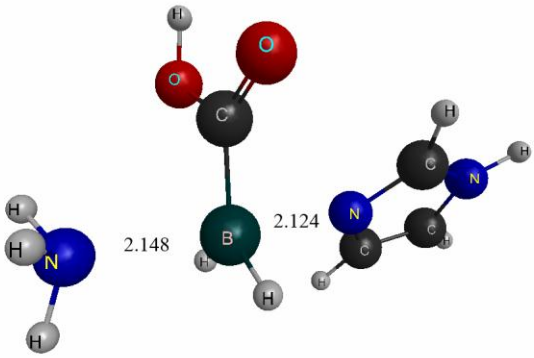


21. NH_3-BH_2CN-Histidine

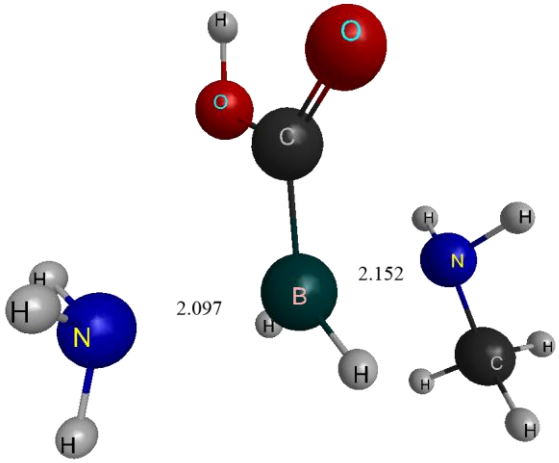
Figure 2.7 Transition state structures (ammonia), Cyanoboranes (18-21)



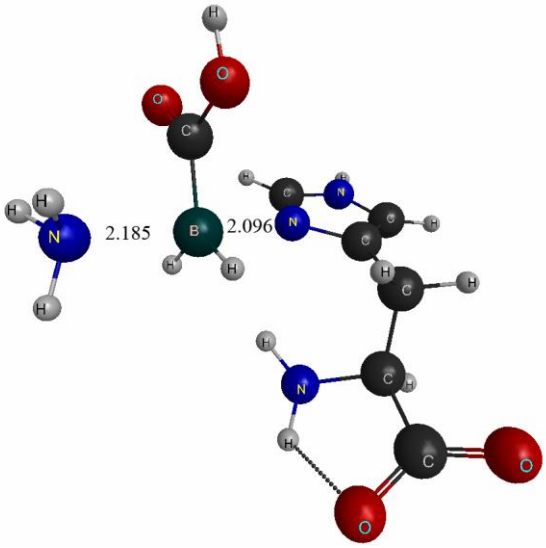
22. NH_3-BH_2COOH-NH_3



24. NH_3-BH_2COOH-Imidazole

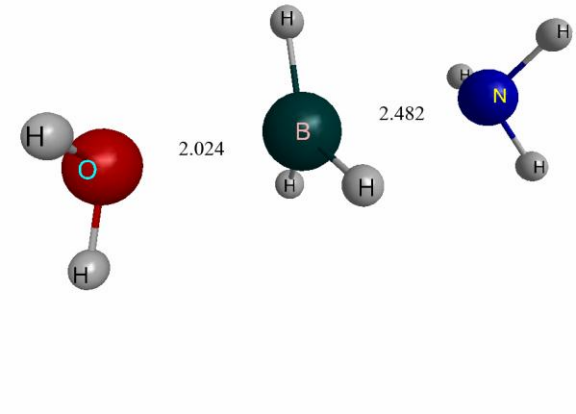


23. NH_3-BH_2COOH-NH_2CH_3

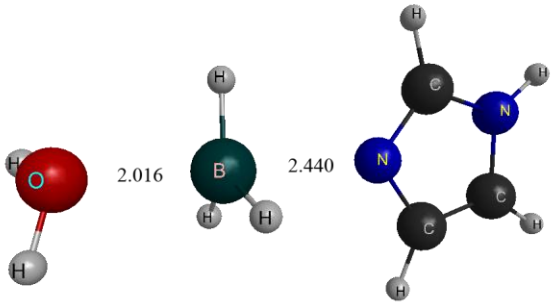


25. NH_3-BH_2COOH-Histidine

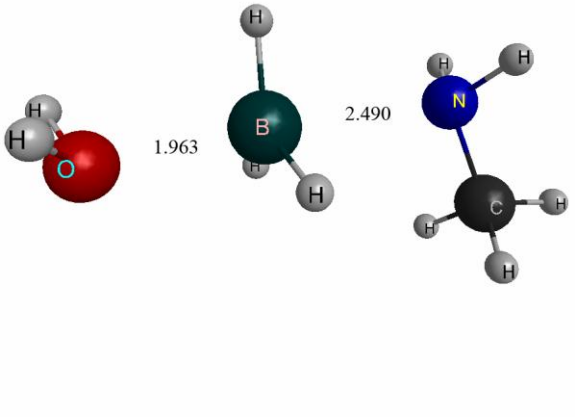
Figure 2.8 Transition state structures (ammonia), Carboxylicboranes (22-25)



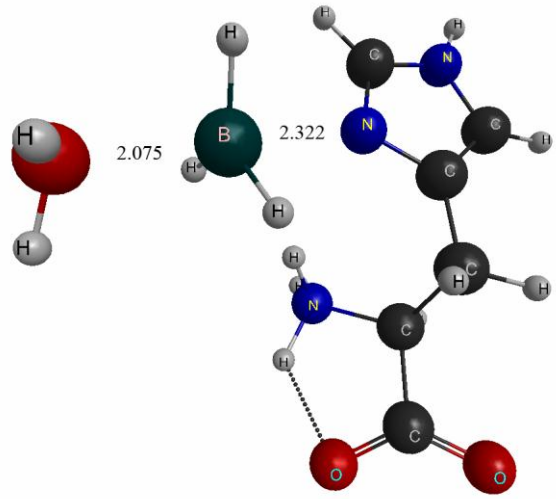
26. H_2O-BH_3-NH_3



28. H_2O-BH_3-Imidazole

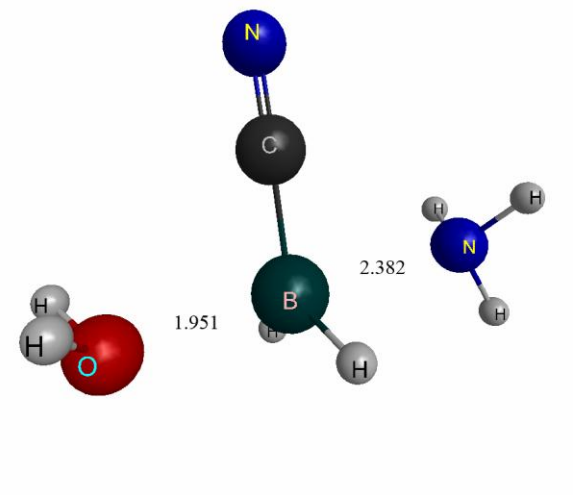


27. H_2O-BH_3-NH_2CH_3

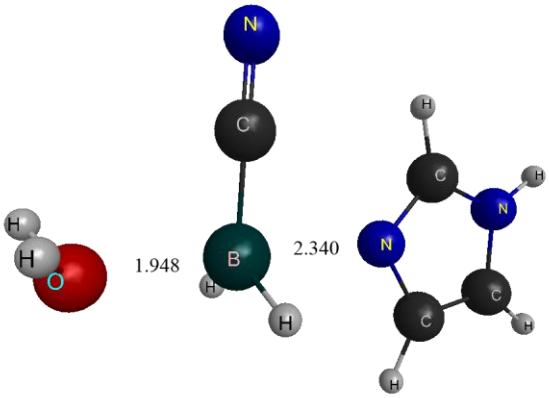


29. H_2O-BH_3-Histidine

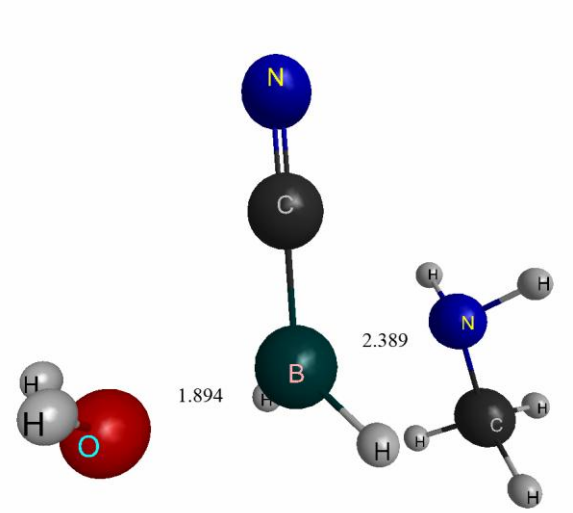
Figure 2.9 Transition State structures (Water), Borane (26-29)



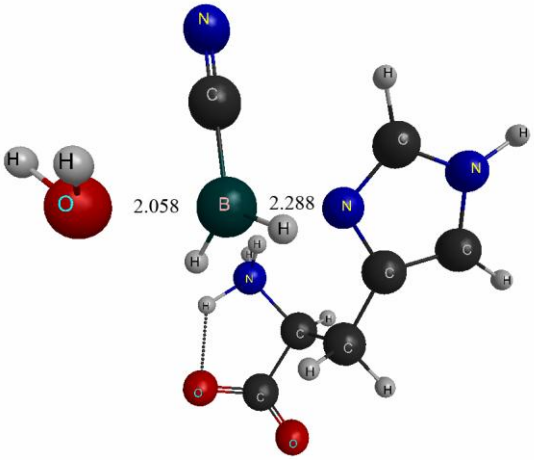
30. H_2O-BH_2CN-NH_3



32. H_2O-BH_2CN-Imidazole

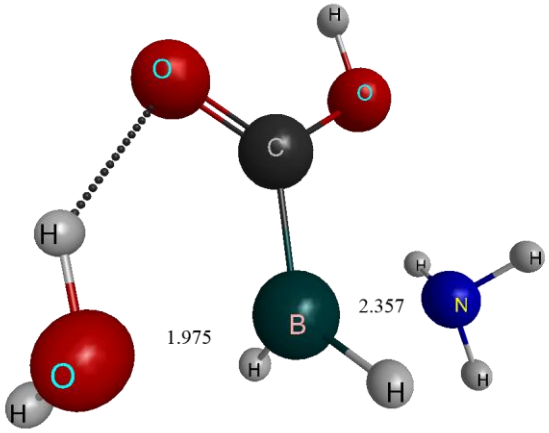


31. H_2O-BH_2CN- NH_2CH_3

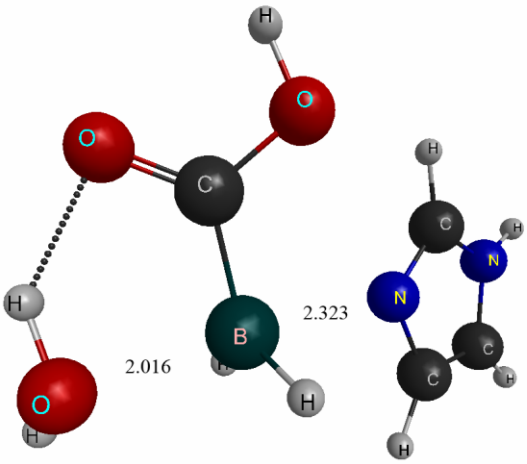


33. H_2O-BH_2CN-Histidine

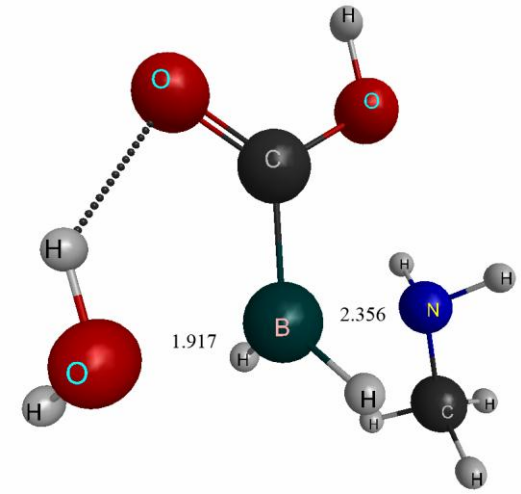
Figure 2.10 Transition State structures (Water), Cyanoboranes (30-33)



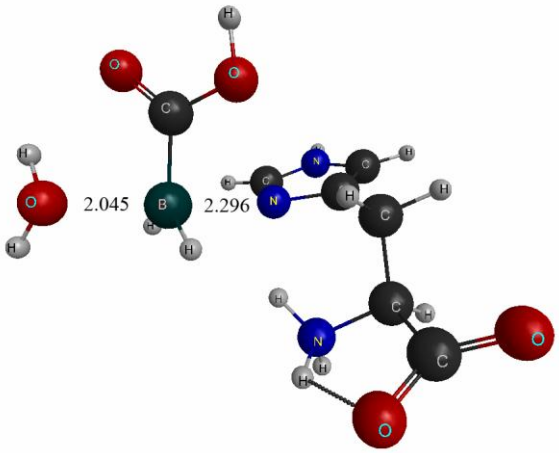
34. H_2O-BH_2COOH-NH_3



36. H_2O-BH_2COOH-Imidazole



35. H_2O-BH_2COOH-NH_2CH_3



37. H_2O-BH_2COOH-Histidine

Figure 2.11 Transition State structures (Water), Carboxylicboranes (34-37)

Endpoint Structures

Endpoints are the final category, illustrated in Figure 2.13 - Figure 2.15 (38-46). Structures from Figure 2.2 (1-3) also represent final endpoint structures as ammonia is also modeled as a Lewis base target site. These molecules represent the final structure after a boron transfer. The general structure of these molecules is described in Figure 2.12.

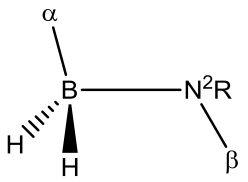
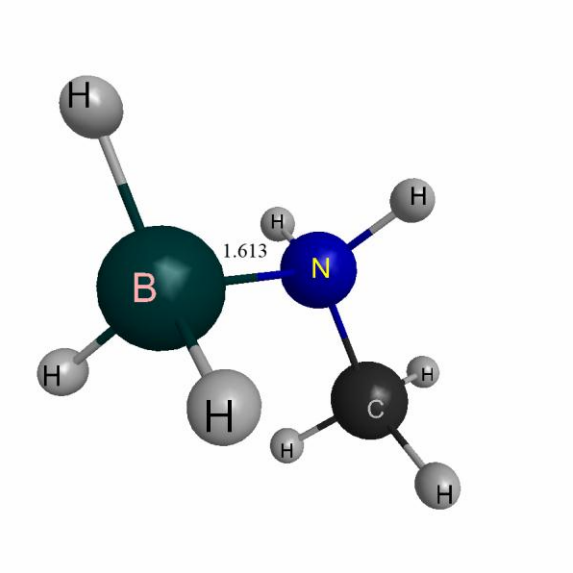


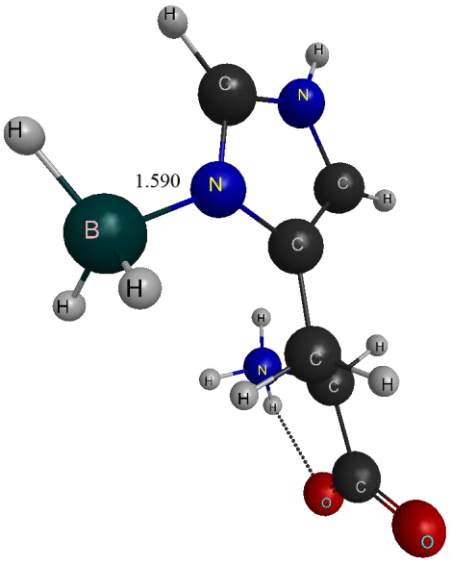
Figure 2.12 End Point

	α		β	
1)	H	1)	H	R=2
2)	CN	2)	CH_3	R=2
3)	COOH	3)	Imidazole	R=0
		4)	Histidine	R=0

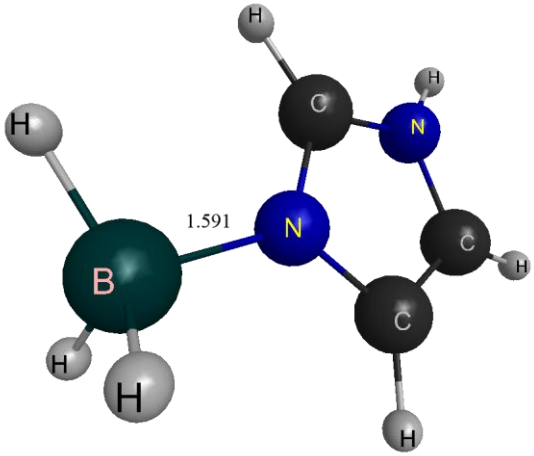
In the β 3 and 4 cases the N is part of the imidazole ring, in these cases β is the carbon adjacent to the other nitrogen in the imidazole ring.



38. BH_3-NH_2CH_3

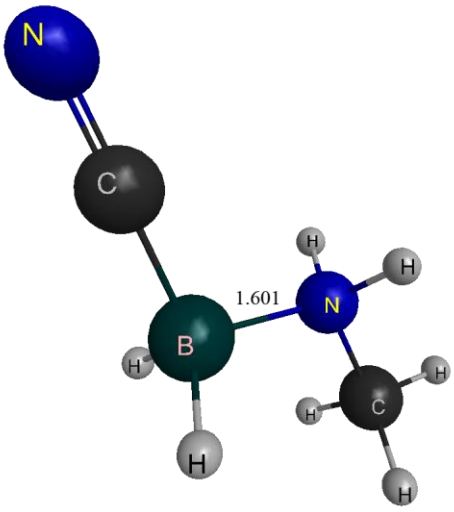


40. BH_3-Histidine

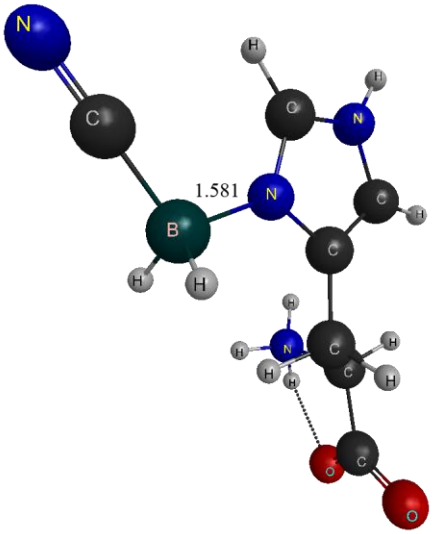


39. BH_3-Imidazole

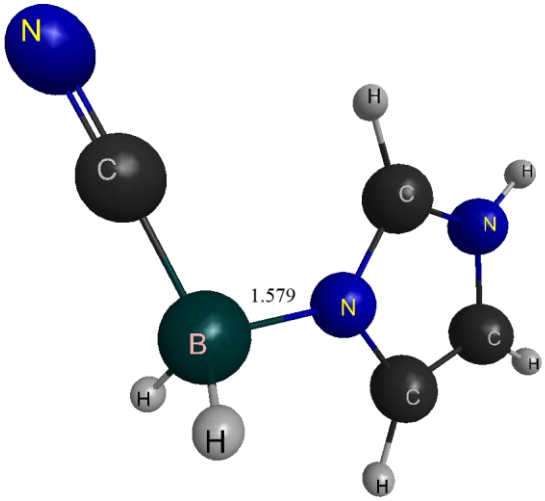
Figure 2.13 Endpoint structures, Boranes (38-40)



41. BH_2CN-NH_2CH_3

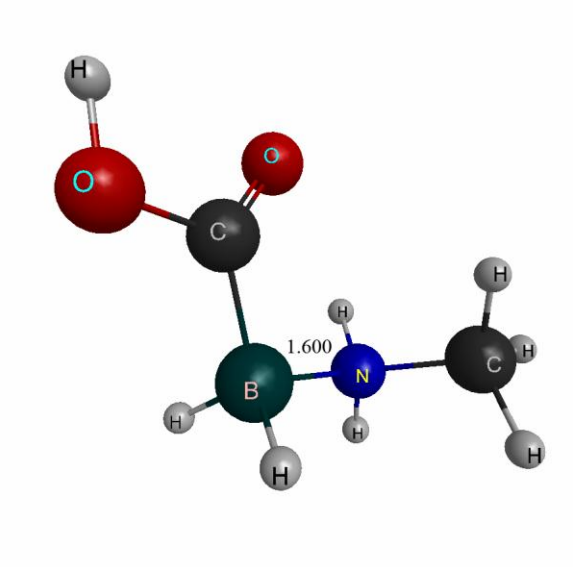


43. BH_2CN-Histidine

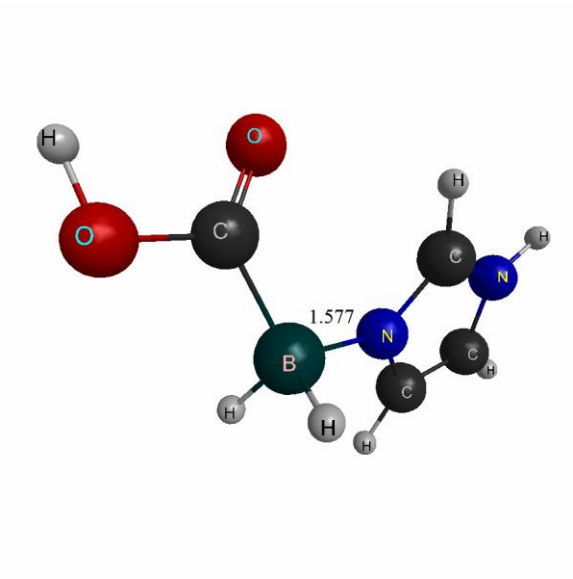


42. BH_2CN-Imidazole

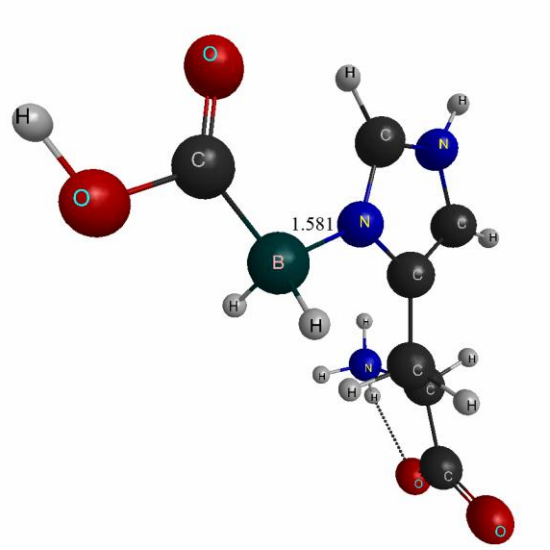
Figure 2.14 Endpoint structures Endpoint structures, Cyanoboranes (41-43)



44. BH_2COOH-NH_2CH_3



45. BH_2COOH-Imidazole



46. BH_2COOH-Histidine

Figure 2.15 Endpoint structures, Carboxylicboranes (44-46)

Table 2.1 Structure summary

Starting points			
1	NH_3	BH_3	
2	NH_3	BH_2CN	
3	NH_3	BH_2COOH	
Fragments	Lewis Base 1	Borane	Lewis Base 2
4		BH_3	
5		BH_2CN	
6		BH_2COOH	
7			NH_3
8			NH_2CH_3
9			Imidazole
10			Histidine
Water intermediates			
11		BH_3	H_2O
12		BH_2CN	H_2O
13		BH_2COOH	H_2O
Transition states to ammonia	Lewis Base 1	Borane	Lewis Base 2
14	NH_3	BH_3	NH_3
15	NH_3	BH_3	NH_2CH_3
16	NH_3	BH_3	Imidazole
17	NH_3	BH_3	Histidine
18	NH_3	BH_2CN	NH_3
19	NH_3	BH_2CN	NH_2CH_3
20	NH_3	BH_2CN	Imidazole
21	NH_3	BH_2CN	Histidine
22	NH_3	BH_2COOH	NH_3
23	NH_3	BH_2COOH	NH_2CH_3
24	NH_3	BH_2COOH	Imidazole
25	NH_3	BH_2COOH	Histidine
Transitions states to water	Lewis Base 1	Borane	Lewis Base 2
26	H_2O	BH_3	NH_3
27	H_2O	BH_3	NH_2CH_3
28	H_2O	BH_3	Imidazole
29	H_2O	BH_3	Histidine
30	H_2O	BH_2CN	NH_3
31	H_2O	BH_2CN	NH_2CH_3
32	H_2O	BH_2CN	Imidazole
33	H_2O	BH_2CN	Histidine
34	H_2O	BH_2COOH	NH_3
35	H_2O	BH_2COOH	NH_2CH_3
36	H_2O	BH_2COOH	Imidazole
37	H_2O	BH_2COOH	Histidine
Endpoints	Lewis Base 1	Borane	Lewis Base 2
38		BH_3	NH_2CH_3
39		BH_3	Imidazole
40		BH_3	Histidine
41		BH_2CN	NH_2CH_3
42		BH_2CN	Imidazole
43		BH_2CN	Histidine
44		BH_2COOH	NH_2CH_3
45		BH_2COOH	Imidazole
46		BH_2COOH	Histidine

Key details about the geometry of the calculated molecules have been outlined in Table 2.2, Table 2.3 and Table 2.4. Table 2.2 catalogues geometries obtained at the MP2/cc-pVTZ level of theory. Table 2.3 and Table 2.4 contain the geometries at MP2/cc-pVTZ PCM level of theory for ammonia and water containing compounds respectively. These tables are included primarily for completeness.

There are however, a few notable features to these geometries. In transition states, the borane moiety is typically planar rather than the tetrahedral configuration observed in the endpoints. The overall geometry of the other molecular components remains approximately the same as the starting material with the only exception being the orientation to the borane moiety.

In the carboxylated borane molecules the carbonyl oxygen typically interacts with hydrogens on the adjacent amine or water. In the gas phase it appears that this interaction is strong enough that the carboxy group does not rotate to the nearly planar conformation that is observed in the solvated state.

Table 2.2 Selected MP2/cc-pVTZ (Gas Phase) Optimized Structures (Angstroms and Degrees).Histidine geometry selected as a zwitterion Starting point and end point structures are redundant therefore references in N^2 columns are duplicates in alternate naming schema for comparison.

Structure	$r(N^1B)$	$r(\alpha B)$	$\angle(\alpha BN^1)$	$\phi(\alpha BNH)$	$\phi(O{=}CBN^1)$	$\angle(N^1BN^2)$	$r(BN^2)$	$r(\beta N^2)$	$\angle(\beta N^2B)$	$\phi(\beta N^2B\alpha)$	$\angle(\alpha BN^2)$	$\phi(O{=}CBN^2)$
1*	1.650	1.206	104.8	180			1.650	1.013	111.1	180	104.8	
2*	1.637	1.577	104.2	180			1.637	1.014	110.7	180	104.2	
3*	1.631	1.611	100.7	-0.5	0.3		1.631	1.029	103.6	-0.5	100.7	0.3
14	2.211	1.197	90.0	180		180.0	2.211	1.011	111.2	180	90.0	
15	2.130	1.199	91.1	179		178.7	2.209	1.459	110.0	180	90.2	
16	2.187	1.199	90.5	-180		179.1	2.200	1.320	124.1	10.9	88.6	
18	2.164	1.569	89.0	-180		177.9	2.164	1.012	112.3	180	89.0	
19	2.087	1.571	90.5	180		179.4	2.165	1.461	110.3	-180	88.8	
20	2.134	1.569	89.8	180		178.0	2.158	1.323	128.2	-0.1	88.2	
22	2.135	1.622	92.1	179	139	178.2	2.157	1.011	112.7	161	88.7	-39.0
23	2.063	1.622	93.3	178	141	177.7	2.157	1.457	111.4	170	88.7	-37.8
24	2.130	1.610	89.6	174	118	177.4	2.152	1.321	122.4	-143	89.0	-64.7
38							1.634	1.474	113.5	-180	105.5	
39							1.608	1.324	125.7	-0.3	105.0	
41							1.625	1.477	113.0	180	104.7	
42							1.602	1.327	126.8	-0.1	104.7	
44							1.616	1.477	114.0	84.9	101.5	10.0
45							1.594	1.327	129.0	1.3	110.0	-0.5

Table 2.3 Selected MP2/cc-pVTZ PCM (Solvated) Optimized Structures Ammonia (Angstroms and Degrees)1*

Structure	r(N^1B)	r(αB)	∠(αBN^1)	ϕ(αBNH)	ϕ(O=CBN^1)	∠(N^1BN^2)	r(BN^2)	r(βN^2)	∠(βN^2B)	ϕ(βN^2Bα)	∠(αBN^2)	ϕ(O=CBN^2)
1*	1.620	1.211	106.7	-179			1.620	1.211	107.7	-179	106.7	
2*	1.608	1.588	107.0	-177			1.608	1.016	110.9	-177	107.0	
3*	1.609	1.611	104.0	5.3	-0.9		1.609	1.021	107.0	5.3	104.0	-0.9
14	2.201	1.190	89.8	-180		179.8	2.210	1.013	111.7	180	90.3	
15	2.140	1.201	90.6	176		179.3	2.188	1.462	111.5	-180	90.1	
16	2.202	1.198	90.2	179		179.4	2.073	1.326	126.0	8.8	89.2	
17	2.232	1.192	89.5	175		179.5	2.129	1.327	120.5	-39.2	90.1	
18	2.137	1.574	90.1	180		180.0	2.138	1.013	108.7	180	89.9	
19	2.086	1.576	91.1	177		179.2	2.127	1.465	109.1	-177	89.6	
20	2.135	1.574	90.4	-179		179.6	2.107	1.327	132.4	9.8	89.8	
21	2.165	1.569	89.7	175		178.1	2.094	1.330	127.2	8.8	90.9	
22	2.154	1.604	88.6	177	98.6	177.4	2.158	1.013	108.1	177	88.8	-81.7
23	2.097	1.606	90.4	180	102	178.6	2.152	1.464	108.6	178	88.5	-78.5
24	2.148	1.602	89.2	174	99.8	177.1	2.124	1.327	129.2	45.7	88.5	-81.9
25	2.185	1.600	88.0	-180	-92.3	177.5	2.096	1.329	121.9	-57.7	88.5	88.2
38							1.613	1.479	113.6	180	113.6	
39							1.591	1.330	125.5	-7.2	106.1	
40							1.590	1.331	125.0	-1.5	105.9	
41							1.601	1.483	112.6	-18	107.7	
42							1.579	1.333	128.2	11.3	108.3	
43							1.581	1.333	126.8	0.3	107.7	
44							1.600	1.481	114.0	80.1	104.7	11.1
45							1.577	1.332	127.2	37.5	110.0	-1.3
46							1.581	1.334	128.2	-0.7	111.9	0.4

Table 2.4 Selected MP2/cc-pVTZ PCM (Solvated) Optimized structures Water (Angstroms and Degrees)

Structure	r(N^1B)	r(αB)	∠(αBN^1)	ϕ(αBNH)	ϕ(O=CBN^1)	∠($N^1B N^2$)	r(BN^2)	r(βN^2)	∠(βN^2B)	ϕ(βN^2Bα)	∠(αBN^2)	ϕ(O=CBN^2)
11	1.636	1.204	102.8	171.1								
12	1.607	1.582	104.5	170.0								
13	1.600	1.608	98.6	-5.6	2.4							
26	2.024	1.194	92.6	167.5		178.0	2.482	1.013	114.4	174.7	85.8	
27	1.963	1.194	96.1	48.3		175.6	2.490	1.463	106.5	179.1	88.1	
28	2.016	1.195	93.2	-176.2		178.1	2.440	1.326	126.8	18.8	86.2	
29	2.075	1.194	93.2	53.9		179.5	2.352	1.328	121.0	-158.0	86.5	
30	1.956	1.571	93.5	-169.0		177.8	2.384	1.013	109.6	178.6	85.7	
31	1.894	1.574	97.9	61.3		176.5	2.387	1.464	106.8	-176.0	85.5	
32	1.948	1.572	96.9	51.8		178.0	2.340	1.327	132.8	6.8	85.0	
33	2.058	1.567	94.1	37.0		176.7	2.288	1.331	118.5	44.6	87.1	
34	1.975	1.627	91.7	6.2	4.1	174.8	2.357	1.013	110.7	-177.8	93.7	175.0
35	1.917	1.627	93.5	-175.1	-10.5	174.5	2.356	1.460	107.1	-175.1	93.5	169.1
36	2.015	1.623	90.6	16.0	-18.6	172.9	2.323	1.327	140.1	-3.0	95.6	157.9
37	2.045	1.613	88.6	41.9	-48.3	175.1	2.296	1.329	115.7	-76.8	89.3	127.3

Discussion

Structural

Histidine was chosen as a prototypical model for biologically relevant base sites in situ. It was chosen because it is often found in active sites of proteins.[35] [36] It appears to be important to use full residues when attempting to understand these systems. In this study imidazole was used as one model for histidine. While imidazole did properly predict the zwitterionic histidine geometries and, enthalpy and free energy sign it did not match energies in magnitude. The enthalpies and free energy predictions varied by as much as 50%, therefore this more compact model does not reliably represent the histidine residue.

A solvent model is required to have reasonable expectation of modeling an in situ system. PCM was selected for its computational tractability when compared to an explicit solvent model. The compounds change both geometrically and energetically when solvation models are utilized. Two geometric indicators allow insight into the changes introduced by an aqueous solvent (Table 2.2 & Table 2.3). First, solvation typically shortens the boron-nitrogen bond in both reactants and products; this is expected because the dative bonds between boron and nitrogen have large electronegativity differences. The shortening of this bond indicates that adding a solvent stabilizes the dative bonds. Second, in all transition states the geometry about borane is planar and the geometry of boron substituents is usually planar as well.

The gas phase carboxy transitions double bonded oxygen is rotated towards the new adduct. In solvent this rotation of the carboxy group is less pronounced. The calculated barrier heights suggest that reactions of carboxylic acid substituted boranes are kinetically

less facile. Carboxy reactions also tend to be less exothermic for transfer reactions to Lewis base sites. It can also be inferred that the solvent helps to mitigate the role of dynamics.

Reaction Pathways

Figure 2.5 describes the reaction pathways for borane moiety transfers. The S_N1 pathway is not shown here for clarity and because previous work has been found it to be higher in energy than the S_N2 pathway. A comparison of S_N2 and S_N1 pathways can be made with column A and G from Table 2.5 respectively. Column A is the S_N2 barrier height, and Column G is the BDE for the indicated system. Previous work has shown that the mean S_N1 pathway is roughly 91% of the BDE in solvent[23] The mean S_N2 pathway barrier is on average only 42% of the BDE. These results confirm previous findings but at higher level of theory.

The direct S_N2 pathway versus a water assisted pathway is shown in Figure 2.16. The S_N2 mechanism is represented by the transfer through the T_1 transition state. A two-step water assisted transfer uses T_2 and T_3. Table 2.5 reports the currently calculated values based on Figure 2.16.

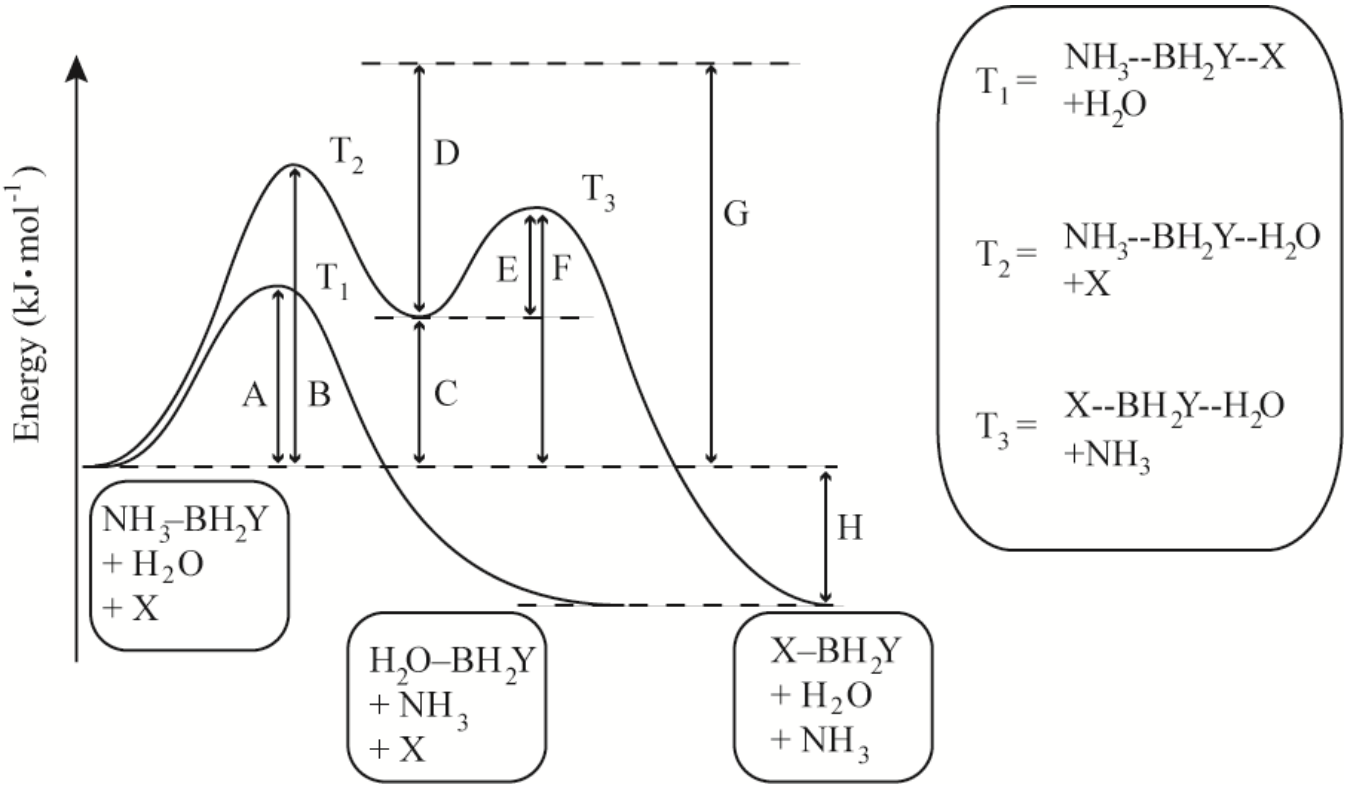


Figure 2.16 Reaction pathways. See Table 2.5 Energy differences for Figure 2.5 at MP2/cc-pVTZ PCM (kJ·mol^{-1} for energy reference.

A. S_N2 Barrier
B. H2O first transition state barrier
C. H2O intermediate energy
D. Dissociation energy of water intermediate
E. Barrier height to second water transition
F. Overall barrier to second water transition
G. Dissociation energy
H. Reaction energy

4) T_1. Transition state for S_N2 pathway
5) T_2. Transition state between ammonia and water intermediate
6) T_3. Transition state between water intermediate and final products

Table 2.5 Energy differences for Figure 2.5 at MP2/cc-pVTZ PCM (kJ·mol^{-1})

Y	X	A	B	C	D	E	F	G	H
H	NH_3	90.9	108.7	79.9	91.0	28.8	108.7	171.0	0.0
H	CH_3NH_2	79.9	108.7	79.9	91.0	21.5	101.4	171.0	-18.3
H	Im	88.1	108.7	79.9	91.0	26.1	106.1	171.0	-7.2
H	Hist	68.2	108.7	79.9	91.0	4.5	84.4	171.0	0.0
CN	NH_3	84.9	112.4	89.1	120.1	23.3	112.4	209.3	0.0
CN	CH_3NH_2	71.0	112.4	89.1	120.1	13.5	102.6	209.3	-19.6
CN	Im	79.2	112.4	89.1	120.1	17.7	106.8	209.3	-10.3
CN	Hist	71.2	112.4	89.1	120.1	2.9	92.1	209.3	-1.0
COOH	NH_3	93.6	114.0	72.4	129.6	41.6	114.0	202.1	0.0
COOH	CH_3NH_2	80.8	114.0	72.4	129.6	30.4	102.9	202.1	-21.0
COOH	Im	87.8	114.0	72.4	129.6	38.9	111.4	202.1	-7.0
COOH	Hist	72.1	114.0	72.4	129.6	20.7	93.2	202.1	4.0

The key to understanding is in considering the overall barrier for each pathway. A water assisted transfer of a borane to a Lewis base site is a two-step process. First, the borane moiety transfers to the water Lewis base site, then the transition from the water to the putative physiological base site.

Comparing column A to columns B, C and F it is possible to determine the feasibility of water assisted reaction pathway. Column A again, is the direct S_N2 barrier height. Along the water assisted pathway, column B is the energy barrier height to water transition state T_2. Column F is the energy barrier for the second transition state T_3. The barrier F from the water intermediate to final Lewis base is lower in energy than the barrier B from the initial compound to the water intermediate. The barrier to transition therefore will be determined by T_2, column B.

Column C is the energy of water intermediate. The water intermediate C is often higher in energy than the entire barrier for the S_N2 pathway A. A is lower in all cases in energy than the barrier to the water intermediary B. A is 72% of the water assisted

mechanism barrier on average; suggesting that the S_N2 pathway is a preferred mechanism of transfer of a borane moiety to Lewis base sites in situ.

Solvent Effects

Comparing solvent effects for the S_N2 pathway reveals several significant differences. When the system is solvated (PCM) the bond dissociation energies and barrier heights are increased relative to the gas phase (Table 2.6 & Table 2.7).

Table 2.6 Boron-nitrogen bond dissociation energies ($kJ \cdot mol^{-1}$)

B-N BDE	Gas	PCM
H_3N-BH_3	141.0	171.0
CH_3H_2N-BH_3	163.6	189.3
Imidazole-BH_3	155.5	178.2
Histidine-BH_3		171.0
H_3N-BH_2CN	171.1	209.3
CH_3H_2N-BH_2CN	195.5	228.9
Imidazole-BH_2CN	197.9	219.6
Histidine-BH_2CN		210.2
H_3N-BH_2COOH	185.6	202.1
CH_3H_2N-BH_2COOH	209.1	223.1
Imidazole-BH_2COOH	198.0	209.1
Histidine-BH_2COOH		198.1

Table 2.7 Transitions state barrier height (kJ·mol-1)

Transition state barrier	Gas	PCM
H_3N-BH_3-NH_3	54.6	90.9
H_3N-BH_3-NH_2CH_3	44.0	79.9
H_3N-BH_3-Imidazole	54.1	88.1
H_3N-BH_3-Histidine		68.2
H_3N-BH_2CN-NH_3	45.8	84.9
H_3N-BH_2CN-NH_2CH_3	33.1	71.0
H_3N-BH_2CN-Imidazole	34.7	79.2
H_3N-BH_2CN-Histidine		71.2
H_3N-BH_2COOH-NH_3	66.9	93.6
H_3N-BH_2COOH-NH_2CH_3	54.4	80.8
H_3N-BH_2COOH-Imidazole	63.82	87.8
H_3N-BH_2COOH-Histidine		72.1

All molecules studied herein are polar, there is added stability associated with the inclusion of a solvent model. The barrier height (Table 2.7) typically increases from gas to solvent more than the bond dissociation energy (Table 2.6) do from gas to solvent, although barrier heights remain well below the bond dissociation energies.

Transfer thermodynamics

Table 2.8 Scaled Enthalpy of reaction at 298K (kJ·mol^{-1})

ΔH^{298}	Gas	PCM
H_3N-BH_3	0.0	0.0
CH_3H_2N-BH_3	-23.4	-21.9
Imidazole-BH_3	-18.4	-13.5
Histidine-BH_3		-11.5
H_3N-BH_2CN	0.0	0.0
CH_3H_2N-BH_2CN	-24.9	-21.9
Imidazole-BH_2CN	-30.1	-17.5
Histidine-BH_2CN		-12.8
H_3N-BH_2COOH	0.0	0.0
CH_3H_2N-BH_2COOH	-24.1	-22.2
Imidazole-BH_2COOH	-16.2	-13.2
Histidine-BH_2COOH		-9.2

Table 2.9 Scaled Gibbs free energy at 298K (kJ·mol^{-1})

ΔG^{298}	Gas	PCM
H_3N-BH_3	0.0	0.0
CH_3H_2N-BH_3	-21.4	-19.8
Imidazole-BH_3	-17.5	-11.4
Histidine-BH_3		-8.1
H_3N-BH_2CN	0.0	0.0
CH_3H_2N-BH_2CN	-22.3	-20.5
Imidazole-BH_2CN	-26.7	-18.5
Histidine-BH_2CN		-9.2
H_3N-BH_2COOH	0.0	0.0
CH_3H_2N-BH_2COOH	-21.6	-20.8
Imidazole-BH_2COOH	-14.5	-12.6
Histidine-BH_2COOH		-0.8

The enthalpies for borane transfer (Table 2.8) are less exothermic when solvated. In all solvated cases, transfer to biologically relevant sites becomes less favored than in the gas phase (Table 2.9). The transfer to the histidine like all the Lewis base sites studied is free energy favored in all cases.

The equilibrium constants for the transfer to histidine are provided in Table 2.10. The equilibrium constants indicate that nontrivial amounts of boranes, that have shown physiological activity, will transfer from small molecule amines to Lewis base sites in the body. The number of histidine nitrogen lone-pairs in the body is relatively large. The probability of transfer to any particular active site, however, may be relatively low. Thus, the hypothesis of competitive Lewis acid/base chemistry can explain not only the overall pharmacological activity, but also the rather general nature of the activity. Borane moiety transfer is potentially plausible for any number of small molecules that include a boron-nitrogen dative bond precisely because there are so many possible transfer sites that are both thermodynamically and kinetically accessible.

Table 2.10 Equilibrium constants

K_{eq}	Gas	PCM
H_3N-BH_3	1.00E+00	1.00E+00
CH_3H_2N-BH_3	5.54E+03	3.00E+03
Imidazole-BH_3	1.16E+03	9.97E+01
Histidine-BH_3		2.60E+01
H_3N-BH_2CN	1.00E+00	1.00E+00
CH_3H_2N-BH_2CN	8.15E+03	3.95E+03
Imidazole-BH_2CN	4.80E+04	1.71E+03
Histidine-BH_2CN		4.10E+01
H_3N-BH_2COOH	1.00E+00	1.00E+00
CH_3H_2N-BH_2COOH	6.05E+03	4.49E+03
Imidazole-BH_2COOH	3.41E+02	1.62E+02
Histidine-BH_2COOH		1.36E+00

The proposed transfer of borane moieties via competitive Lewis acid/base mechanism for physiological action is viable both thermodynamically and kinetically. Looking specifically at histidine products in solvent in the following table reference we can see that while the transfer to the biologically relevant histidine residue is not the most exothermic product studied (Table 2.8), borane transfer to histidine does have the lowest barrier heights (Table 2.7). The free energy of transfer for the histidine is spontaneous (Table 2.9). This thermodynamic viability is the most compelling evidence that this mechanism is the responsible for the observed physiological activities.

Boron-Nitrogen Bond

Computational studies also afford information about bonding trends. These trends could help suggest strategies for modifying boron-nitrogen dative bond containing substances to enhance biological activity for example. From the perspective of chemical bonding trends, hydrogen, carboxylic acid groups and cyano groups are increasingly electron withdrawing. Dative bond dissociation energies would be expected to increase respectively, because electron withdrawing from boron will tend to make it a stronger Lewis acid.[24] This predicted behavior was observed in the solvent phase, while the trend is not observed in the gas phase (Table 2.6). For example, the BDE for H_3N-BH_3, H_3N-BH_2CN and H_3N-BH_2COOH are 141.0, 171.1 and 185.6 respectively. We would however expect that the cyano bond energy would be greater than the carboxylic acid's. This trend is observed in the gas phase with 171.0, 209.3 and 202.1 respectively

The noncovalent interaction between the carboxylic acid group and the amines may explain the anomalous behavior. As previously mentioned in the PCM solvated phase the

solvent helps provides a better model of the systems, and the anomalously large contribution of the non-covalent interaction is removed between the adjacent groups, and therefore the expected trend emerges.

A final interesting comparison is afforded by considering the pK_b. pK_b is a measure of Lewis acid/base chemistry when the Lewis acid is a proton The comparison of pK_b of methylamine (3.37), ammonia (4.75), imidazole (6.92), and histidine (Im) (7.96) suggests that imidazole and histidine are weaker bases than ammonia.[37,38] If boranes behave like protons, imidazole and histidine should have the weaker dative bond to boron and methyl amine a stronger bond. However, when comparing reaction energies for transfer of the borane from the amine-borane to imidazole and histidine (Table 2.8), an exothermic transition is observed. This effect may be due to an electrostatic interaction between the positively charged regions of the imidazole ring and the negatively charged hydrogens or substituents on the boron. This indicates that other considerations need to be accounted for when attempting to understand the nature of Lewis acid/base chemistry than just the electron withdrawing/donating character of substituents.

Conclusions

There are three steps to trying to understand the mechanism by which the boron-nitrogen containing compounds are active. Step one is to correlate activity to the proposed mechanism. Step two is to test the mechanism for feasibility and step three is to determine the dynamics of such borane transfer in models of biological systems**.** Previous work has shown that molecules with strong dative bonds are less likely to be show physiological activity. Current work shows that competitive Lewis acid/base chemistry is a model that is

both thermodynamically and kinetically viable**.** Moving forward, the ability to model the dynamics of borane transfer could provide more detailed molecular scale insight into the pharmacological mechanisms of molecules that contain B-N dative bonds. Such studies will require the incorporation of non-covalent interactions in addition to the quantum mechanical calculations associated with bond breaking and formation. The development of methodologies for calculation of non-covalent interactions is an area of active research, and recent reports suggest a number of possible ways that such calculations may become feasible in the future.[39-42]

This study has looked at boron-nitrogen dative bonds in an attempt to understand the nature of Lewis acid/base transfers as a proposed mechanism for the activity of certain small molecule pharmacophores. Several levels of ab initio theory were used as well as PCM as a solvent model. The bond dissociation energies and barrier heights calculated for this study confirm the S_N2 pathway is significantly lower in energy than bond dissociation or water assisted transfer and therefore is the preferred transfer mechanism for Boron-Nitrogen dative bonds.

This study has shown that the kinetic barrier to transition is not prohibitively large. The calculations in solvated states indicate that transfers of borane moieties to residues such as histidine are thermodynamically favorable. The calculated equilibrium constants also show that nontrivial amounts of borane moieties will transfer to Lewis base sites in situ. This supports the hypothesis that the physiological activity observed for a wide range of borane containing species[16-21] may be explained by competitive Lewis acid/base chemistry. A borane moiety transferring to potentially important Lewis base sites thus inhibiting a crucial pathway remains a viable mechanism for the observed activity.

References

(1) Barth, R. F.; Soloway, A. H.; Fairchild, R. G.; Brugger, R. M. *Cancer* **1992**, *70*, 2995-3007.

(2) Hawthorne, M. F. *Angew Chem Int Edit* **1993**, *32*, 950-984.

(3) Slatkin, D. N. *Brain* **1991**, *114*, 1609-1629.

(4) Barth, R. F.; Yang, W.; Soloway, A. H. *Cancer Res.* **1997**, *57*, 1129.

(5) Burnham, B. S.; Chen, S. Y.; Sood, A.; Spielvogel, B. F.; Hall, I. H. *Met.-Based Drugs* **1995**, *2*, 221.

(6) Hall, I. H.; Burnham, B. S.; Rajendran, K. G.; Chen, S. Y.; Sood, A.; Spielvogel, B. F.; Shaw, B. R. *Biomed. Pharmacother.* **1993**, *47*, 79.

(7) Hall, I. H.; Spielvogel, B. F.; Griffin, T. S.; Docks, E. L.; Brotherton, R. J. *Res Commun Chem Path* **1989**, *65*, 297-317.

(8) Hall, I. H.; Das, M. K.; Harchelroad, F.; Jr, n.; McPhail, A. T.; Spielvogel, B. F. *J. Pharm. Sci.* **1981**, *70*, 339.

(9) Sood, A.; Sood, C. K.; Spielvogel, B. F.; Hall, I. H.; Wong, O. T.; Mittakanti, M.; Morse, K. *Arch Pharm* **1991**, *324*, 423-432.

(10) Eisen, E. J.; Jones, E. E.; Rajendran, K. G.; Hall, I. H. *Growth Dev. Aging* **1996**, *60*, 7.

(11) Hall, I. H.; Wong, O. T.; Sood, A.; Sood, C. K.; Spielvogel, B. F.; Shrewsbury, R. P.; Morse, K. W. *Pharmacol. Res.* **1992**, *25*, 259.

(12) Das, M. K.; Maiti, P. K.; Roy, S.; Mittakanti, M.; Morse, K. W.; Hall, I. H. *Arch. Pharm. (Weinheim, Ger.)* **1992**, *325*, 267.

(13) Hall, I. H.; Chen, S. Y.; Rajendran, K. G.; Sood, A.; Spielvogel, B. F.; Shih, J. *Environ. Health Perspect.* **1994**, *102*, 21.

(14) Sood, A.; Spielvogel, B. F.; Shaw, B. R.; Carlton, L. D.; Burnham, B. S.; Hall, E. S.; Hall, I. H. *Anticancer Res* **1992**, *12*, 335-343.

(15) Sood, C. K.; Sood, A.; Spielvogel, B. F.; Yousef, J. A.; Burnham, B.; Hall, I. H. *J. Pharm. Sci.* **1991**, *80*, 1133.

(16) Hall, I. H.; Burnham, B. S.; Elkins, A.; Sood, A.; Powell, W.; Tomasz, J.; Spielvogel, B. F. *Met.-Based Drugs* **1996**, *3*, 155.

(17) Hall, I. H.; Burnham, B. S.; Chen, S. Y.; Sood, A.; Spielvogel, B. F.; Morse, K. M. *Met.-Based Drugs* **1995**, *2*, 1.

(18) Hall, I. H.; Rajendran, K. G.; Chen, S. Y.; Wong, O. T.; Sood, A.; Spielvogel, B. F. *Arch Pharm* **1995**, *328*, 39-44.

(19) Hall, I. H.; Starnes, C. O.; Mcphail, A. T.; Wisianneilson, P.; Das, M. K.; Harchelroad, F.; Spielvogel, B. F. *J Pharm Sci* **1980**, *69*, 1025-1029.

(20) Rajendran, K. G.; Burnham, B. S.; Chen, S. Y.; Sood, A.; Spielvogel, B. F.; Shaw, B. R.; Hall, I. H. *J. Pharm. Sci.* **1994**, *83*, 1391.

(21) Sood, A.; Sood, C. K.; Spielvogel, B. F.; Hall, I. H.; Wong, O. T. *J. Pharm. Sci.* **1992**, *81*, 458.

(22) Rajendran, K. G.; Chen, S. Y.; Sood, A.; Spielvogel, B. F.; Hall, I. H. *Biomed. Pharmacother.* **1995**, *49*, 131.

(23) Fisher, L. S., University of Wisconsin - Milwaukee, 1999.

(24) Fisher, L. S.; McNeil, K.; Butzen, J.; Holme, T. A. *J Phys Chem B* **2000**, *104*, 3744-3751.

(25) Pople, J. A.; Binkley, J. S.; Seeger, R. *Int J Quantum Chem* **1976**, 1-19.

(26) Dill, J. D.; Pople, J. A. *J. Chem. Phys.* **1975**, *62*, 2921.

(27) Ditchfie.R; Hehre, W. J.; Pople, J. A. *J Chem Phys* **1971**, *54*, 724-&.

(28) Hehre, W. J.; Ditchfie.R; Pople, J. A. *J Chem Phys* **1972**, *56*, 2257-&.

(29) Roothaan, C. C. J. *Rev Mod Phys* **1951**, *23*, 69-89.

(30) Miertus, S.; Scrocco, E.; Tomasi, J. *Chem Phys* **1981**, *55*, 117-129.

(31) Dunning, T. H. *J Chem Phys* **1971**, *55*, 716-&.

(32) Schmidt, M. W.; Baldridge, K. K.; Boatz, J. A.; Elbert, S. T.; Gordon, M. S.; Jensen, J. J.; Koeski, S.; Matsunaga, N.; Nguyen, K. A.; Su, S.; Windus, T. L.; Dupuis, M.; Montgomery, J. A. *J. Comput. Chem.* **1993**, *14*, 1347.

(33) Gordon, M. S.; Schmidt, M. W. In *Theory and Applications of Computational Chemistry, the First Fourty Years*; Dykstra, C. E., Frenking, G., Kim, K. S., Scuseria, G. E., Eds.; Elsevier: Amsterdam, 2005, p 1167-1189.

(34) Sinha, P.; Boesch, S. E.; Gu, C. M.; Wheeler, R. A.; Wilson, A. K. *J Phys Chem A* **2004**, *108*, 9213-9217.

(35) Morgan, W. T.; Muster, P.; Tatum, F.; Kao, S. M.; Alam, J.; Smith, A. *J Biol Chem* **1993**, *268*, 6256-6262.

(36) Boots, J. W. P.; Vandongen, W. D.; Verheij, H. M.; Dehaas, G. H.; Haverkamp, J.; Slotboom, A. J. *Bba-Protein Struct M* **1995**, *1248*, 27-34.

(37) Lide, D. R. e. *CRC Handbook of Chemistry and Physics*; 76th Student ed.; CRC Press: Boca Raton, 1995.

(38) Budavari, S. *The Merck index : an encyclopedia of chemicals, drugs, and biologicals*; 12th ed.; Merck: Whitehouse Station, NJ, 1996.

(39) Foster, M. E.; Sohlberg, K. *J Chem Theory Comput* **2010**, *6*, 2153-2166.

(40) Alecu, I. M.; Truhlar, D. G. *J Phys Chem A* **2011**, *115*, 2811-2829.

(41) Contreras-Garcia, J.; Johnson, E. R.; Keinan, S.; Chaudret, R.; Piquemal, J. P.; Beratan, D. N.; Yang, W. T. *J Chem Theory Comput* **2011**, *7*, 625-632.

(42) Schneebeli, S. T.; Bochevarov, A. D.; Friesner, R. A. *J Chem Theory Comput* **2011**, *7*, 658-668.

CHAPTER 3. COMPLETE ROTATION FOR THE EVALUATION OF THE POTENTIAL ENERGY SURFACE

To be submitted to the Journal of Chemical Theory and Computation

Abstract

A tool and methodology has been created for the creation of atomics coordinates that permit systematic searching of the configuration space on a potential energy surface. This tool is called Complete Rotation for the Evaluation of the Potential Energy Surface (CREPES). This tool produces input files for the GAMESS computational software package. It was originally created to generate saccharide conformers but can be used generally to generate a complete set of rotational conformers. As a calibration of the methodology, conformational searching of the carbohydrates, D-glucose and D-galacturonic acid were performed.

Introduction

Computer modeling of carbohydrates has a long history however there are many problems left to solve due to saccharide versatility and complexity. Saccharides are polar and very flexible. Much computational work on saccharides has been done at the Molecular mechanics (MM) level of theory[1-24] again in part because of their size and conformational complexity. MM has tried to parameterize these stereochemical features to varying effectiveness with parameters to account for the anomeric effect, the exoanomeric effect, and the gauche effect. Adding a water solvent further complicates the system.

The study of saccharides with computational methods is challenging due to the number of possible conformations, this significantly increases the time necessary to find low-

lying conformational energy states. The prevalence of mono-, di-, and polysaccharides in biological systems motivates continued efforts to find computationally reasonable methods for exploring such conformational energy states. One approach for handling the conformational challenge of polysaccharides is to focus on the inter-saccharide conformations of the monosaccharide units. This approach ignores the variations of the configurations of the monosaccharide unit and targets the two rotations between monomer units[4,6,25-28].

It is important to study the monosaccharide units themselves for two reasons. First, computational understanding of monosaccharides will inform understanding of their biological properties and stereochemical constraints. Second, understanding of the structural conformations of monosaccharide units allow for informed modeling of in the aforementioned unitized models for polysaccharides. In comparison to molecular mechanics, there have been relatively few attempts to understand these saccharides at the ab initio level of theory. Most efforts largely focused on low energy conformers of D-glucose.[29-34]

The large number of degrees of freedom for D-glucose and similar compounds makes finding low energy conformers significantly more challenging than many other systems. Consider three expected staggered rotational conformations of each of the six hydroxyl groups, alpha and beta anomers, and a minimum of two overall structural conformations, and around 3000 conformational structures are hypothesized for D-glucose.

Rotational analyses of D-glucose and similar saccharides have typically been limited to computationally rapid calculations like MM; the use of Hartree-Fock (HF) has been limited due largely to computational costs. Current computational power and ability to

automate conformational searches with HF calculations, however, has rendered this limitation significantly less pertinent.

Some methods that can be used to find low energy conformers are simulated annealing (SA)[35] in molecular dynamics and Monte Carlo (MC)[36,37] methods. SA and MC are proven methods for sampling the PES. These methods while possibly being more efficient do not inform the user about the entire potential energy surface. The systematic search of the potential energy surface ensures confidence in minima identified and identifies for chemical intuition components that are of concern to the configuration of the structure.

Complete Rotation for the Evaluation of the Potential Energy Surface (CREPES) is an automation tool that utilizes user-defined input and a GAMESS input deck, generating additional input decks of conformational iterations needed to sample the potential energy surface. This chapter will describe the use of CREPES to identify the low-lying energy conformers of D-glucose and D-galacturonic acid. Structures found with the CREPES program will be compared to those found in experimentation and using other searching methods.

Methods

The CREPES tool allows users to define three parameters for each rotation for structure generating: the rotating functional groups (compound rotations are allowed), rotation step size, and total angle of rotation for each rotating group. CREPES creates input deck files based on a supplied GAMESS[38,39] input file. Once these parameters are set CREPES generates all the iterative combinations of the defined rotations, operating on the $DATA group of supplied input deck.

GAMESS input decks consist of a series of input groups. Each input group controls variables used in a GAMESS calculation. The number of input groups and the content of those groups can vary widely. The only required GAMESS input group is the $DATA group. This input group contains the nuclear coordinates for the calculation. Since CREPES operates on the $DATA group, it can be used for any level of theory because other GAMESS input groups are unaffected by the conformational searching. CREPES systematically names output files by the defined rotations relative to the input file.

CREPES can generate a significant amount of configurations with a small number of parameters. This creates some logistic difficulties in dealing with the content that is created and in analysis. Grep, sed, awk, shell, and Python scripts become absolutely essential in managing data. Scripts that are frequently used in managing the large amount of data are found in the appendix at the end of this chapter. Often several scripts in conjunction are required. Since so much of this is controlled by file names proper verbose naming of files becomes more important than labels in the GAMESS files. Scripts are required to submit the GAMESS jobs created by CREPES. They are required to parse the content of the log files and to format it for analysis. Typically data is output to comma separated value (CSV) files for utilization in a spreadsheet application such as Microsoft Excel. Evaluation and visualization was assisted with macros in Excel. Results from spreadsheet analysis are used as selection criteria for further analysis.

A three step methodology was developed in order to balance the accuracy and efficiency in the search for all important conformers. The three steps are bulk screening, fine screening and final optimization. CREPES is used to generate the initial set of structures for bulk screening. For our current case study CREPES was used to generate the α and β

structures for the expected $^4C_1$ chair conformation of D-glucose[29] and D-Galacturonic acid[40] for bulk screening.

It is possible to use high-level theory for optimizations of each generated configuration. This strategy, however, is not an efficient use of computational resources. Large portions of the configurational space are sufficiently high energetically that they do not contribute significantly to models of the chemical behavior of saccharides. A bulk screening process, to select a smaller subset for refinement, is therefore a much more efficient method for screening of the configurational energy topology.

Minimizing saccharide structures requires polarizable basis set and a solvent model for proper modeling.[15] Therefore, bulk screening of structures for the current study involved optimizing at RHF/3-21G(d) PCM. Generated configurations are optimized using a truncated optimization of 20 steps with step sizes of 0.1 bohr. The small step size and limited steps, helps ensure that generated structures are allowed to relax, while modestly being constrained to the potential energy wells that they have been labeled for by CREPES. The modest basis set allows calculations to be sufficiently fast for bulk-level screening and the large number of putative conformers calculated. Jobs for this step were submitted one configuration per 8-core node and typically completed within 90 seconds.

Fine screening involves selecting a small subsection of the bulk pass to further optimize. Using this methodology the user selects energetically low-lying structures, and any additional structures of interest that may be desired. To determine candidates for fine screening, all structures are plotted along a total energy axis to empirically identify which conformers are low enough in energy to warrant further scrutiny.

Natural breaks in the density of states arise that are typically within 3-7 RT of the minimum energy structure. kT, the more commonly recognized energy scaling factor is a molecular scale value. At macroscopic scale the more appropriate term is RT. RT = kT *N_A. At 298.15 K RT is 2.479kJ·mol^{-1} or 0.944 mE_h·mol^{-1}.

The number of structures selected for fine screen is dependent on the user's selection of cutoffs. Typically 20-60 structures are selected for a saccharide though this can be significantly higher if there are many structures low on the conformational energy profile.

Fine screening optimizations maintain 0.1 Bohr step size at MP2/6-31G(d) PCM level of theory. Unlike the bulk screening where optimization is intentionally cut short, these structures are fully optimized. Depending on the users intended purpose the treatment of fine screening selected structures can vary. If the user is only attempting to find the global minima depending on the observed energy gaps, the bulk of fine screening structures are ignored and only the minimum energy structure moves on to final optimization. It is important to evaluate how fine screening structures to determine candidates for further investigation.

If the fine screening energy structures are to be used as a representative model of a distribution of states, the fine screening energy structures must be analyzed with significantly more scrutiny. Fully optimized structures commonly move from the original CREPES identified configuration to a different local minimum on the potential energy surface. These minima are often geometrically very similar and consequently are redundant with other identified minima structures. Only a single minimum energy structure is chosen from the redundant conformers and that single conformer is carried forward in subsequent screening and analysis.

For example, two structures may be low lying. 0.0.0.0.0.0 and 0.0.0.0.0.120. The starting structures are nearly identical with only the final rotation being offset by 120 degrees. During full optimization of 0.0.0.0.0.120 the OH group of rotation 6 may have moved back to a structure very similar to 0.0.0.0.0.0. The structures are named 0.0.0.0.0.0 and 0.0.0.0.0.120 however they have the nearly identical optimized structure and are in the same potential energy well but are near but not equal minima energies. At this point the structure with the lower absolute energy is selected for further consideration and the other is discarded.

The fine screening step is a time consuming process that includes significant user judgment, but it is an important step. Each configuration must be visualized and checked carefully for proper configuration naming as described above. Scripts designed to identify atoms that have significantly moved since the initial configuration assists the process of checking for proper configuration naming. This step is particularly important when determining Boltzmann populations.

The last step is final optimization. The lowest energy configuration after fine screening is completely optimized at increasingly higher levels of theory. Other structures of particular interest may also be optimized from the fine structure results for comparison. The fine screening pass for glucose was through the CCSD(T)/cc-pVTZ//MP2/cc-pVTZ PCM [41-44,45-47] level of theory. Galacturonic acid fine screening was through MP2/cc-pVTZ PCM level of theory.

Nomenclature

One significant complication for describing large-scale conformational modeling of saccharides lies in the nomenclature system. A few systems have been devised over the years to logically describe the stable conformers of saccharides. The briefest of these systems will be used here, with notation to other potentially familiar nomenclature schemes.

The terms anomeric carbon and anomeric hydroxyl refer to C(1) and O(1)H respectively (Figure 3.1). Saccharides can cyclize by a hydroxyl group attacking either of two sides of the aldehyde end of the sugar. The stereochemistry differs only at the anomeric carbon; they are known as α and β anomers. The β position is defined as the anomeric -OH being on the same side of the ring as the terminating carbon (C(6)) while the α position is on the opposite side (Figure 3.1).

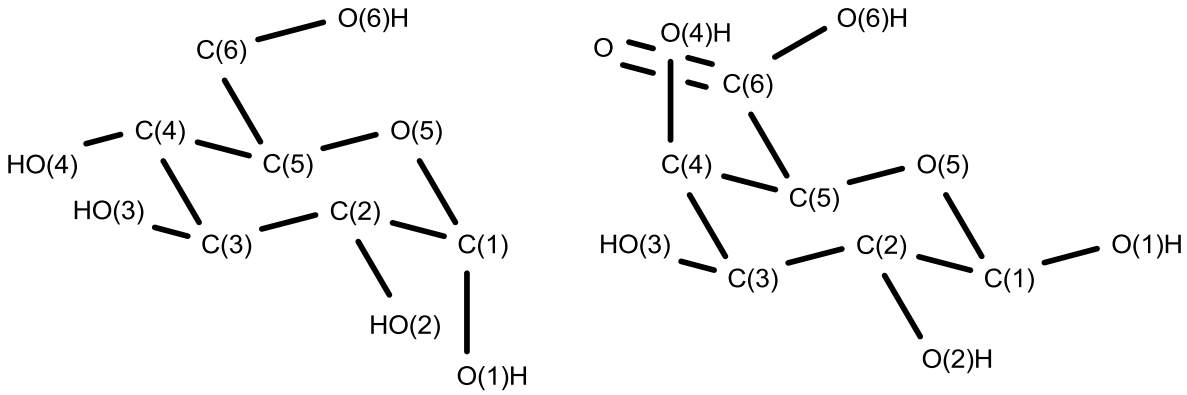


Figure 3.1 α structure of glucose and β structure of galacturonic acid (Shown in dash projection to emphasize stereochemistry)

Once cyclized, the remaining hydroxyl groups each have three expected configurations gauche(-), gauche(+), and trans (**ḡ**, **g**, and **t** respectively). Figure 3.2 illustrates these different positions for the D-glucose configuration. The symbol indicates the dihedral angle H-O-C(n)-C(n-1) relationship (or the ring oxygen in the anomeric hydroxyl case). **ḡ** denotes a clockwise (viewed O to C) rotation; **g** indicates a counter clockwise rotation and **t** the anti position. The hydroxylated methyl group is indicated with a capital representation of this symbol (**Ḡ**, **G** and **T** they are sometimes referred to as gg, gt, and tg respectively). For historic reasons, in glucose the hydroxymethyl group are labeled with respect to O(5) rather than C(4). Galacturonic acid's carboxylic acid group is designated based on the hydroxyl oxygen (O(6)) dihedral angle to the ring oxygen O(5).

Saccharides can cyclize in different ring sizes. Five and six membered rings are referred to as furanosyl (*f*) and pyranosyl (*p*) respectively. For brevity sugars are noted in the following shorthand way, anomer-ISOMER-sugar short notation *cyclization* A for acidic form, e.g. α-D-Gal*p*A.

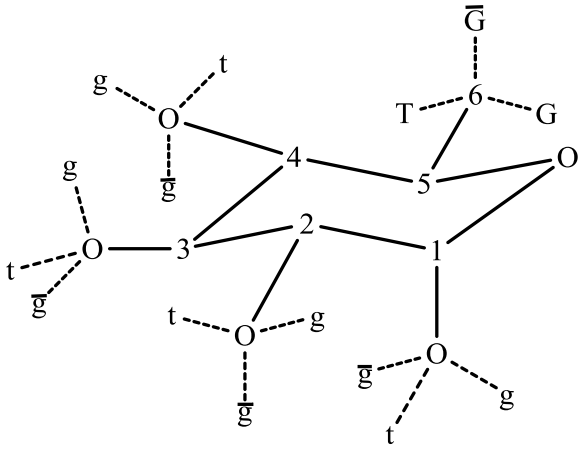


Figure 3.2 α-D-Glu*p* (Shown in dash projection to emphasize stereochemistry)

There are two conformations of ring structure that are expected to be energetically low lying. They are both of the two chair conformations. The two conformations are labeled - $^{1}C_{4}$ and $^{4}C_{1}$. This notation refers to the stereochemistry with the low numbered carbons in the foreground (ring oxygen in the back) and C(1) on the right side. $^{4}C_{1}$ refers to having C(1) in the down position and C(4) in the up position $^{1}C_{4}$ is the opposite configuration.

Results

Glucose

All glucose structures identified in this study are $^{4}C_{1}$. $^{1}C_{4}$ is unstudied here as it is virtually unobserved experimentally and has been reasonably well modeled previously.[29] All 729 conformations of both α and β -D-Glu*p* were bulk screened. Fine screening cutoffs were made at seven and five RT above minimum identified structure energy for α and β respectively. As is previously described, cutoffs are decided empirically based on the observed density of states. The relative absolute energies of each of the 20-step optimized bulk screening configurations have been plotted in Figure 3.3 and Figure 3.4 to diagram the conformational energy profile. The energy cut off limit of the fine screening pass is shown in red. The fine screening cutoff for the alpha structure is higher than that of the beta configuration to allow for some sampling of a larger portion of the PES. The density of states of beta states significantly increases with a threshold cutoff any higher than 5RT above minimum energy structure.

The mapping of the conformational energy profile is strictly based on the name of the states. Therefore different portions of the map which are largely separated may indeed by very structurally similar so it is important to select as much of the system as is computationally tractable, while simultaneously ignoring states that have low occupation.

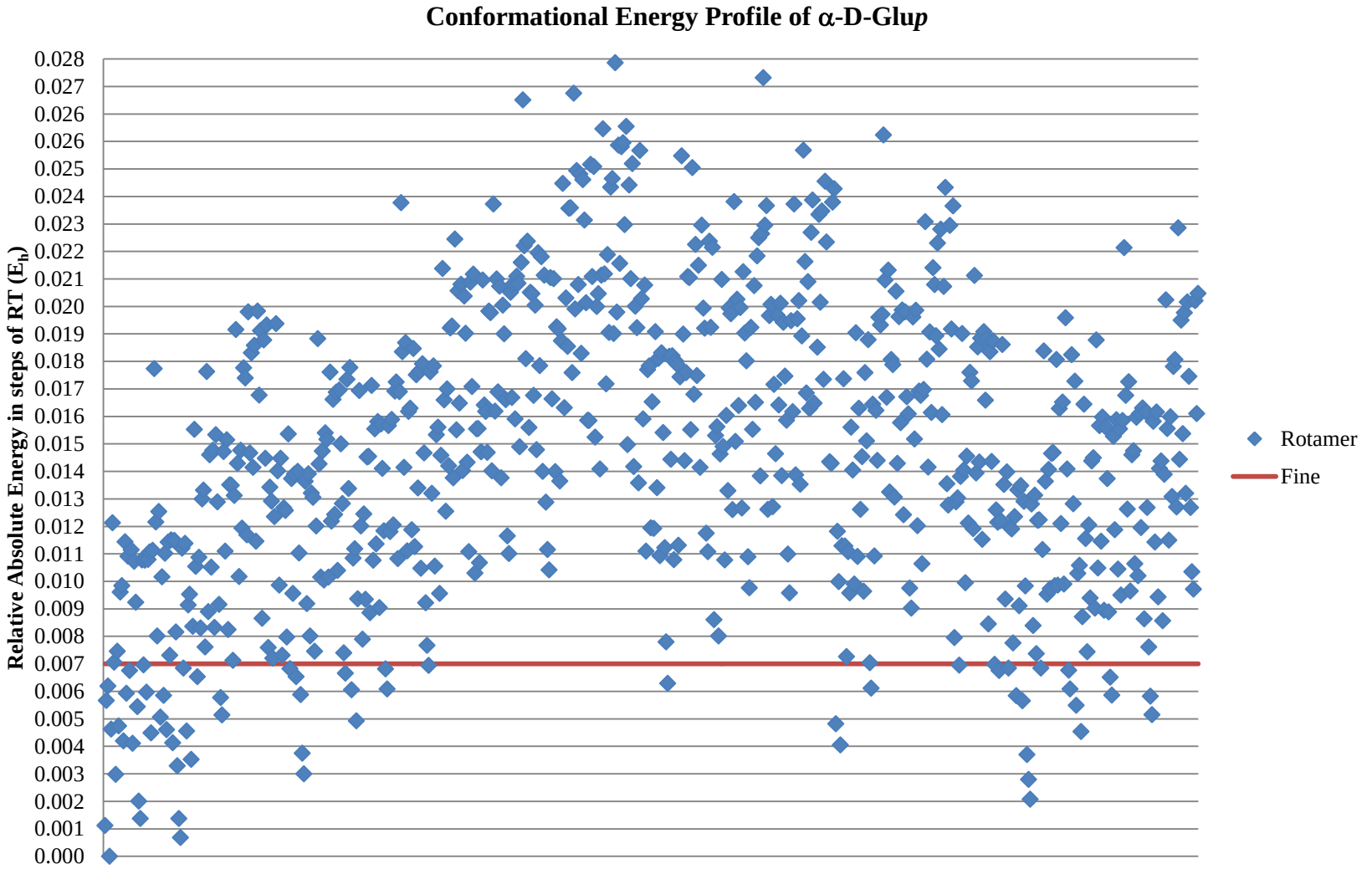


Figure 3.3 Conformational energy profile of α-D-Glu*p* modeled by bulk screening energies
(RT is approximately equal to $1mE_h$)

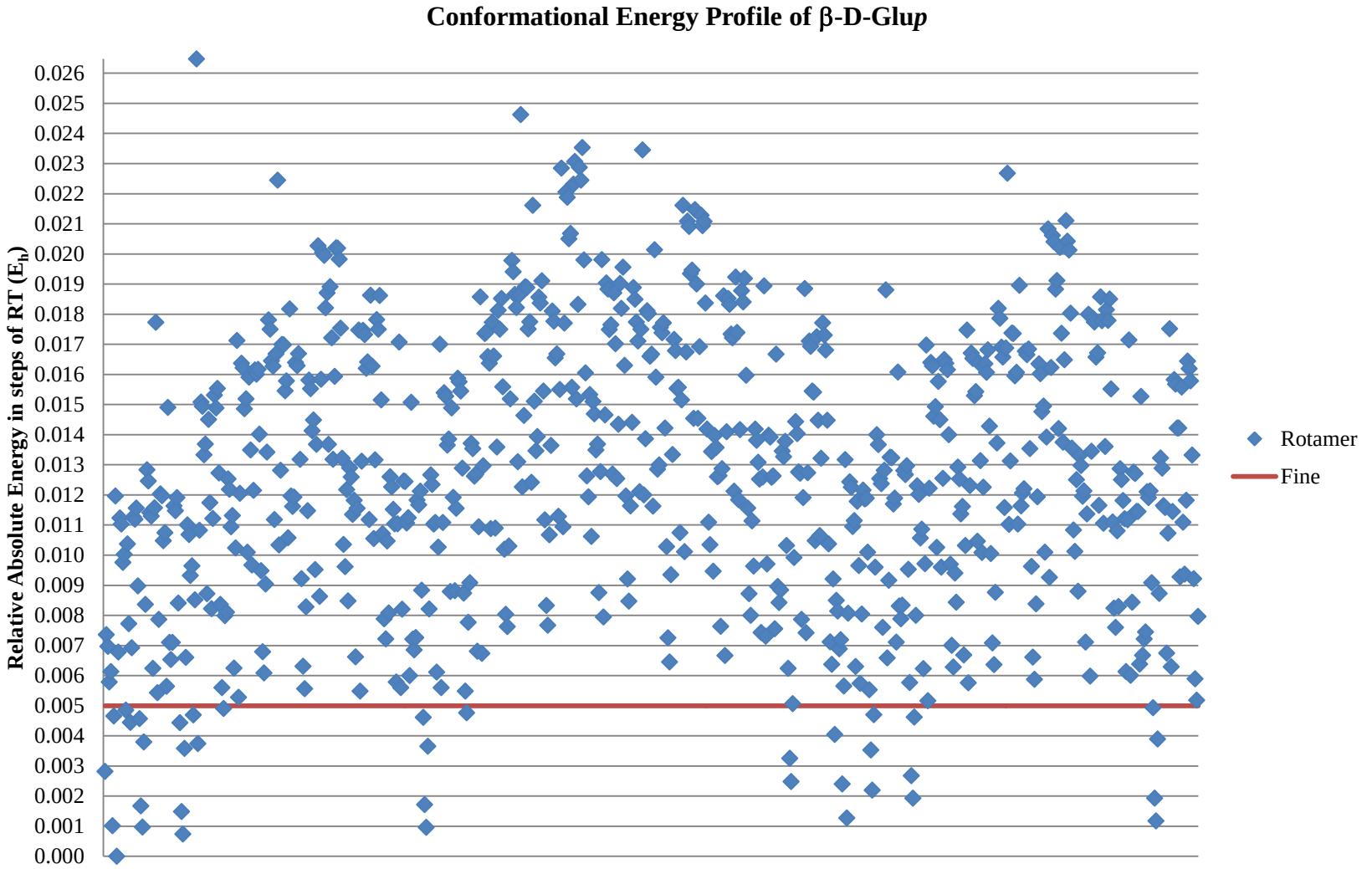


Figure 3.4 Conformational energy profile of β-D-Glu*p* modeled by bulk screening energies
(RT is approximately equal to 1mE_h)

The apparent global minimum energy structure identified for α-D-Glu*p* is **gḡḡgḠg**, the counter clockwise rotation (CCR), and β-D-Glu*p* minimum is **ḡggḡḠg** (CCR). The alpha structure is lower in energy than the beta structure at both CCSD(T)/cc-pTVZ and MP2/cc-pVTZ PCM by 6.3 and 5.4 kJ·mol^{-1} respectively. This is in reasonable agreement with a previous similar calculation.[30]

The anomeric equilibrium of α and β anomers in aqueous solution has been establish to be 36:64 experimentally as determined by NMR.[48] The β anomer populations therefore have lower free energy by -1.426 kJ·mol^{-1}. The calculation of free energies from the molecules identified by the fine screening is afforded via the information derived from the determination of the hessian. From this work a Boltzmann distribution of conformers was generated (3.1).

$$P_x = \left(\frac{e^{-E_i/kT}}{\sum_i e^{-E_i/kT}} \right) \qquad \textbf{(3.1)}$$

The total population of α anomers were summed and compared to the sum of β anomers to calculate the free energy of an α → β anomerization at 298.15K (3.2).

$$G = -RT \ln \left(\frac{\Sigma P_\beta}{\Sigma P_\alpha} \right) \qquad \textbf{(3.2)}$$

Table 3.1 contains the Boltzmann distribution of populations for the alpha states of D-glucose, as selected by fine screening and calculated by equation 3.1 for the solvated phase. Table 3.2 shows the beta populations.

Table 3.1 Relative Free Energies and Boltzmann-Averaged Populations (298 K) of α-D-glu*p*

α-D-glu*p*	G(kJ)	P(%)
0.0.0.0.0.0	0.0	0.9
0.0.0.0.0.120	0.6	0.7
0.0.0.0.0.240	-2.7	2.5
0.0.0.0.120.0	1.6	0.4
0.0.0.0.240.120	-0.3	1.0
0.0.0.120.0.0	0.6	0.7
0.0.0.120.120.0	1.8	0.4
0.0.0.120.240.120	-0.5	1.0
0.0.0.240.0.0	-0.2	0.9
0.0.0.240.120.120	2.2	0.3
0.0.0.240.120.240	1.2	0.5
0.0.0.240.240.120	-0.3	1.0
0.0.120.120.0.0	-0.3	1.0
0.0.120.120.120.0	1.1	0.5
0.0.120.240.0.0	0.2	0.8
0.0.120.240.120.120	1.4	0.5
0.0.120.240.120.240	1.5	0.5
0.0.120.240.240.120	-0.7	1.1
0.0.240.0.0.0	-0.1	0.9
0.0.240.0.120.0	0.7	0.6
0.0.240.0.240.120	-0.2	0.9
0.0.240.240.120.120	0.6	0.7
0.0.240.240.120.240	0.9	0.6
0.120.120.240.0.0	-0.3	1.0
0.120.120.240.120.120	0.2	0.8
0.120.120.240.120.240	1.0	0.6
0.120.240.240.120.240	0.9	0.6
0.240.0.0.0.0	0.0	0.9
0.240.0.0.120.0	0.3	0.8
0.240.0.240.120.120	0.9	0.6
0.240.0.240.120.240	1.4	0.5
0.240.120.240.120.240	0.7	0.6
240.120.0.0.120.0	-1.6	1.6
240.120.120.0.0.0	-2.2	2.1
240.120.120.0.120.0	-1.2	1.4
240.120.120.120.0.0	-0.6	1.1
240.120.120.240.0.0	-1.2	1.4
240.120.120.240.120.120	0.5	0.7
240.120.120.240.120.240	1.3	0.5
240.120.240.0.120.0	-0.4	1.0
240.120.240.240.120.120	1.4	0.5
240.120.240.240.120.240	2.0	0.4
240.240.0.0.0.0	-0.6	1.1
240.240.0.0.120.0	0.5	0.7
240.240.0.240.120.120	1.3	0.5
240.240.0.240.120.240	2.1	0.4
240.240.120.240.120.120	2.4	0.3
240.240.120.240.120.240	3.0	0.3
	Total α Population	38.6

Table 3.2 Relative Free Energies and Boltzmann-Averaged Populations (298 K) of β-D-glu*p*

β-D-glu*p*	G(kJ)	P(%)
0.0.0.0.0.0	-4.4	5.0
0.0.0.0.120.240	-2.2	2.1
0.0.0.0.240.240	-2.9	2.7
0.0.0.120.120.240	-1.4	1.5
0.0.0.120.240.240	-2.6	2.4
0.0.0.240.240.0	-0.2	0.9
0.0.0.240.240.120	-1.4	1.5
0.0.120.240.120.240	-1.7	1.7
0.0.120.240.240.0	-0.9	1.2
0.0.120.240.240.120	0.6	0.7
0.0.240.0.120.240	-2.9	2.7
0.0.240.0.240.240	-1.4	1.5
0.0.240.240.240.120	-1.3	1.4
0.240.120.240.120.240	-4.3	4.8
0.240.120.240.240.0	-1.0	1.3
0.240.120.240.240.120	-0.3	1.0
0.240.240.240.240.120	1.2	0.5
120.240.120.240.240.0	-4.5	5.2
120.240.120.240.240.120	-4.5	5.2
240.0.0.0.0.0	-3.0	2.8
240.0.0.0.120.240	-3.0	2.8
240.0.0.0.240.240	-1.1	1.3
240.0.0.240.240.0	-0.5	1.1
240.0.0.240.240.120	1.0	0.6
240.0.120.120.120.0	0.1	0.8
240.0.120.240.240.120	1.0	0.6
240.240.120.240.120.240	-3.6	3.7
240.240.120.240.240.0	-1.8	1.8
240.240.120.240.240.120	-0.8	1.2
240.240.120.240.240.120	-1.3	1.5
	Total β Population	61.4

Table 3.3 contains the Boltzmann distribution of populations for the alpha states of D-glucose, as selected by fine screening and calculated by equation 3.1 for the gas phase. Table 3.4 contains the beta populations.

Table 3.3 Relative Free Energies and Boltzmann-Averaged Populations (298 K) of α-D-glu*p*, gas phase

α-D-glu*p* Gas	G(kJ)	P(%)
0.0.0.0.0.0	-0.4	0.8
0.0.0.0.0.120	-2.1	1.5
0.0.0.0.0.240	-0.5	0.8
0.0.0.0.120.0	-0.9	0.9
0.0.0.0.240.120	0.6	0.5
0.0.0.120.0.0	0.2	0.6
0.0.0.120.120.0	-1.6	1.2
0.0.0.120.240.120	-1.6	1.2
0.0.0.240.0.0	1.1	0.4
0.0.0.240.120.120	-3.4	2.5
0.0.0.240.120.240	1.2	0.4
0.0.0.240.240.120	-0.4	0.8
0.0.120.120.0.0	1.0	0.4
0.0.120.120.120.0	-0.7	0.9
0.0.120.240.0.0	0.8	0.5
0.0.120.240.120.120	0.2	0.6
0.0.120.240.120.240	-1.6	1.2
0.0.120.240.240.120	-0.8	0.9
0.0.240.0.0.0	0.2	0.6
0.0.240.0.120.0	-1.4	1.1
0.0.240.0.240.120	-1.7	1.3
0.0.240.240.120.120	-2.6	1.8
0.0.240.240.120.240	-1.9	1.4
0.120.120.240.0.0	-0.8	0.9
0.120.120.240.120.120	-0.2	0.7
0.120.120.240.120.240	-1.4	1.1
0.120.240.240.120.240	-1.9	1.4
0.240.0.0.0.0	-0.8	0.9
0.240.0.0.120.0	-0.7	0.9
0.240.0.240.120.120	-1.7	1.3
0.240.0.240.120.240	-2.4	1.7
0.240.120.240.120.240	0.0	0.7
240.120.0.0.120.0	-0.4	0.8
240.120.120.0.0.0	-2.4	1.7
240.120.120.0.120.0	-0.4	0.8
240.120.120.120.0.0	-0.5	0.8
240.120.120.240.0.0	0.5	0.5
240.120.120.240.120.120	-0.7	0.9
240.120.120.240.120.240	1.2	0.4
240.120.240.0.120.0	0.8	0.5
240.120.240.240.120.120	0.6	0.5
240.120.240.240.120.240	-0.2	0.7
240.240.0.0.0.0	-0.5	0.8
240.240.0.0.120.0	0.5	0.5
240.240.0.240.120.120	0.4	0.6
240.240.0.240.120.240	-0.6	0.8
240.240.120.240.120.120	0.3	0.6
240.240.120.240.120.240	-0.4	0.8
	Total α Population	43.5

Table 3.4 Relative Free Energies and Boltzmann-Averaged Populations (298 K) of β-D-glu*p*, gas phase

β-D-glu*p* Gas	G(kJ)	P(%)
0.0.0.0.0.0	-2.9	2.1
0.0.0.0.120.240	-1.5	1.2
0.0.0.0.240.240	-2.4	1.7
0.0.0.120.120.240	-2.6	1.8
0.0.0.120.240.240	-1.6	1.3
0.0.0.240.240.0	-2.4	1.7
0.0.0.240.240.120	-2.9	2.1
0.0.120.240.120.240	-1.3	1.1
0.0.120.240.240.0	-1.9	1.4
0.0.120.240.240.120	-2.8	2.0
0.0.240.0.120.240	-1.8	1.3
0.0.240.0.240.240	-4.3	3.7
0.0.240.240.240.120	-3.6	2.8
0.240.120.240.120.240	-1.9	1.4
0.240.120.240.240.0	-2.5	1.8
0.240.120.240.240.120	-3.1	2.2
0.240.240.240.240.120	-2.4	1.7
120.240.120.240.240.0	-3.0	2.2
120.240.120.240.240.120	-3.2	2.3
240.0.0.0.0.0	-3.0	2.2
240.0.0.0.120.240	-1.8	1.4
240.0.0.0.240.240	-1.9	1.4
240.0.0.240.240.0	-2.9	2.1
240.0.0.240.240.120	-1.5	1.2
240.0.120.120.120.0	-2.2	1.6
240.0.120.240.240.120	-2.2	1.6
240.240.120.240.120.240	-4.1	3.4
240.240.120.240.240.0	-2.2	1.6
240.240.120.240.240.120	-2.9	2.1
240.240.120.240.240.120	-2.9	2.1
	Total β Population	56.5

The free energy of anomerization, 1.15 kJ·mol^{-1} and populations ratio, 39:61 correlate well with experiment (Table 3.5). To investigate the role of a solvent model in this result, single point energies were calculated from the gas phase configurations using the PCM geometries. The Boltzmann averaged populations and corresponding free energy are shown

in Table 3.5 for comparison. In the gas phase the anomeric effect provides notable stabilization of the axial position of the α anomer, which in turn influences the distributions of conformers and the resulting free energies. As one would expect the PCM model presents a better model for the experimentally observed system.

Table 3.5 Free Energy of anomerization. α→ β Glucose

	ΣP_α	ΣP_β	G(kJ)
MP2/631G(d) PCM	0.39	0.61	-1.16
MP2/631G(d) (gas phase)	0.44	0.57	-0.64
Exp	0.36	0.64	-1.46

In addition to the anomerization free energy, the orientation of the exocyclic hydroxymethyl group at C(6) has been an elusive problem in D-glucose.[30,31,34] Ratios of **G̃:G:T** have been determined experimentally. The experimentally determined ratios have been determined by Nashida et al at 56:44:0 and 53:45:2 in α and β is respectively[49]. Barnett and Naidoo using a new Karplus equation derived a ratio of 35:57:3[50]. The **T** position is experimentally negligibly observed.[51]

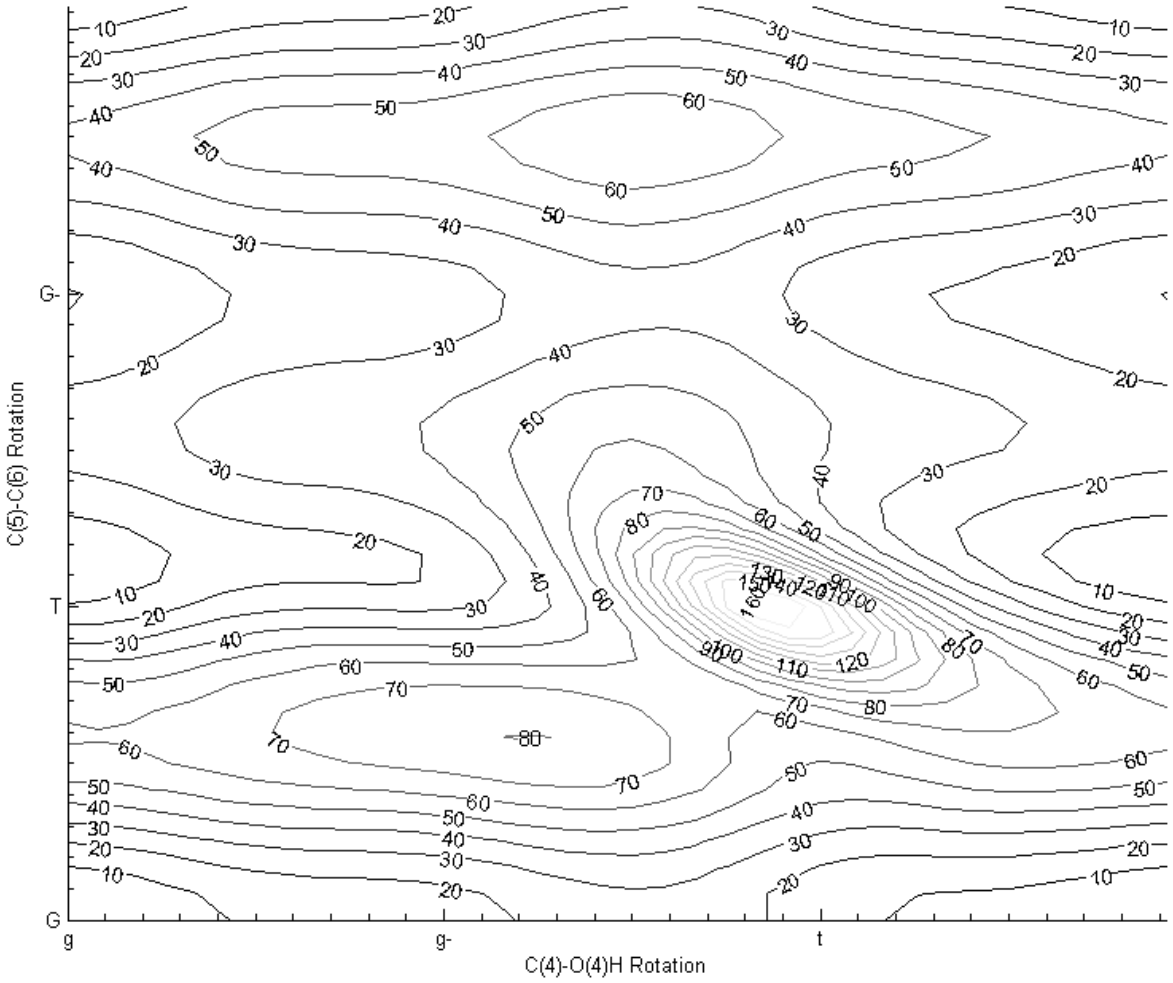


Figure 3.5 O(4)H–C(6)OH rotational PES map contours in increments of 10 kJ·mol^{-1} from minimum

The rotation O(4)H and C(6)OH is of particular importance to the orientation of the exocyclic rotation of the hydroxyl methyl group. van der Waals interactions can obscure the potential energy surface however, CREPES allows us to get a good description of these 2 dimensions of the potential energy surface (PES). CREPES was used to map the PES in 10 degree step sizes along both O(4)H and C(6)OH rotations (Figure 3.5). This was done at HF/6-31G(d) known for having a fortuitous canceling of error.[30]

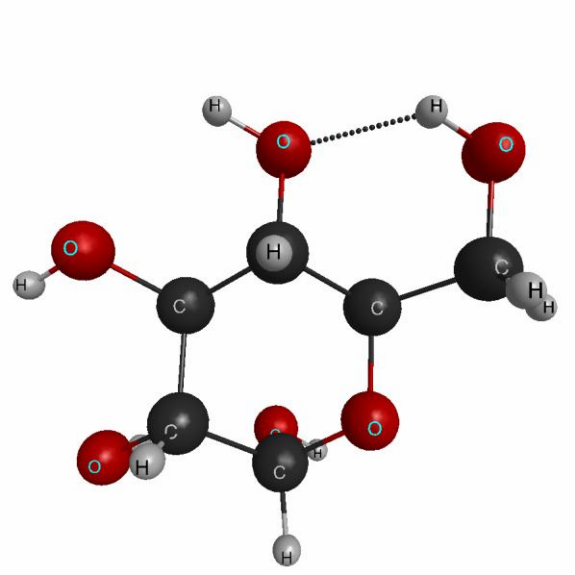


Figure 3.6 T rotation of C(6)OH with g rotation of O(4)H

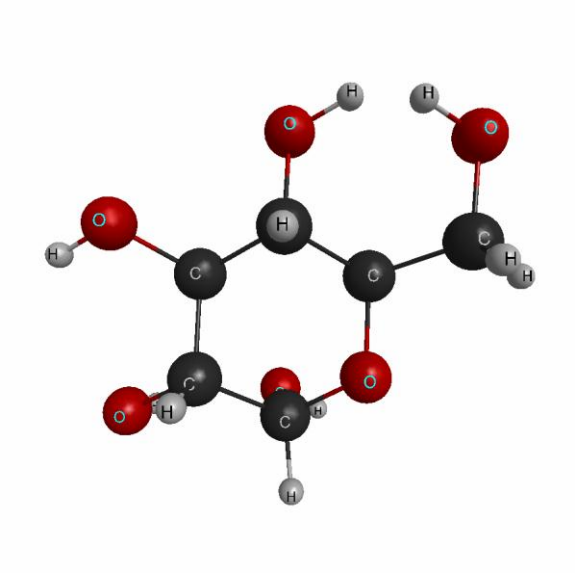


Figure 3.7 T rotation of C(6)OH with t rotation of O(4)H

In the current study, the **T** configuration was frequently selected for fine screening (after the bulk screening pass with CREPES) due to its low energy. The **T** position provides a low lying valley for the hydroxyl to reside in because intramolecular O(6)H-O(4) hydrogen boding can occur(Figure 3.6). While gauche effects are frequently cited as the primary

component of selection preference it insufficiently explains the exclusion of the **T** rotation.[50] A **t** rotation of O(4)H causes a strong steric repulsion (Figure 3.7). This strong interaction may, in addition to those effects, help explain its near total exclusion.

The continuum (PCM) model for solvent does not include explicit water molecules so it cannot account for the reported bridged hydrogen bonds found in the **G** conformation[50]. To investigate this potential shortcoming a calculation was carried out with the addition of an ab initio water as a bridge from the exocyclic hydroxyl to the ring oxygen. The calculation was carried out for the three minimum energy configurations of the **$\bar{G}$**, **G**, and **T**. Typically 2 hydrogen bonds were formed for the **$\bar{G}$**, and **G**. However for **T**, the hydroxyl is in the anti position and consequently only 1 hydrogen bond could form. This result leads to **T** being the highest energy conformer in both α and β by 4.6 and 11.8 $kJ \cdot mol^{-1}$ respectively. Thus, in addition to the potential energy features inherent to the molecule (Figure 3.5) the role of solvent in conformational stability more completely explains the lack of experimental observation of the **T** conformer.

In this work, D-glucose is being used to demonstrate some potential uses of CREPES. Further ab inito work with explicit solvent models could be done to make a better approximation of this system however such studies are outside the scope of the screening methodology being described.

Galacturonic Acid

The utility of CREPES lies in the ability to scan the PES of any saccharide to identify important conformers. This capability is particularly important for sugars that have few or no computational modeling efforts reported. For example, galacturonic acid is present in large

amounts in plant cell walls as it is the monomer that forms the backbone of the pectin fraction. Despite its importance in cell wall structure and the capture of cations it is virtually unstudied as its monomer unit.[52]

CREPES has been used to generate 2916 structures for both the alpha and beta conformers of D-Galacturonic acid. The bulk screening process identified 59 alpha structures and 62 beta structures for fine screening within 5 RT of the minima energy structure (Figure 3.8, Figure 3.9). Ultimately 13 alpha and 13 beta structures turned out to be unique fully optimized conformers of galacturonic acid. All calculated galacturonic acid structures are the $^4C_1$ chair conformation. As with glucose galacturonic acid's primarily observed conformation is the $^4C_1$ conformer. The α-D-Gal*p*A minimum structure is **gḡḡḡ0-0** (CCR) and the β-D-Gal*p*A is **ḡgḡḡ0-0** (CCR). The minimum alpha structure is lower in energy than the beta structure at MP2/cc-pVTZ PCM by 6.35 kJ·mol^{-1}.

Table 3.6 contains the Boltzmann distribution of populations for the alpha states of D-galacturonic acid, as selected by fine screening and calculated by equation 3.1 for the solvated phase. Table 3.7 shows the beta populations. The ratio of α and β anomers is found to be 37:63 this corresponds to a free energy change of 1.29 kJ·mol^{-1}.

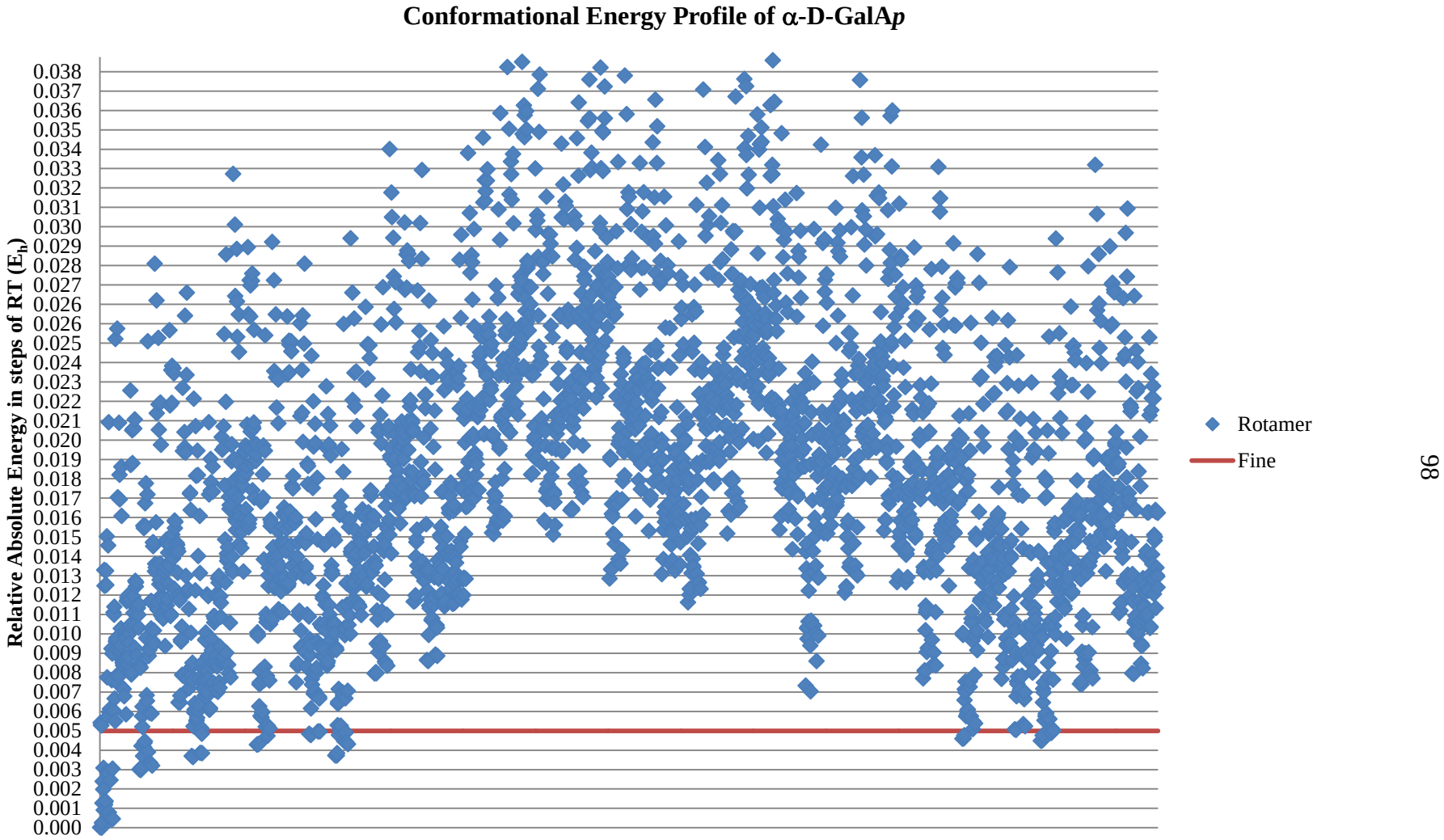


Figure 3.8 Conformational Energy Profile of α-D-GalA*p* modeled by bulk screening energies
(RT is approximately equal to 1mE_h)

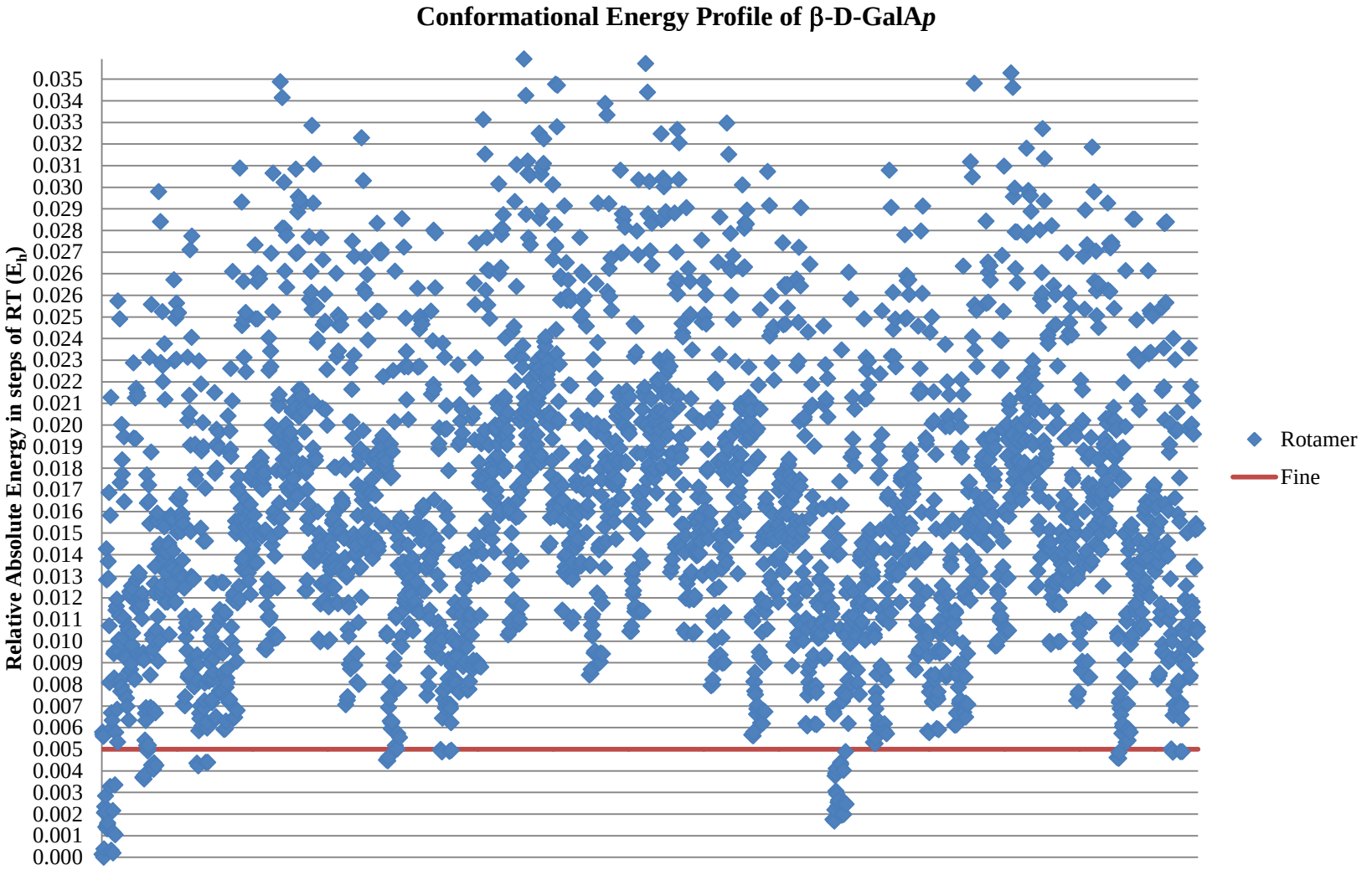


Figure 3.9 Conformational Energy Profile of β-D-GalA*p* modeled by bulk screening energies
(RT is approximately equal to 1mE$_h$)

Table 3.6 Relative Free Energies and Boltzmann-Averaged Populations (298 K) of α-D-Gal*p*A

α-D-galpA	G(kJ)	P(%)
0.0.0.0.0.0	0.0	3.1
0.0.0.0.180.180	0.9	2.2
0.0.120.0.0.0	-0.9	4.4
0.0.240.120.0.0	0.5	2.6
0.0.120.0.180.180	-0.7	4.2
0.240.0.0.0.0	-1.6	5.9
0.120.240.120.0.0	2.2	1.3
0.0.0.0.60.0	0.9	2.1
0.120.120.0.0.0	-0.1	3.3
240.240.0.0.0.0	2.2	1.3
240.120.120.0.0.0	0.1	3.0
0.0.120.0.60.0	3.7	0.7
0.240.0.0.60.0	0.1	3.0
	Total α Population	37.3

Table 3.7 Relative Free Energies and Boltzmann-Averaged Populations (298 K) of β-D-Gal*p*A

β-D-galpA	G(kJ)	P(%)
0.0.0.0.0.0	-2.2	7.7
0.0.0.0.180.180	-0.3	3.6
240.0.0.0.0.0	-1.0	4.6
240.0.0.0.180.180	-1.2	5.1
0.0.120.0.0.0	0.2	2.9
0.0.240.120.0.0	0.8	2.3
0.240.240.120.0.0	-0.8	4.4
0.240.120.0.0.0	-1.9	6.6
0.0.120.0.180.180	-1.4	5.5
240.240.240.120.0.0	-2.3	7.8
0.0.0.0.60.0	1.2	1.9
240.240.120.0.0.0	-2.6	8.9
240.0.0.0.60.0	2.2	1.3
	Total β Population	62.7

Conclusion

Because there are many possible sugar molecules present within plant biochemistry, the ability to study the conformational degrees of freedom for chemical structure with a reasonable level of *ab initio* theory is likely to be important for the development of improved models. The inherent conformational complexity of saccharides, however, has long presented an obstacle to this type of work. While the generation of all possible conformations manually is too tedious, CREPES provides an important automation tool to begin an investigation into highly variable systems. It provides a rather exhaustive survey of minimum energy structures and can help identify structures for study that may not be intuitively apparent. CREPES allows enough exploration of the conformational space to have confidence that all relevant conformers have been found.

As a test if the methodology, the α:β anomerization ratio for glucose was determined by Boltzmann population to be 39:61. This matches reasonably closely with the experimentally determined value of 36:64. Galacturonic acid's α:β anomerization ratio was found to be 37:63 However, no experimental data exists for correlation.

Initial studies with this conformer searching methodology were able to identify all conformers that had previously been reported for the most widely studied sugar, D-Glucose. The CREPES system is devised to utilize increasingly expensive computational methods as the number of low-energy conformers becomes better defined and therefore smaller. Because of this strategy, it is not surprising that calculations reported here for D-glucose (which we have undertaken as a calibration exercise) generally agree with this body of literature.

It is important to note that CREPES is a good starting point for a potential energy surface search not an endpoint. Some caveats persist; solvent models are restricted to continuum, and human judgment is required for selection criteria.

The main advantage of CREPES is that it provides the ability, to study less commonly modeled sugar systems. In the work presented here, galacturonic acid was surveyed using CREPES and 26 structures were identified as low-energy conformers. This type of computational data serves not only to enhance the possibility of investigating the biochemistry of these sugars in plant cell walls, but also may help inform subsequent models that would seek to describe polysaccharide chains involving unitized galacturonic acid units.

References

(1) Allinger, N. L.; Chen, K. H.; Lii, J. H.; Durkin, K. A. *J Comput Chem* **2003**, *24*, 1447-1472.

(2) Allinger, N. L.; Chen, K. S.; Katzenellenbogen, J. A.; Wilson, S. R.; Anstead, G. M. *J Comput Chem* **1996**, *17*, 747-755.

(3) Allinger, N. L.; Chen, K. S.; Lii, J. H. *J Comput Chem* **1996**, *17*, 642-668.

(4) Boutherin, B.; Mazeau, K.; Tvaroska, I. *Carbohyd Polym* **1997**, *32*, 255-266.

(5) Brooks, B. R.; III, C. L. B.; Jr, A. D. M.; Nilsson, L.; Petrella, R. J.; Roux, B.; Won, Y.; Archontis, G.; Bartels, C.; Boresch, S.; Caflisch, A.; Caves, L.; Cui, Q.; Dinner, A. R.; Feig, M.; Fischer, S.; Gao, J.; Hodoscek, M.; Im, W.; Kuczera, K.; Lazaridis, T.; Ma, J.; Ovchinnikov, V.; Paci, E.; Pastor, R. W.; Post, C. B.; Pu, J. Z.; Schaefer, M.; Tidor, B.; Venable, R. M.; Woodcock, H. L.; Wu, X.; Yang, W.; York, D. M.; Karplus, M. *J Comput Chem* **2009**, *30*, 1545-1614.

(6) Gouvion, C.; Mazeau, K.; Heyraud, A.; Taravel, F. R.; Tvaroska, I. *Carbohyd Res* **1994**, *261*, 187-202.

(7) Hatcher, E.; Guvench, O.; MacKerell, A. D. *J Phys Chem B* **2009**, *113*, 12466-12476.

(8) Lii, J. H.; Allinger, N. L. *J Phys Chem A* **2008**, *112*, 11903-11913.

(9) Lii, J. H.; Chen, K. H.; Allinger, N. L. *J Comput Chem* **2003**, *24*, 1504-1513.

(10) Lii, J. H.; Chen, K. H.; Allinger, N. L. *J Phys Chem A* **2004**, *108*, 3006-3015.

(11) Lii, J. H.; Chen, K. H.; Durkin, K. A.; Allinger, N. L. *J Comput Chem* **2003**, *24*, 1473-1489.

(12) Lii, J. H.; Chen, K. H.; Grindley, T. B.; Allinger, N. L. *J Comput Chem* **2003**, *24*, 1490-1503.

(13) Lii, J. H.; Chen, K. H.; Johnson, G. P.; French, A. D.; Allinger, N. L. *Abstr Pap Am Chem S* **2004**, *227*, U266-U266.

(14) Lii, J. H.; Chen, K. H.; Johnson, G. P.; French, A. D.; Allinger, N. L. *Carbohyd Res* **2005**, *340*, 853-862.

(15) Lii, J. H.; Ma, B. Y.; Allinger, N. L. *J Comput Chem* **1999**, *20*, 1593-1603.

(16) Ma, B. Y.; Schaefer, H. F.; Allinger, N. L. *J Am Chem Soc* **1998**, *120*, 3411-3422.

(17) Nevins, N.; Chen, K. S.; Allinger, N. L. *J Comput Chem* **1996**, *17*, 669-694.

(18) Nevins, N.; Lii, J. H.; Allinger, N. L. *J Comput Chem* **1996**, *17*, 695-729.

(19) Stortz, C. A.; Johnson, G. P.; French, A. D.; Csonka, G. I. *Carbohyd Res* **2009**, *344*, 2217-2228.

(20) Styron, S. D.; French, A. D.; Friedrich, J. D.; Lake, C. H.; Kiely, D. E. *Journal of Carbohydrate Chemistry* **2002**, *21*, 27-51.

(21) Vanommeslaeghe, K.; Hatcher, E.; Acharya, C.; Kundu, S.; Zhong, S.; Shim, J.; Darian, E.; Guvench, O.; Lopes, P.; Vorobyov, I.; MacKerell, A. D. *J Comput Chem* **2010**, *31*, 671-690.

(22) Xia, J. C.; Daly, R. P.; Chuang, F. C.; Parker, L.; Jensen, J. H.; Margulis, C. J. *J Chem Theory Comput* **2007**, *3*, 1620-1628.

(23) Xia, J. C.; Daly, R. P.; Chuang, F. C.; Parker, L.; Jensen, J. H.; Margulis, C. J. *J Chem Theory Comput* **2007**, *3*, 1629-1643.

(24) Yui, T.; Ogawa, K. *Trends Glycosci Glyc* **2005**, *17*, 159-176.

(25) Bouchemalchibani, N.; Braccini, I.; Derouet, C.; Dupenhoat, C. H.; Michon, V. *Int J Biol Macromol* **1995**, *17*, 177-182.

(26) Noto, R.; Martorana, V.; Bulone, D.; San Biagio, P. L. *Biomacromolecules* **2005**, *6*, 2555-2562.

(27) Hricovini, M.; Bystricky, S.; Malovikova, A. *Carbohyd Res* **1991**, *220*, 23-31.

(28) da Silva, C. O. *Theor Chem Acc* **2006**, *116*, 137-147.

(29) Barrows, S. E.; Dulles, F. J.; Cramer, C. J.; French, A. D.; Truhlar, D. G. *Carbohyd Res* **1995**, *276*, 219-251.

(30) Barrows, S. E.; Storer, J. W.; Cramer, C. J.; French, A. D.; Truhlar, D. G. *J Comput Chem* **1998**, *19*, 1111-1129.

(31) Corchado, J. C.; Sanchez, M. L.; Aguilar, M. A. *J Am Chem Soc* **2004**, *126*, 7311-7319.

(32) Cramer, C. J.; Truhlar, D. G. *J Am Chem Soc* **1993**, *115*, 5745-5753.

(33) Cramer, C. J.; Truhlar, D. G.; French, A. D. *Carbohyd Res* **1997**, *298*, 1-14.

(34) Brown, J. W.; Wladkowski, B. D. *J Am Chem Soc* **1996**, *118*, 1190-1193.

(35) Kirkpatrick, S.; Gelatt, C. D.; Vecchi, M. P. *Science* **1983**, *220*, 671-680.

(36) Metropolis, N.; Rosenbluth, A. W.; Rosenbluth, M. N.; Teller, A. H.; Teller, E. *J Chem Phys* **1953**, *21*, 1087-1092.

(37) Li, Z. Q.; Scheraga, H. A. *Proceedings of the National Academy of Sciences of the United States of America* **1987**, *84*, 6611-6615.

(38) Schmidt, M. W.; Baldridge, K. K.; Boatz, J. A.; Elbert, S. T.; Gordon, M. S.; Jensen, J. J.; Koeski, S.; Matsunaga, N.; Nguyen, K. A.; Su, S.; Windus, T. L.; Dupuis, M.; Montgomery, J. A. *J. Comput. Chem.* **1993**, *14*, 1347.

(39) Gordon, M. S.; Schmidt, M. W. In *Theory and Applications of Computational Chemistry, the First Fourty Years*; Dykstra, C. E., Frenking, G., Kim, K. S., Scuseria, G. E., Eds.; Elsevier: Amsterdam, 2005, p 1167-1189.

(40) Lamba, D.; Fabrizi, G.; Matsuhiro, B. *Acta Crystallogr C* **1994**, *50*, 1494-1497.

(41) Olson, R. M.; Bentz, J. L.; Kendall, R. A.; Schmidt, M. W.; Gordon, M. S. *J Chem Theory Comput* **2007**, *3*, 1312-1328.

(42) Bentz, J. L.; Olson, R. M.; Gordon, M. S.; Schmidt, M. W.; Kendall, R. A. *Comput Phys Commun* **2007**, *176*, 589-600.

(43) Piecuch, P.; Kucharski, S. A.; Kowalski, K.; Musial, M. *Comput Phys Commun* **2002**, *149*, 71-96.

(44) Piecuch, P.; Kowalski, K.; Pimienta, I. S. O.; Kucharski, S. A. *Acs Sym Ser* **2002**, *828*, 31-64.

(45) Pople, J. A.; Binkley, J. S.; Seeger, R. *Int J Quantum Chem* **1976**, 1-19.

(46) Frisch, M. J.; Headgordon, M.; Pople, J. A. *Chemical Physics Letters* **1990**, *166*, 275-280.

(47) Aikens, C. M.; Webb, S. P.; Bell, R. L.; Fletcher, G. D.; Schmidt, M. W.; Gordon, M. S. *Theor Chem Acc* **2003**, *110*, 233-253.

(48) Angyal, S. J. *Angewandte Chemie-International Edition* **1969**, *8*, 157-&.

(49) Nishida, Y.; Ohrui, H.; Meguro, H. *Tetrahedron Lett* **1984**, *25*, 1575-1578.

(50) Barnett, C. B.; Naidoo, K. J. *J Phys Chem B* **2008**, *112*, 15450-15459.

(51) Mason, P. E.; Neilson, G. W.; Enderby, J. E.; Saboungi, M. L.; Cuello, G.; Brady, J. W. *J Chem Phys* **2006**, *125*.

(52) Ramamoorthy, S.; Leppard, G. G. *J. Theor. Biol.* **1977**, *66*, 527-540.

APPENDIX

This appendix indicates what many of the more complex scripts utilized in CREPES and general GAMESS administration of large batch jobs do. While this list ignores much of the smaller daily sed scripts, it contains a list of most of the scripts that are used on a semi regular basis. Many scripts have usage information built in but not all. Many scripts are used in conjunction with xargs. Most helper scripts are used for consecutive actions on context and therefore must be referenced by script rather than being passed with the pipe (|) command. Most scripts rely on references to a list of the contents of a folder being passed that is formatted without the file extension. Scripts are listed in alphabetical order, with child files moved relevant to parent script The general form of this list is as follows.

Name: The name of the script

Usage: What any usage flags if any mean

Description: Brief description of what the script is used for

Script: The actual syntax of the scripts themselves. (bear in mind that most of these scripts were written to complete a particular required repetative task and were not written formally therefore many of the variables and methods are not elegant)

Name:

cat2files

Usage:

cat2files [file you wish to append to] [file you wish to append]

Description:

Appends two files File1+File2. This script is typically used in updating input decks with the coordinates found in a log file. It concatenates the two files and in a later formatting step the contents of the files are cleared of all but the pertinent information.

Script:

```
#!/bin/tcsh
switch ($#)
        case 0:
        echo "Usage: cat2files [file you wish to append to] [file you wish to append]"
        breaksw
        case 1:
        echo "Usage: cat2files [file you wish to append to] [file you wish to append]"
        breaksw
        case 2:
                cat $2 >> $1
        breaksw
        case *:
        echo "To many arguments"
        breaksw
endsw
```

Name:

catfiles

Usage:

catfiles $filelist $extension $groupToAdd

catfiles $1 $2 $3

$1 = file name containing list of files without extension
$2 = indicates extension you wish to add to input
$3 = indicates the group from the data file you wish to use

Description:

This tool that allows the appending of a $group from dat file to a batch of input files.

Script:

```
switch ($#)
        case 0:
        echo 'Usage: catfiles $1 $2 $3'
        echo '$1 = file name containg list of files without extension'
        echo '$2 = indicates extension you wish to add to input'
        echo '$3 indicates the grpup from the data file you wish to use'
        breaksw
        case 1:
        echo 'Usage: catfiles $1 $2 $3'
        echo '$1 = file name containg list of files without extension'
        echo '$2 = indicates extension you wish to add to input'
        echo '$3 indicates the grpup from the data file you wish to use'
        breaksw
        case 2:
        echo 'Usage: catfiles $1 $2 $3'
        echo '$1 = file name containg list of files without extension'
        echo '$2 = indicates extension you wish to add to input'
        echo '$3 indicates the grpup from the data file you wish to use'
        breaksw
        case 3:
more $1 | xargs -t -I {} cleangroups.py {} $2 $3
more $1 | xargs -t -I {} catfiles.2 "{}.inp" "{}.log" "{}.$2.$3.txt" "{}.$2.inp"
more $1 | xargs -t -I {} rm "{}.$2.$3.txt"
        breaksw
        case *:
        echo "To many arguments"
        breaksw
endsw
```

Name:

catfiles.2

Usage:

catfiles.2 File1 File2 File3 File4

Files 1-3 are files to be concatenated file 4 is the new file

Description:

This is a child script to catfiles. This is a functional file that simply concatenates the 3 files into a 4th

Script:

```
cat $1 $2 $3 >> $4
```

Name:

cleangroups.py

Usage:

arg1 = file name no extension
arg2 = extenstion to add
arg3 = dat file group to parse

Description:

Helper tool for catfiles. Pulls the appropriate group where many groups of the same name exist in the dat file

Script:

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

import sys

arg1 = sys.argv[1]
arg2 = sys.argv[2]
arg3 = sys.argv[3]

f = open(arg1+'.dat')

argcount = 0
for line in f:
        if '$'+arg3 in line:
                argcount = 1 + argcount
f = open (arg1+'.dat')
countgroups=0

lf = open(arg1+'.' + arg2+ '.' + arg3 +'.txt', 'a')
for line in f:
        if '$'+arg3 in line:
                countgroups = countgroups + 1
        if  countgroups==argcount:
                lf.write(line)
                if '$END' in line:
                        countgroups=0
```

Name:

cg

Usage:

cg filelist

filelist is a no extent list of files

Description:

Simple quick tool for removing files that flag GAMESS for errors on start (dat and log files).

User specific. Typically used to clean up after a failure of a batch job of some kind.

Script:

```
#!/bin/tcsh

switch ($#)
        case 0:
        echo "Usage: cg FileName"
        breaksw
        case 1:
        cat $1 |xargs -I {} rm "{}.log"
        cat $1 |xargs -I {} rm  "/home/verhaag/scr/{}.dat"
        cat $1 |xargs -I {} rm  "/home/verhaag/scr/{}.rst"
        rag
        breaksw
endsw
```

Name:

checktool

Usage:

checktool $1 $2 $3
$1 = input
$2 = output
$3 for verbose printing (v will do)

Description:

checktool identifies difference between input structure and a located structure. It returns Max movement (atom displacement of the atom that moved the most), mean movement (average displacement of all the atoms), max move atom (atom that moved the most's name) and file name again. This script is useful in identifying structures that have moved from the potential energy well that they are named for in CREPES.

Script:

```
#!/bin/tcsh

switch ($#)
        case 0 :
        echo 'Checktool Identifies difference between input structure and a located strucutre'
        echo 'It returns Max movement, mean movement, max move atom and file name again'
        echo 'Usage: checktool $1 $2 $3'
        echo '$1 = input'
        echo '$2 = output'
        echo '$3 for verbose printing (v will do)'
        breaksw
        case 1 :
        echo 'Checktool Identifies difference between input structure and a located strucutre'
        echo 'It returns Max movement, mean movement, max move atom and file name again'
        echo "Usage: checktool $1 $2"
        echo "$1 = input"
        echo "$2 = output"
        echo '$3 for verbose printing (v will do)'
        breaksw
        case 2:
                cp $1 asdf.out
                cat2files asdf.out $2

                sed  -i  -n  -e  '/^cp/p'  -e  '/^C1/,/END/p'  -e  '/LOCATED/,/INTERNUCLEAR/p'
asdf.out
                sed -i '/LOCATED/,/---/d' asdf.out
                sed -i 's/^$/C1/' asdf.out
                sed -i '/ANGS/d' asdf.out
                sed -i '/C1/d' asdf.out
```

```
                sed -i 's/cp //' asdf.out
                sed -i 's/.inp.*$//' asdf.out
# asdf formatting is no header $data group 1 $END name of file2 $datagroup 2 (again no
header) $END
                check.py

                rm asdf.out
        breaksw
        case 2:
                cp $1 asdf.out
                cat2files asdf.out $2

                sed -i -n -e '/^cp/p' -e '/^C1/,/END/p' -e '/LOCATED/,/INTERNUCLEAR/p'
asdf.out
                sed -i '/LOCATED/,/---/d' asdf.out
                sed -i 's/^$/C1/' asdf.out
                sed -i '/ANGS/d' asdf.out
                sed -i '/C1/d' asdf.out
                sed -i 's/cp //' asdf.out
                sed -i 's/.inp.*$//' asdf.out
# asdf formatting is no header $data group 1 $END name of file2 $datagroup 2 (again no
header) $END
                check.py v
                rm asdf.out
        breaksw
endsw
```

Name:

check.py

Usage:

N/A

Description:

Helper tool to checktool. Does the actual math involved, where checktool does formatting.

Script:

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

import math
import sys

data = open('asdf.out')


switch="input"
list = []
namecounter = 0
for line in data:
        if "END" in line:
                switch = "output"
                namecounter = 1
                continue
        if namecounter == 1:
                name = line.rstrip("\n")
                namecounter=0
                continue
        element = line.split()
        row = []
        row.append(element)
        row.append(switch)
        list.append(row)
#print list

input = []
output = []

for line in range(len(list)):

        if list[line][1] == 'input':
                input.append(list[line][0])
        else:
                output.append(list[line][0])

#print input
#print output

diff = []
for row in range(len(output)):
        currentrow = []
#       currentrow.append(output[row][0])
#       currentrow.append(output[row][1])
        for element in range(2,5):
                difference = float(output[row][element]) - float(input[row][element])
                currentrow.append(difference)
```

```
        diff.append(currentrow)
#print diff

vec=[]
for row in range(len(diff)):
        vec.append((diff[row][0] ** 2 + diff[row][1] ** 2 + diff[row][2] ** 2) ** .5)
#print vec

#to print complete list
if len(sys.argv) > 1:
        for atom in range(len(vec)):
                print str(round((vec[atom]),4)) + ",\t" + output[atom][0]
if vec == []:
        print "not located      \t" + name
else:
        number = 0
        for atom in range(len(vec)):
                if vec[atom] == max(vec):
                        number = atom
        print  str(round(max(vec),4))  +  ",\t"  +  str(round(sum(vec)/len(vec),4))    +  ",\t"  +
output[number][0] +str(number + 1) + ",\t" + name
```

Name:

cleanall

Usage:

cleanall $list $group $extension
cleanall $1 $2 $3
$1 = file list
$2 = Which group to keep from dat file eg. VEC (case sensitive)
$3 = group extension
$list = no file extension list of files to be operated on
$group = functional group to be added to input files
$extension= group extension

Description:

Performs same function as cleancatinp only on batch jobs

Script:

```
#!/bin/sh

switch ($#)
        case 0:
        echo 'Usage: cleanall $1 $2 $3'
        echo '$1 = file list'
        echo '$2 = Which group to keep from dat file eg. VEC (case sensitive)'
       echo '$3 = group extension'
        breaksw
        case 1:
        echo 'Usage: cleanall $1 $2 $3'
        echo '$1 = file list'
        echo '$2 = Which group to keep from dat file eg. VEC (case sensitive)'
       echo '$3 = group extension'
        breaksw
        case 2:
        echo 'Usage: cleanall $1 $2 $3'
        echo '$1 = file list'
        echo '$2 = Which group to keep from dat file eg. VEC (case sensitive)'
       echo '$3 = group extension'
        breaksw
        case 3:
#clean out the bulk of the log file and dat file
more  $1  |  xargs  -I  {}  sed  -n  -i  -e  '1,/C1/p'  -e  '/LOCATED/,/INTERNUCLEAR/p'  -e
'/$'$2'/,/$END/p' "{}.$3.inp"

#Clean out heading of the log
more $1 | xargs -I {} sed -i  '/LOCATED/,/---/d' "{}.$3.inp"

more $1 | xargs -I {} cleancatinp.2 "{}.$3.inp" fdsafdsatemp
        breaksw
        case *:
        echo "To many arguments"
        breaksw
endsw
```

Name:

cleancatinp

Usage:

cleancatinp $1 $2
$1 = file name
$2 = Which group to keep from dat file eg. VEC (case sensitive)(optional)

Description:

This file cleans the concatenated input files (typically from catfiles) for input

Script:

```
#!/bin/tcsh

switch ($#)
        case 0:
        echo 'Usage: cleancatinp $1 $2'
        echo '$1 = file name'
        echo '$2 = Which group to keep from dat file eg. VEC (case sensitive)(optional)'
        breaksw
        case 1:
#clean out the bulk of the log file and dat file
sed -n -i -e '1,/C1/p' -e '/LOCATED/,/INTERNUCLEAR/p' $1

#Clean out heading of the log
sed -i '/LOCATED/,/---/d' $1
cleancatinp.2 $1 fdsafdsatemp

        breaksw
        case 2:
#clean out the bulk of the log file and dat file
sed -n -i -e '1,/C1/p' -e '/LOCATED/,/INTERNUCLEAR/p' -e '/$'$2'/,/$END/p' $1

#Clean out heading of the log
sed -i '/LOCATED/,/---/d' $1

cleancatinp.2 $1 fdsafdsatemp
        breaksw
        case *:
        echo "To many arguments"
        breaksw
endsw
```

Name:

cleancatinp.2

Usage:

cleancatinp.2 $1 $2
$1 = input file
$2 = output file

Description:

Helper tool for cleancatinp. Clear some of the more problematic syntax that needs to be removed.

Script:

```
#!/bin/sh
sed '
/^$/{
N
/\n.*ANGS/ D
}' $1 > $2

sed '
/ANGS/ c\
\ $END
' $2 > $1
rm $2
```

Name:

CREPES

Usage:

See Discription

Description:

Welcome to CREPES
Complete Rotation for the Evaluation of the Potential Energy Surface

This tool is designed to automatically generate GAMESS input files that sample the potential energy surface (PES) based on rotational parameters supplied by the user.

Requirements

C++
Python

Components
---
readme.crepes
--General description of crepes

rotation.cpp
--C++ source code for the rotation code

crepes
--python script used for the execution of crepes

rotation
--A simple tool for using the rotation code to complete a rotation on 1 bond.

Installation

Extract to desired installation location.

If you are reading this you have probably already succeeded at this.

bunzip2 crepes.tar.bz2
tar -xvf crepes.tar

Change to the crepes directory (I will assume for the purpose of this documentation that the install path for crepes is as below)

cd /home/USER/crepes

Compile rotation.cpp to an app called crot
To do this
1. Identify your C++ compiler

There are a lot of C++ compilers out there.
The most common are
GNU, Intel, and Pathscale
To identify try the following

which g++
which icc
which pathCC

2. Based on which compiler you have, run the appropriate command.
g++ -o crot rotation.cpp
icc -o crot rotation.cpp
pathCC -o crot rotation.cpp

3. Add CREPES to your $PATH
Path is a *nix variable that tells the operating system where to look for its executables to save yourself the effort of typing out the entire path to crepes every time add the crepes path to your $PATH. This line is typically added to your .tcshrc .cshrc or .bashrc file in your home directory. To determine your current shell type.

echo $SHELL
or
ps -p $$ |tail -1

Based on you shell add the appropriate line
tcsh
set path = ("/home/USER/crepes" $path)
bash
path=$PATH:/home/USER/crepes
csh

setenv path $path:/home/USER/crepes

Usage

NOTICE
currently ^C (Ctl-C escape sequence) only works at questions
where CREPES is not asking for an integer
NOTICE

CREPES will ask for input, I have denoted input steps with
--
Below each input question I elaborate on what specifically the question is
asking for.

CREPES will generate a folder named for your inputfile called inputfile.output
In that folder will be created input files based on user input defined
during the running of CREPES.

Step 1

CREPES takes one argument, the name of the GAMESS input file.
All files need to end in .inp
If no arguments are supplied
--Enter a GAMESS input file:
Crepes is asking for the name of the file to be iterated upon for
rotation.

Step 2

In this step CREPES determines information about the axes of rotation.
--Please enter the number of rotatable groups:
CREPES is asking how many rotations total you are performing on your molecule.
Be careful CREPES currently does not have a confirmation step here.
CREPES will ask a series of question for each individual rotation.
CREPES will order output nomenclature based on the order that you define
your rotations.
Atom numbers are indicated by their order in the input deck
This order is typically easy to visualize by using another free GAMESS
tool, macmolplt

--Enter atom number of rotation point 1 for rotation 1

This question is asking for the atom number of the first point on the line about which you would like to rotate your group of atoms. This atom does not move for this rotation but can move in another rotation.

--Enter atom number of rotation point 2 for rotation 1:
This question is asking for the atom number of the second point on the line about which you would like to rotate your group of atoms. This atom does not move for this rotation but can move in another rotation.

--Number of atoms moving by rotation 1:
This question is asking how many atoms are moving with a particular rotation not which atoms. It does not include the atoms you are rotating about. ie An alcohol will only have 1 atom moving.

--Total rotation (in degrees) 1 (360) :
This question is asking how many degrees your total rotation will be.
It has a default of 360 degrees.
It also has a notice.
For rotations less than 360 add one step size to total rotation.
For nomenclature reason an extra step is required,
ie You would like to do a rotation of 180 degrees in 30 degree steps.
You would enter 210 in total rotation.

--Step size (in degrees) of rotation 1 (120) :
This question is asking the increment of degrees you would like each Step
Default step size is 120 degrees
ie step of 120 in a 360 total rotation will generate structures for
0,120,240

These questions will be repeated for each of the number of rotations you indicated in question 1

Step 3

CREPES will now ask question about the atoms that are moving.

--Enter the atoms that move for rotation 1 (comma separated):
This is asking for a list of atom numbers that are moving.
This does not include the atoms that you are rotating about.
input should look something like 3,4,5

This process will repeat for each rotation.

Step 4

After all data is input about the rotations CREPES will display the input data.
Carefully look over the defined rotations.
If there is an error indicate that they are not all correct
CREPES will ask which rotation needs to be corrected
you can only correct one at a time.

CREPES will then generate the input decks.
Output names are formatted in the following way

sample.inp
sample.output/
sample.rot1.rot2.etc.inp

looking from rotation point 2 towards point 1 rotations occur in a
clockwise direction. if you wish to rotate in the opposite direction
input your rotation points in the opposite order.

History

This project started out as a simple tool to help me save some time
when I needed to do a rotational analysis of a functional group on a
given molecule. As part of my research we started looking at
carbohydrates. I attempted to use molecular mechanics(MM) codes to help
identify global minima. I found that the MM codes that were available were
not very rigorous in their sampling of the PES. I was lucky enough to
have a large amount of resources available. So it occurred to me a better
method for identifying global minima would be to use a small basis set
and generate a complete set of conformers. Thus, came the creation of
CREPES. This shows where CREPES is strongest; in environments with high
resource availability and with molecules with high number of degrees of
freedom. I tried to make CREPES as general as possible so that it would
be functional for as many other people as possible.

Wish list

Some things that were not added but could be relatively easily to improve
performance or functionality
--add escape sequence from integer questions
currently ^C only works at questions where CREPES is not asking

for an integer.

--re-write C++ code in python

there should be no reason why python could not handle the transforms
this would reduce requirements

--Find a way to make non-compound rotations more efficient.

currently every rotation is recalculated every time it is called
many rotations are the same every time they are calculated
if they could be stored in memory it would reduce times that
the C++ would have to be called

Script:

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

# Require imports
import sys
import os
import shutil
import tempfile
#global variables
rotationArray = []
movingAtoms = []
numrotablegroups = 0
edititem=''
inpfilename=''
headerfile=''
datafile=''
tailfile=''
dataArray=[0]
inputdirectory=''
outputdirectory=''
crepespath = sys.path[0]

def usage():
        print "Usage: crepes []"
        print "                          gamess input file"
        print "                          input files must end in inp"
def inpfiletest(argv):
        global inpfile
        global inpfilename
        if len(sys.argv) == 1:
                inpfilename = raw_input('Enter a GAMESS input file: ')
        else:
                inpfilename = sys.argv[1]
        if  inpfilename.endswith("inp"):
                return
        else:
                print "This is not a games input file."
                sys.exit()
def fillrotationArray(edititem):
        global rotationArray
        #test if this is a new rotation or and edit of a misentered
        if edititem != 0:
                rotation = edititem - 1
                fillrotationquest(rotation)
        else:
                for rotation in range(numrotablegroups):
                        fillrotationquest(rotation)
def fillrotationquest(rotation):
                        global edititem
                        print "----"
```

```
#       Debug Prints
#                       print "edititem = " + str(edititem)
#                       print "rotation = " + str(rotation)

# Expand array for new rotations
                        if edititem == 0:
                                rotationArray.extend([0])
                        rotationArray[rotation] = [0 , 0, 0, 0, 0, 0]
# Fill the rotation array
                        question = "Enter atom number of rotation point 1   for rotation " +
str(rotation+1) + ": "
                        inputcheck = 0
                        while inputcheck==0:
                                try:
                                        rotationArray[rotation][0] =  int(raw_input(question))
                                        inputcheck = 1
                                except:
                                        print "This is not an integer"
                        question = "Enter atom number of rotation point 2   for rotation " +
str(rotation+1) + ": "
                        inputcheck = 0
                        while inputcheck==0:
                                try:
                                        rotationArray[rotation][1] =  int(raw_input(question))
                                        inputcheck = 1
                                except:
                                        print "This is not an integer"
                        question = "Number of atoms moving by rotation " + str(rotation+1) + ":
"
                        inputcheck = 0
                        while inputcheck==0:
                                try:
                                        rotationArray[rotation][2] =  int(raw_input(question))
                                        inputcheck = 1
                                except:
                                        print "This is not an integer"
                        stepcheck = 0
                        while stepcheck == 0:
                                print 'For rotations less than 360 add one step size to total
rotation'
                                question = "Total rotation (in degrees) " + str(rotation+1) + "
(360) : "
                                inputcheck = 0
                                stepsize =  raw_input(question)
                                if stepsize == "":
                                        rotationArray[rotation][3] = 360
                                else:
                                        while inputcheck==0:
                                                try:
                                                        rotationArray[rotation][3]              =
int(stepsize)
                                                        inputcheck = 1
                                                except:
                                                        print "This is not an Number"
                                question   =   "Step   size   (in   degrees)   of   rotation   "   +
str(rotation+1) + " (120) : "
                                inputcheck = 0
                                stepsize =  raw_input(question)
                                if stepsize == "":
                                        rotationArray[rotation][4] = 120
                                else:
                                        while inputcheck==0:
                                                try:
                                                        rotationArray[rotation][4]              =
int(stepsize)
                                                        inputcheck = 1
                                                except:
                                                        print "This is not an Number"

# Check Input for logic error
```

```
                                if rotationArray[rotation][4] < rotationArray[rotation][3]:
                                        stepcheck = 1
                                else:
                                        print 'Rotation step is larger than the total rotation.'


# Tag for compound rotations
# Originaly intended to make non compound rotations more efficient by storing rather than
# calculating each step. In practice found that calculation was not a signifcant impediment
#  therfore setting to denote all as compound in spite of there being no functionality
implemented
#                       question = "Is rotation " + str(rotation+1) + " a compound rotation
(Y/n): "
#                       inputcheck = 0
#                       while inputcheck==0:
#                               compound = raw_input(question)
#                               try:
#                                       rotationArray[rotation][5] =  bool(eval(compound))
#                                       inputcheck = 1
#                               except:
#                                       if compound == "":
#                                               rotationArray[rotation][5] = bool(1)
#                                               inputcheck = 1
#                                       elif  compound == "y":
#                                               rotationArray[rotation][5] = bool(1)
#                                               inputcheck = 1
#                                       elif compound == "n":
#                                               rotationArray[rotation][5] = bool(0)
#                                               inputcheck = 1
#                                       else:
#                                               print "This is not a Boolean"
                                rotationArray[rotation][5] = bool(1)
def fillmovingatom(edititem):
# Check if this is a new rotation or a edit of misentered data
        if edititem != 0:
                rotation = edititem - 1
                fillmovingquest(rotation)
        else:
                for rotation in range(numrotablegroups):
                        fillmovingquest(rotation)
def fillmovingquest(rotation):
# fill the moving atoms array
        print "----"
        global movingAtoms
        movingAtoms.extend([0])
        movingAtoms[rotation]=[]
        for atom in range(rotationArray[rotation][2]):
                movingAtoms[rotation].extend([0])
        question = "Enter the atoms that move for rotation " + str(rotation+1) + " (comma
separated): "
        inputcheck = 0
        while inputcheck==0:
# test to see if if the number of atoms moving matches how many the entered
                try:
                        rotablelist = list(eval("(" + raw_input(question) + ",)"))
                        if  len(rotablelist) == rotationArray[rotation][2]:
                                try:
                                        movingAtoms[rotation] =  rotablelist
                                        movingAtoms[rotation].sort()
                                        inputcheck = 1
                                except:
                                        print "This is not a list"
                        else:
                                confimmovables     =     raw_input('You     entered     '     +
str(rotationArray[rotation][2]) + ' for the number of atoms that need to be moved for this
rotation;\n You entered a different number of atoms.\n Is this list correct? (y/n) ')
                                if confimmovables == 'y':
                                        movingAtoms[rotation] = rotablelist
                                        rotationArray[rotation][2] = len(movingAtoms[rotation])
```

```
                                        inputcheck = 1
                except:
                        print "This is not a list"
def inputquestions():
#initiate data acquisition edititem=0 indicates a new rotation
        inputcheck=0
        global numrotablegroups
        global edititem
        while inputcheck == 0:
                try:
                        numrotablegroups = int(raw_input('Please enter the number of rotable
groups: '))
                        inputcheck = 1
                except:
                        print "This is not an integer"
        edititem=0
        fillrotationArray(edititem)
        fillmovingatom(edititem)
def confirminput():
# Print the inputed data for review before executing
        for rotation in range(numrotablegroups):
                print '---'
                print 'Rotation ' + str(rotation+1)
                print 'Axix of rotation: ' + str(rotationArray[rotation][0]) + " , " +
str(rotationArray[rotation][1])
                print 'Number of atoms moving with rotation: ' +
str(rotationArray[rotation][2])
                print 'Angle of rotation total/steps: ' + str(rotationArray[rotation][3]) + "
/ " + str(rotationArray[rotation][4])
                print 'Compound rotation: ' + str(rotationArray[rotation][5])
                print 'Atoms that move for rotation: ' + str(movingAtoms[rotation])
def editinput():
# after displaying data allow user to indicate changes that need to be made
        print "---"
        confirmed = raw_input( "Are all of these correct (Y/n): ")
        if confirmed == '':
                confirmed = 'y'
        if confirmed != 'y':
                question='Enter the rotation that needs to be corrected [1-'
+str(numrotablegroups) + ']: '
                inputcheck = 0
                while inputcheck==0:
                        global edititem
                        edititem = raw_input(question)
                        try:
                                edititem = int(edititem)
                                inputcheck=1
                        except:
                                print "This is not an integer"
                print "Correcting " + str(edititem)
                fillrotationArray(edititem)
                fillmovingatom(edititem)
                confirminput()
                editinput()
def parseinpfile():
# Parse input deck so that the file is also broken up so that it can be reassembles after
rotations
        pathisclear = 0
        while pathisclear == 0:
                global inpfilename
                global inpfile
                print 'Reading ' + inpfilename
                inpfile = open(inpfilename)
                global headerfile
                global datafile
                global tailfile
                global inputdirectory
                inputdirectory = inpfilename.rstrip('.inp') + '.input/'
                if not os.path.exists(inputdirectory):
```

```
                        os.mkdir(inputdirectory)
                        os.chdir(inputdirectory)
                        headerfile = open('headerfile.txt', 'w')
                        datafile = open('datafile.txt','w')
                        tailfile = open('tailfile.txt','w')
                        sym = 0
                        for line in inpfile:
                                if sym == 0:
                                        headerfile.write(line)
                                if 'C1' in line:
                                        sym=1
                                if '$END' in line and sym == 1:
                                        sym =  2
                                if sym == 1 and 'C1' not in line:
                                        datafile.write(line)
                                if sym == 2:
                                        tailfile.write(line)
                        print "Parse complete. "
                        headerfile.close()
                        datafile.close()
                        tailfile.close()
                        os.chdir('..')
                        pathisclear = 1
                else:
                        print "Directory " + inputdirectory + " already exits."
                        overwrite = raw_input("Overwrite(Y/n):")
                        if overwrite == 'y':
                                shutil.rmtree(inputdirectory,ignore_errors=True)
                        elif overwrite=="":
                                shutil.rmtree(inputdirectory,ignore_errors=True)
                        else:
                                sys.exit()
def copyInputFile():
# Prepare a place for generation of output files and copy data over
        global inpfilename
        global inputdirectory
        global outputdirectory
        pathisclear = 0
        while pathisclear == 0:
                global inpfilename
                global outputdirectory
                outputdirectory = inpfilename.rstrip('.inp') + '.output/'
# Check if path already exists and query user for what to do
                if not os.path.exists(outputdirectory):
#       Debug Prints
#                       print os.getcwd()
                        os.mkdir(outputdirectory)
                        instr= inputdirectory + 'datafile.txt'
                        outstr = outputdirectory + 'datafile.txt'
                        shutil.copy( instr, outputdirectory)
                        pathisclear = 1
                else:
                        print "Directory " + outputdirectory + " already exits."
                        overwrite = raw_input("Overwrite(Y/n):")
                        if overwrite == 'y':
                                shutil.rmtree(outputdirectory,ignore_errors=True)
                        elif overwrite=="":
                                shutil.rmtree(outputdirectory,ignore_errors=True)
                        else:
                                sys.exit()
        return
def createdataArray():
# create an array of the incoming data
        global dataArray
        datafile = open(inputdirectory + 'datafile.txt','r')
        for line in datafile:
                dataArray.extend([line.rstrip("\n")])
        datafile.close()
def formateddatafile():
```

```
# format input data so that it can properly be handled by C++ code
        os.chdir(outputdirectory)
        formateddatafile = open('formateddatafile.txt','w')
        datafile = open('datafile.txt','r')
        for line in datafile:
#       Debug Prints
#               print line.rstrip('\n')
                splitline = line.split()
                for element in range(len(splitline)):
                        splitline[element] = splitline[element] + ","
                string = ''.join(splitline)
                string = string.replace('.0,','.0\t')
                string = string.rstrip(",")
#       Debug Prints
#               print string
                formateddatafile.write(string+'\n')
        formateddatafile.close()
        os.chdir("..")
def iterator( rotationParameters ):
# Itterate through all possible combinations of rotations and generate files
        totalIndexLength = len(rotationParameters.rotationArray)
        # open the current file  and load contents into memory for use later
        currentdataArray = []
        currentDataFile = open(rotationParameters.currentFileNameStr,'r')
        for line in currentDataFile:
                currentdataArray.extend([line.rstrip("\n")])
        currentDataFile.close()
        # perform my rotations
        # set up some variables for driving the for loop based off our first input data
element
#       Debug Prints
#       print "rotationParameters.currentIndex " + str(rotationParameters.currentIndex)
        if rotationParameters.currentIndex >= totalIndexLength:
                return
        firstPropertiesElement                                                          =
rotationParameters.rotationArray[rotationParameters.currentIndex]
#       Debug Prints
#       print 'Testing syntax'
#       print firstPropertiesElement
#       print rotationArray
#       print movingAtoms
        startingDegree = 0
        endingDegree = firstPropertiesElement[3]
        stepDegree = firstPropertiesElement[4]
#       Debug Prints
#       print "startingDegree " + str(startingDegree)
#       print "stepDegree " + str(stepDegree)
#       print "endingDegree " + str(endingDegree)
        # loop through for my for loop
        degreeCounter = startingDegree
        while degreeCounter < endingDegree:
                # what information do we need to rotate; pass it into the rotateAtoms call
                localRotationParameters = rotationParameters.clone()
                localRotationParameters.currentDegree = degreeCounter
                rotationResultdataArray        =        rotateAtoms(        currentdataArray,
localRotationParameters )
#       Debug Prints
#               print "rotationResultdataArray", rotationResultdataArray
#               for line in rotationResultdataArray:
#                       print line
                # get a new file name
                oldFileNameStr = rotationParameters.currentFileNameStr.rstrip('.txt')
                newFileNameStr = oldFileNameStr + "." + str(degreeCounter) + ".txt"
                # loop through the rotationresult and write it out to the new file
                #  save  the  rotations    to  a  new  data  file  based  off  the  rotations:
"datafile.0.txt"
#               print currentdataArray
                print "Writing " + str(newFileNameStr)
                newDataFile = open( newFileNameStr, 'w' )
```

```
#       Debug Prints
#               print "currentdataArray "
#               linecounter=0
#               for line in currentdataArray:
#                       linecounter=linecounter+1
#                       print linecounter,line
#               print "rotationResultdataArray "
#               for line in rotationResultdataArray:
#                       print line
#               print "rotationParameters.currentIndex ", rotationParameters.currentIndex
                rotatedatomcounter=2
                for atom in movingAtoms[rotationParameters.currentIndex]:
                        if rotatedatomcounter < len(rotationResultdataArray):
#       Debug Prints
#                               print "atom ", atom
#                               print "rotatedatomcounter ", rotatedatomcounter
#                               print       "rotationResultdataArray[rotatedatomcounter]       ",
rotationResultdataArray[rotatedatomcounter]
                                currentdataArray[atom-
1]=rotationResultdataArray[rotatedatomcounter]
                                rotatedatomcounter = rotatedatomcounter + 1
                        else:
                                break
                for line in range(len(currentdataArray)):
                        currentdataArray[line]         =         currentdataArray[line].replace('
',',',2)
                        currentdataArray[line] = currentdataArray[line].replace('       ',',')
                        currentdataArray[line] = currentdataArray[line].replace(',,'       ',1)
                        currentdataArray[line] = currentdataArray[line].replace('.0,','.0\t',1)
#       Debug Prints
#               print "currentdataArray after replace "
#               linecounter=0
#               for line in currentdataArray:
#                       linecounter=linecounter+1
#                       print linecounter,line
                for atom in currentdataArray:
                        # write to the file
                        newDataFile.write(atom + "\n")
                newDataFile.close()

                # check to see if we should recurse; stopping condition
#       Debug Prints
#               print "totalIndexLength " + str(totalIndexLength)
#               print             "rotationParameters.currentIndex             "             +
str(rotationParameters.currentIndex)
                if rotationParameters.currentIndex < totalIndexLength:
                        newRotationParameters = rotationParameters.clone()
                        newRotationParameters.currentIndex = rotationParameters.currentIndex +1
                        newRotationParameters.currentDegree = degreeCounter
                        newRotationParameters.currentFileNameStr = newFileNameStr
                        iterator( newRotationParameters )
                #increment degrees based on propertries of input
                degreeCounter = degreeCounter + stepDegree
def rotateAtoms( dataArray, rotationParameters):
        returndataArray = dataArray
        # do the rotation of the atoms, whatever that is, need to identify what information we
need to do the rotation and call the c execution
        # takes the input dataArray and copies it/modifies it and fills the return Data Array
so it can be written to disk
        #index
        rotateIndex = rotationParameters.currentIndex
        #angle
        rotateCurrentDegree = rotationParameters.currentDegree
        #rotationArray values
        rotateArrayValues = rotationParameters.rotationArray[rotateIndex]
        #moving atoms
        rotateMovingAtoms = rotationParameters.movingAtoms[rotateIndex]
        #step size
        rotatestepsize= rotationParameters.rotationArray[rotateIndex][4]
```

```
#       Debug Prints
#       print rotateArrayValues[0]
#       print rotateArrayValues[1]
#       print dataArray[rotateArrayValues[0]-1]
#       print dataArray[rotateArrayValues[1]-1]
#       print dataArray
#       print currentdataArray
#       print range(len(dataArray))
        sourceData = tempfile.NamedTemporaryFile()
        sourceData.write(dataArray[rotateArrayValues[0]-1]+"\n")
        sourceData.write(dataArray[rotateArrayValues[1]-1]+"\n")
        for line in range(len(dataArray)):
                if line + 1 in rotateMovingAtoms:
#                       print dataArray[line]
                        sourceData.write(dataArray[line]+"\n")
        sourceData.seek(0)
#       Debug Prints
#       print ""
#       print "tempfile"
#       print sourceData.read()
#       sourceData.seek(0)
        #atoms to rotate around
        # look up in atoms data based on ids in moving attoms and rotation array
        returndataArray = rotateAtom(rotateMovingAtoms,  rotatestepsize, rotateCurrentDegree,
sourceData.name)
#       Debug Prints
#       print"returned data"
#       for atom in returndataArray:
#               print atom.rstrip("\n")
        return returndataArray
def rotateAtom(rotateMovingAtoms,  rotatestepsize, rotateCurrentDegree, sourceData):
# Call the rotate atom C++ code
        returnRotation = []
#       sourceData = 'formateddatafile.txt'
#       Debug Prints
#       print 'str(len(rotateMovingAtoms))' + str(len(rotateMovingAtoms))
#       print 'str(rotateCurrentDegree)' + str(rotateCurrentDegree)
#       print "str(sourceData)" + str(sourceData)
        # the rotator imports previous step coordinates therefore it gets passed the step size
        if rotateCurrentDegree == 0:
#               print  "executing  ","./crot  "  +  str(len(rotateMovingAtoms))  +  "  "  +
str(rotateCurrentDegree) + " " + str(sourceData)
                tempstore = os.popen(str(crepespath)+"/crot " + str(len(rotateMovingAtoms)) +
" " + str(rotateCurrentDegree) + " " + str(sourceData), 'r')
        else:
#               print  "executing  ","./crot  "  +  str(len(rotateMovingAtoms))  +  "  "  +
str(rotatestepsize) + " " + str(sourceData)
                tempstore = os.popen(str(crepespath)+"/crot " + str(len(rotateMovingAtoms)) +
" " + str(rotatestepsize) + " " + str(sourceData), 'r')
        for line in tempstore:
                returnRotation.append(line.rstrip('\n'))
#       for line in returnRotation:
#               print line.rstrip('\n')
        return returnRotation
class RotationParameters:
# define a class object that passes information up and down the itteration
        def  __init__  (self,  currentIndex  =  0,  rotationArray  =  None,  movingAtoms  =  None,
currentFileNameStr = None, currentDegree = None ):
                self.currentIndex = currentIndex
                self.rotationArray = rotationArray
                self.movingAtoms = movingAtoms
                self.currentFileNameStr = currentFileNameStr
                self.currentDegree = currentDegree
        def clone( self):
                returnParameters = RotationParameters()
                returnParameters.currentIndex = self.currentIndex
                returnParameters.rotationArray = self.rotationArray
                returnParameters.movingAtoms = self.movingAtoms
                returnParameters.currentFileNameStr = self.currentFileNameStr
```

```
                returnParameters.currentDegree = self.currentDegree
                return returnParameters
def fileformat():
# format the files that are output from the C++ code
# recompile the input deck from the headers of the original file
        basename = inpfilename.rstrip('inp')
        for file in os.listdir(outputdirectory):
                syntaxname = file.lstrip("formateddatafile.")
                dataname = syntaxname.rstrip('txt')
                filename = basename + dataname + 'inp'
                shutil.copy(inputdirectory + 'headerfile.txt' , outputdirectory + filename)
                print "Formating ", filename
                openfile = open(outputdirectory + filename, 'a')
                opendatafile = open(outputdirectory + file,'r')
                for line in opendatafile:
                        formline = line
                        formline = formline.replace('\t',',')
                        formline = formline.replace(',',',  ')
                        formline = formline.replace('  -',' -')
                        formline =  formline.replace(',','\t')
                        openfile.write(formline)
                tailfile = open(inputdirectory + 'tailfile.txt', 'r')
                for line in tailfile:
                        openfile.write(line)
def cleanup():
#clean out unnecessary files from output that will cause problems in the format step
        try:
                os.remove(outputdirectory + 'datafile.txt')
        except:
                print 'File datafile.txt does not exist'
        try:
                os.remove(outputdirectory + 'formateddatafile.txt')
        except:
                print 'File formateddatafile.txt does not exist'
        maxseps=0
        for file in os.listdir(outputdirectory):
                seps = file.count('.')
                if seps > maxseps:
                        maxseps = seps
        for file in os.listdir(outputdirectory):
                seps = file.count('.')
                if seps < maxseps:
#       Debug Prints
#                       print file,seps,maxseps
                        os.remove(outputdirectory+file)
def finalcleanup():
#Clear files that are not used by the user may be helpful in testing to not call this
function
        print "Cleaning up"
        for file in os.listdir(outputdirectory):
                if "txt" in file:
                        try:
                                os.remove(outputdirectory + file)
                        except:
                                print "File " + file + "Does not exist"
        shutil.rmtree(inputdirectory)
def main(argv):
# Main thread of the crepes function calls teh individual steps
        print '\n CREPES \n'
        inpfiletest(argv)
        inputquestions()
        confirminput()
        editinput()
        parseinpfile()
        copyInputFile()
        createdataArray()
        formateddatafile()
        # setup the parameters for the rotations
        newBaseFile = outputdirectory + "formateddatafile.txt"
```

```
        myRotationParameters = RotationParameters()
        myRotationParameters.currentIndex = 0
        myRotationParameters.rotationArray = rotationArray
        myRotationParameters.movingAtoms = movingAtoms
        myRotationParameters.currentFileNameStr = newBaseFile
        # run the rotations
        iterator( myRotationParameters )
        cleanup()
        fileformat()
#       Debug Prints
#       print rotationArray
#       print movingAtoms
        finalcleanup()

main(sys.argv)
```

Name:

ctepes

Usage:

ctepes $inputfile

Description:

Operation is similar to CREPES. Follow on screen instructions. This script has no robustness built in.

Script:

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

# Require imports
import sys
import os
import shutil
import tempfile
import copy
#global variables
edititem=''
inpfilename=''
headerfile=''
datafile=''
tailfile=''
inputdirectory=''
outputdirectory=''
transpath = sys.path[0]
#array of info for each moving atom
movatom=[]
dataArray=[]

def usage():
        print "Usage: trans []"
        print "                          gamess input file"
        print "                          input files must end in inp"
def inpfiletest(argv):
        global inpfile
        global inpfilename
        if len(sys.argv) == 1:
                inpfilename = raw_input('Enter a GAMESS input file: ')
        else:
                inpfilename = sys.argv[1]
        if  inpfilename.endswith("inp"):
                return
        else:
                print "This is not a games input file."
                sys.exit()
def queryuser():
        global movatom
        question = "Enter the number of moving groups: "
        numbermovinggroups=int(raw_input(question))
        for atom in range(numbermovinggroups):
                question = "Enter the atom number for group " + str(atom+1) +": "
                line = int(raw_input(question))
                question = "Enter the number of steps for group " + str(atom+1) +": "
```

```
                steps = int(raw_input(question))
                question = "Enter step size for group " + str(atom+1) +": "
                stepsize = float(raw_input(question))
                movatom.append([line, steps, stepsize])
#       print movatom
def parseinpfile():
# Parse input deck so that the file is also broken up so that it can be reassembles after
rotations
        pathisclear = 0
        while pathisclear == 0:
                global inpfilename
                global inpfile
                print 'Reading ' + inpfilename
                inpfile = open(inpfilename)
                global headerfile
                global datafile
                global tailfile
                global inputdirectory
                inputdirectory = inpfilename.rstrip('.inp') + '.input/'
                if not os.path.exists(inputdirectory):
                        os.mkdir(inputdirectory)
                        os.chdir(inputdirectory)
                        headerfile = open('headerfile.txt', 'w')
                        datafile = open('datafile.txt','w')
                        tailfile = open('tailfile.txt','w')
                        sym = 0
                        for line in inpfile:
                                if sym == 0:
                                        headerfile.write(line)
                                if 'C1' in line:
                                        sym=1
                                if '$END' in line and sym == 1:
                                        sym =  2
                                if sym == 1 and 'C1' not in line:
                                        datafile.write(line)
                                if sym == 2:
                                        tailfile.write(line)
                        print "Parse complete. "
                        headerfile.close()
                        datafile.close()
                        tailfile.close()
                        os.chdir('..')
                        pathisclear = 1
                else:
                        print "Directory " + inputdirectory + " already exits."
                        overwrite = raw_input("Overwrite(Y/n):")
                        if overwrite == 'y':
                                shutil.rmtree(inputdirectory,ignore_errors=True)
                        elif overwrite=="":
                                shutil.rmtree(inputdirectory,ignore_errors=True)
                        else:
                                sys.exit()
def copyInputFile():
# Prepare a place for generation of output files and copy data over
        global inpfilename
        global inputdirectory
        global outputdirectory
        pathisclear = 0
        while pathisclear == 0:
                global inpfilename
                global outputdirectory
                outputdirectory = inpfilename.rstrip('.inp') + '.output/'
# Check if path already exists and query user for what to do
                if not os.path.exists(outputdirectory):
#       Debug Prints
#                       print os.getcwd()
                        os.mkdir(outputdirectory)
                        instr= inputdirectory + 'datafile.txt'
                        outstr = outputdirectory + 'datafile.txt'
```

```
                        shutil.copy( instr, outputdirectory)
                        pathisclear = 1
                else:
                        print "Directory " + outputdirectory + " already exits."
                        overwrite = raw_input("Overwrite(Y/n):")
                        if overwrite == 'y':
                                shutil.rmtree(outputdirectory,ignore_errors=True)
                        elif overwrite=="":
                                shutil.rmtree(outputdirectory,ignore_errors=True)
                        else:
                                sys.exit()
        return
def createdataArray():
# create an array of the incoming data
        global dataArray
        datafile = open(inputdirectory + 'datafile.txt','r')
        dataArray =    []
        for line in datafile:
                element = line.split()
                dataArray.append(element)
def formateddatafile():
# format input data so that it can properly be handled by C++ code
        os.chdir(outputdirectory)
        formateddatafile = open('formateddatafile.txt','w')
        for line in dataArray:
#               print line
                linestring = ''
                for element in line:
                        linestring = linestring + "\t" + str(element)
#                       print linestring
                formateddatafile.write(linestring + "\n")

        formateddatafile.close()
        os.chdir("..")
def iterator(translationparameters):
        global dataArray

        totalindexlength = len(translationparameters.movatom)
#       print "totalindexlength " + str(totalindexlength)
#       Check to see if we have reached maximum depth
        if translationparameters.curentIndex >= totalindexlength:
                return
#       pull parameters for current step
        firstPropertiesElement                                                  =
translationparameters.movatom[translationparameters.curentIndex]
        startingStep = 0
        endingStep = firstPropertiesElement[1]
        stepsize = firstPropertiesElement[2]

        currentdataArray = []
        currentDataFile = open(translationparameters.currentfilenamestr,'r')
        for line in currentDataFile:
                element = line.split()
                currentdataArray.append(element)
        currentDataFile.close()

        stepCounter = startingStep
        while stepCounter < endingStep:
                localTranslationParameters = translationparameters.clone()
                localTranslationParameters.currentstep = stepCounter
                translationResultsDataArray           =           translate(currentdataArray,
localTranslationParameters)


                namingcurrentMovingAtom                                                 =
translationparameters.movatom[translationparameters.curentIndex][0]
                namingoriginPosition=dataArray[namingcurrentMovingAtom-1][2]
                nametag= round(float(namingoriginPosition) + stepCounter*stepsize,3)
                oldFileNameStr = translationparameters.currentfilenamestr.strip('.txt')
```

```
                newFileNameStr = oldFileNameStr + "-" + str(nametag) + ".txt"
                print "Writing " + str(newFileNameStr)
                newDataFile = open( newFileNameStr, 'w' )
                #print dataArray

                currentdataArray = copy.deepcopy(translationResultsDataArray)

                for line in currentdataArray:
                        printline = ''
                        #print line
                        for element in line:
                                #print element
                                printline = printline  + str(element) + '   '
                        #print printline
                        newDataFile.write(printline + "\n")
                newDataFile.close()
                #print            "translationparameters.curentIndex            "            +
str(translationparameters.curentIndex)
        #recurrsor
                if translationparameters.curentIndex < totalindexlength:
                        newtranslationparameters = translationparameters.clone()
                        newtranslationparameters.curentIndex                                    =
translationparameters.curentIndex + 1
                        newtranslationparameters.currentfilenamestr = newFileNameStr
                        newtranslationparameters.currentstep = stepCounter
                        iterator(newtranslationparameters)
                        stepCounter = stepCounter + 1

class translationparameters:
        def  _init_  (self,  currentIndex  =  0,  movatom  =  None,  currentfilenamestr  =  None,
currentstep = None):
                self.currentIndex = curentIndex
                self.movatom = movatom
                self.currentfilenamestr = currentfilenamestr
                self.currentstep = currentstep
        def clone(self):
                returnparameters = translationparameters()
                returnparameters.curentIndex = self.curentIndex
                returnparameters.movatom = self.movatom
                returnparameters.currentfilenamestr = self.currentfilenamestr
                returnparameters.currentstep = self.currentstep
                return returnparameters
def translate(currentdataArray, translationparameters):
        returnDataArray = currentdataArray
        currentMovingAtom                                                                       =
translationparameters.movatom[translationparameters.curentIndex][0]
#       print "currentMovingAtom " + str(currentMovingAtom)
        originPosition=dataArray[currentMovingAtom-1][2]
#       print "originPosition " +str(originPosition)
        currentPosition                =                float(originPosition)                +
translationparameters.movatom[translationparameters.curentIndex][2]*(translationparameters.cu
rrentstep)
#       print    "translationparameters.movatom[translationparameters.curentIndex][2]    "    +
str(translationparameters.movatom[translationparameters.curentIndex][2])
#       print "translationparameters.currentstep " + str(translationparameters.currentstep)
#       print "currentPosition " + str(currentPosition)
#       print dataArray
        #print "\n"
        returnDataArray[currentMovingAtom-1][2] = currentPosition
        #print dataArray
        #print "\n"
        return returnDataArray
def cleanup():
#clean out unnecessary files from output that will cause problems in the format step
        try:
                os.remove(outputdirectory + 'datafile.txt')
        except:
                print 'File datafile.txt does not exist'
        try:
```

```
                os.remove(outputdirectory + 'formateddatafile.txt')
        except:
                print 'File formateddatafile.txt does not exist'
        maxseps=0
        for file in os.listdir(outputdirectory):
                seps = file.count('-')
                if seps > maxseps:
                        maxseps = seps
        for file in os.listdir(outputdirectory):
                seps = file.count('-')
                if seps < maxseps:
#       Debug Prints
#                       print file,seps,maxseps
                        os.remove(outputdirectory+file)
def fileformat():
# format the files that are output from the C++ code
# recompile the input deck from the headers of the original file
        basename = inpfilename.rstrip('.inp')
        for file in os.listdir(outputdirectory):
                syntaxname = file.lstrip("formateddatafile.")
                dataname = syntaxname.rstrip('txt')
                filename = basename + dataname + 'inp'
                shutil.copy(inputdirectory + 'headerfile.txt' , outputdirectory + filename)
                print "Formating ", filename
                openfile = open(outputdirectory + filename, 'a')
                opendatafile = open(outputdirectory + file,'r')
                for line in opendatafile:
                        formline = line
                        formline = formline.replace('\t',',')
                        formline = formline.replace(',',',  ')
                        formline = formline.replace('  -',' -')
                        formline =  formline.replace(',','\t')
                        openfile.write(formline)
                tailfile = open(inputdirectory + 'tailfile.txt', 'r')
                for line in tailfile:
                        openfile.write(line)
def finalcleanup():
#Clear files that are not used by the user may be helpful in testing to not call this
function
        print "Cleaning up"
        for file in os.listdir(outputdirectory):
                if "txt" in file:
                        try:
                                os.remove(outputdirectory + file)
                        except:
                                print "File " + file + "Does not exist"
        shutil.rmtree(inputdirectory)
def main(argv):
        global movatom
        inpfiletest(argv)
        parseinpfile()
        copyInputFile()
        createdataArray()
        queryuser()
        formateddatafile()
        basefile = outputdirectory + "formateddatafile.txt"
        mytranslationparameters=translationparameters()
        mytranslationparameters.curentIndex = 0
        mytranslationparameters.movatom = movatom
        mytranslationparameters.currentfilenamestr = basefile
        mytranslationparameters.currentstep = 0
        iterator(mytranslationparameters)
        cleanup()
        fileformat()
        finalcleanup()

main(sys.argv)
```

Name:

inpincupd

Usage:

inpincupd $file

Description:

Tool to update input deck with last coordinates from an out of steps job (as opposed to an optimized job) Requires appended log file to inp.

Script:

```
sed -n -i -e '1,/C1/p' -e '/TOO MANY STEPS TAKEN/,/INTERNUCLEAR/p' $1
sed -i '/FAILURE/,/---/d' $1
sed -i 's/^$/$END/' $1
sed -i '/ANGS/d' $1
sed -i '/C1/{x;p;x;}' $1
sed -i -n -e '1,/DATA/p' -e '/^$/,/END/p' $1
```

Name:

inpupd

Usage:

inpupd $file

Description:

Tool to update input deck with last coordinates from an optimization. Requires appended log file to inp

Script:

```
sed -n -i -e '1,/C1/p' -e '/LOCATED/,/INTERNUCLEAR/p' $1
sed -i '/LOCATED/,/---/d' $1
sed -i 's/^$/$END/' $1
sed -i '/ANGS/d' $1
sed -i '/C1/{x;p;x;}' $1
sed -i -n -e '1,/DATA/p' -e '/^$/,/END/p' $1
```

Name:

locinternuc

Usage:

locinternuc $1 $2 $3
$1 = file name
$2 = atom 1
$3 = atom 2

Description:

Tool to determine inter nuclear distance between 2 atoms in data group. Works on optimized

log files

Script:

```
#!/bin/tcsh

#set echo

switch ($#)
        case 0 :
         echo "Usage: locinternuc  $1 $2 $3"
         echo "$1 file name $2 atom 1 $3      atom 2"
        breaksw
        case 1 :
         echo "Usage: locinternuc  $1 $2 $3"
         echo "$1 file name $2 atom 1 $3 atom 2"
        breaksw
        case 2 :
         echo "Usage: locinternuc  $1 $2 $3"
         echo "$1 file name $2 atom 1 $3 atom 2"
        breaksw
        case 3 :
        sed -n -e '/LOCATED/,/INTERNUCLEAR/p' $1 |sed -e'/LOCATED/,/---/d' -e 's/^$/$END/' -e
'/ANGS/d' -e '/C1/{x;p;x;}' |sed -n -e '1,/DATA/p' -e '/^$/,/END/p' > input.inp
        internuc.py $2 $3 $1
        rm input.inp
        endif
        breaksw
endsw
```

Name:

stepsinternuc

Usage:

stepsinternuc $1 $2 $3
$1 = file name
$2 = atom 1
$3 = atom 2

Description:

Tool to determine inter nuclear distance between 2 atoms in data group. Works on to many steps taken log files

Script:

```
#!/bin/tcsh

switch ($#)
        case 0 :
        echo "Usage: stepsinternuc  $1 $2 $3"
        echo "$1 file name $2 atom 1 $3 atom 2"
        breaksw
        case 1 :
         echo "Usage: stepsinternuc  $1 $2 $3"
         echo "$1 file name $2 atom 1 $3      atom 2"
        breaksw
        case 2 :
         echo "Usage: stepsinternuc  $1 $2 $3"
         echo "$1 file name $2 atom 1 $3      atom 2"
        breaksw
        case 3 :
        sed -n -e '/TOO MANY STEPS TAKEN/,/INTERNUCLEAR/p' $1 | sed -e '/FAILURE/,/---/d' -e
's/^$/$END/'  -e  '/ANGS/d'  -e  '/C1/{x;p;x;}'  |sed  -n  -e  '1,/DATA/p'  -e  '/^$/,/END/p'  >
input.inp
        internuc.py $2 $3 $1
#       rm input.inp
        breaksw
endsw
```

Name:

internuc.py

Usage:

Python script to determine inter nuclear distance between two atoms passed input.inp $1, and $2 which are atoms of interest $3 is original file name

Description:

Helper file to locinternuc and stepsinternuc

Script:

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

#python script to determin inter nuclear distance between two atoms
#passed input.inp $1 $2 which are atoms of interest $3 is original file name

import math
import sys

atm1 = sys.argv[1]
atm2 = sys.argv[2]
name = sys.argv[3]

data = open('input.inp')

list = []

for line in data:
        element = line.split()
        list.append(element)
#print list


diff=[]

atm1 = int(atm1)-1
atm2 = int(atm2)-1


for element in range(2,5):
        difference = float(list[atm1][element]) - float(list[atm2][element])
#       print difference
        diff.append(difference)

#print diff

vec=(diff[0] ** 2 + diff[1] ** 2 + diff[2] ** 2) ** .5
print str(round(vec,4)) + "\t" + name
```

Name:

ml

Usage:

ml $extension $listname

Description:

Makes a list of files with an extension without the extension

Script:

```
switch ($#)
        case 0:
        echo "Usage: makelist extension filename"
        breaksw
        case 1:
        echo "Usage: makelist extension filename"
        breaksw
        case 2:
        echo ok
ls *."$1" > $2

sed -i "s/.$1//" $2
        breaksw
        case *:
        echo "To many arguments"
        breaksw
endsw
```

Name:

pullegas

Usage:

N/A

Description:

Pulls out the absolute energy of log files in a gas phase optimization

Script:

```
pullegas.2 |sed 's/.log.*-/\t-/'
```

Name:

pullegas.2

Usage:

N/A

Description:

Helper tool for pullegas

Script:

```
#!/bin/sh

ls *.log|xargs -I {} pullegas.1 {}|sed -e '
/.log$/ {
N
s/\n//
}'
```

Name:

pullegas.1

Usage:

N/A

Description:

Helper tool for pullegas.1

Script:

```
#!/bin/tcsh
#pwd
echo $1
grep "TOTAL ENERGY =" $1 | tail -n 1
```

Name:

pullallthermo

Usage:

N/A

Description:

Pulls and displays all thermodynamic calculated in a hessian calculation

Script:

```
#!/bin/tcsh

echo "Name \tZPE \tEt \tEr \tEtot \tH \tG \tS \tVibTherm"
foreach i ( ls *.log)
grep -asH -A 29 "THERMOCHEMISTRY AT T=" $i |\
sed '/THERMOCHEMISTRY/,/MOLECULE/d' |\
sed -e '2,12d' -e '15d' |\
sed 's/\.log.*KCAL\/MOL//' |\
sed 's/ \+/\n/g' |\
sed -e '3,5d' -e '7,13d' -e '15,21d' -e '25,26d' -e '28,35d' -e '37d' |\
sed -e ':a;N;s/\n/\t/g' |\
sed -e ':a;N;s/\n/\t/g' |\
sed -e ':a;N;s/\n/\t/g' |\
sed -e ':a;N;s/\n/\t/g'
end
```

Name:

pullesol

Usage:

pullesol no args = list all the energies in solvent
pullesol 1 arg = used for pullesolmin to id minimum structure

Description:

Pulls out the absolute energy of log files in a solven phase optimization

Script:

```
#!/bin/tcsh

switch ($#)
        case 0:
        grep -i "total free" *.log | grep A.U. | sed -e 's/.log.*=. /\t/' -e 's/ A.*$//'
        breaksw
        case 1:
        grep -i "total free" *.log | grep A.U. | sed -e 's/.log.*=. /\t/' -e 's/ A.*$//' |grep
$1
        breaksw
        case *:
        echo "Usage: pullesol no args = list all the energies in solvent "
        echo "Usage: pullesol 1 arg =  used for pullesolmin to id minimum structure "
        echo "To many arguments"
        breaksw
endsw
```

Name:

pullesolmin

Usage:

N/A

Description:

Pulls out the absolute energy of log files in a solvent phase optimization and identifies the lowest energy

Script:

```
grep -i 'total free' *.log | sed -n '/A.U./p' | sed -e 's/^.*=.[ \t]*//' -e 's/ .*$//' | sort
| tail -n 1 | sed 's/^.*\.//' |xargs -I {} pullesol {}
```

Name:

pullthermo

Usage:

N/A

Description:

Pulls out the thermodynamic information from log files

Script:

```
#!/bin/tcsh

grep -ra -A 30 "THERMOCHEMISTRY AT T=" *.log |\
sed '/THERMOCHEMISTRY/,/MOLECULE/d' |\
sed -n -e '/KCAL/p' -e '/TOTAL/p' |\
sed 's/\.log.*KCAL\/MOL/,/' |\
sed 's:MOL:MOL\\:' |\
sed -e :a -e '/\\$/N; s/\\\n//;ta' |\
sed 's/KJ\/MOL.*TOTAL/,/' |\
sed 's/ [[:space:]]*/,/g' |\
sed 's/,,,/,/' |\
sed 's/,,/,/' |\
sed 's/,/\t/g'
```

Name:

pulltime

Usage:

pulltime

no flag pulls the time

any flag prints context of time output as well

Description:

Pulls out the total wall clock time from log files in folder

Script:

```
#!/bin/tcsh

#useage is just pulltime any flag should add names

switch ($#)
        case 0:
        foreach file (*.log)
                tac $file |grep -m 1 WALL|sed -e 's/ TOTAL WALL CLOCK TIME=. //' -e 's/
SECONDS.*$//'
        end
        breaksw
        case 1:
        foreach file (*.log)
                echo -n "$file \t"
                grep --null NODES $file
                tac $file |grep -m 1 WALL|sed -e 's/ TOTAL WALL CLOCK TIME=. //' -e 's/
SECONDS.*$//'
        end
        breaksw
endsw
```

Name:

rag

Usage:

Run All GAMESS

rag filelist cpus time(##:##:##) queue(optional) version(optional)

Description:

Script that allows multiple submissions to torque queueing system

Script:

```
#!/bin/tcsh

switch ($#)
        case 0:
        echo "Usage: rag filelist cpus time(##:##:##) queue(optional) version(optional)"
        breaksw
        case 1:
        echo "Usage: rag filelist cpus time(##:##:##) queue(optional) version(optional)"
        breaksw
        case 2:
        echo "Usage: rag filelist cpus time(##:##:##) queue(optional) version(optional)"
        breaksw
        case 3:
        cat $1 | xargs -t -I {} gms -l "{}.log" -p $2 -w $3 "{}.inp"
        breaksw
        case 4:
        cat $1 | xargs -t -I {} gms -q $4 -l "{}.log" -p $2 -w $3 "{}.inp"
        breaksw
        case 5:
        cat $1 | xargs -t -I {} gms -v $5 -q $4 -l "{}.log" -p $2 -w $3 "{}.inp"
         breaksw
        case *:
        echo "To many arguments"
        breaksw
endsw
```

Name:

renamefiles

Usage:

renamefiles string1 string2

Description:

Renames batches of files based on a sed search and replace

Script:

```
#!/bin/tcsh

switch ($#)
        case 0:
        echo "Usage: renamefiles string1 string2"
        breaksw
        case 1:
        echo "Usage: renamefiles string1 string2"
        breaksw
        case 2:

foreach file (*.*)
        set newname=`echo $file | sed -e "s/$1/$2/"`
        mv $file $newname
end
        breaksw
        case *:
        echo "Usage: renamefiles string1 string2"
        breaksw
endsw
```

Name:

rotation

Usage:

rotation inputfile totrotation increment

Description:

Simple tool for doing a quick rotation in a file

Script:

```
#!/bin/tcsh
switch ($#)
        case 0:
        echo "Usage: rotation inputfile totrotation increment"
        breaksw
        case 1:
        echo "Usage: rotation inputfile      totrotation increment"
         breaksw
        case 2:
        echo "Usage: rotation inputfile      totrotation increment"
         breaksw
        case 3:
                sed -n -e '1,/C1/p' $1.inp > rothead.inp
                sed -n -e '/^ $END/,$p' $1.inp > rottail.inp
                sed -n -e '/C1/,/$END/p' $1.inp > rotdata.inp
                sed -i -e '1d' -e '$d' rotdata.inp

                echo "how many atoms will rotate (do not include points on point of rotation)"
                set atoms=$<

                echo "enter atom number of rotation point 1"
                set atm1=$<
                echo "enter atom number of rotation point 2"
                set atm2=$<

                set counter=1
                set rotatms = ( 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0)
                # current rotation atoms is limited to 40
                while ($counter <= $atoms)
                        echo "enter atom number"
                        @ rotatms[$counter] = $<
                        @ counter = $counter + 1
                end

                set counter = 1
                set sedcom = ( "sed -e $atm1' d' -e $atm2' d' ")
                while ($counter <= $atoms)
                        set sedcom = ( $sedcom "-e  $rotatms[$counter]' d' " )
                        @ counter = $counter + 1
                end
                set sedcom = ($sedcom "rotdata.inp > cleanedrotdata.inp" )
                eval $sedcom

                sed -i -e 's/[ \t]*[ \t]/,/g' -e 's/.0,/.0\t/' -e 's/,/      /' rotdata.inp

                set counter = 1
                sed -n $atm1'p' rotdata.inp > transdata
```

```
                sed -n $atm2'p' rotdata.inp >> transdata
                while ($counter <= $atoms)
                        sed -n $rotatms[$counter]'p' rotdata.inp >> transdata
                        @ counter = $counter + 1
                end


                sed -i -e 's/[ \t]*[ \t]/,/g' -e 's/,/, /g' -e 's/, -/,-/g' -e 's/.0,/.0\t/' -
e 's/,/     /' -e 's/,/      /g' cleanedrotdata.inp

                set tr = $2
                set inc = $3
                set cr = 0
                while ($cr < $tr)
                        #echo $cr
                        @ cr = $cr + $inc
                        cat rothead.inp > $1.$cr.inp
                        crot  $atoms $cr transdata >> $1.$cr.inp
                        cat cleanedrotdata.inp >> $1.$cr.inp
                        cat rottail.inp >> $1.$cr.inp
                        sed -i 's/,/\t/' $1.$cr.inp
                end

                rm rothead.inp rotdata.inp rottail.inp transdata cleanedrotdata.inp

        breaksw
        case *:
        echo "To many arugments"
        echo "Usage: rotation inputfile       totrotation increment"
    breaksw
endsw
```

Name:

crot

Usage:

N/A

Description:

C++ binary to help rotation

Script:

See rotation.cpp

Name:

rotation.cpp

Usage:

N/A

Description:

Source C++ file for crot for utilization in CREPES

Script:

```
//app for rotation about a line
#include <iostream>
#include <cmath>
#include <fstream>
#include <stdio.h>
#include <cstring>

using namespace std;

#define pi 3.14159265

//define variables
        double rot1[3], rot2[3];
        double ux,uy,uz;
        double uxn,uxd;
        string line;
        string first;
        string second;
        string strxrot1, stryrot1, strzrot1;
        string strxrot2, stryrot2, strzrot2;
        int deg;
        int atoms=0;
        double transmat[3][3] = {{ 1,1,1},{1,1,1},{1,1,1}};
        int rotid=0;

// args are #of atoms to rotate
// degree of rotation
// input file
double **createcoordarrays (int);
string *createnamearray (int);
void assigntransmat(double* , double*, int );
void readfile();
void printdata(int, string*, double* , double*,double**);



int main (int argc, char **argv){
//      cout << "program initialized \n";
        atoms = atoi(argv[1]);
//      cout << "input atoms : " << atoms << endl;

        //maintain precision
        cout.precision(8);
        cout << fixed;

//      cout << "SET PRECISION\n";
```

```
        // setup some variables
        double **coord = createcoordarrays(atoms);
//      cout << "COORD ARRAY CREATED\n";

        double **transf = createcoordarrays(atoms);
//      cout << "TRANSF ARRAY CREATED\n";

        int allatoms = atoms + 2;

//      cout << "allatoms: " << allatoms << endl;

        string *name = createnamearray(allatoms);
//      cout << "NAME " << endl;



        // open file
//      cout << "OPENING FILE" << endl;
        ifstream coordsfile;
        coordsfile.open(argv[3]);
        //error if file does not exist
        if (!coordsfile){
                cout << " File does not exist\n";
                exit (1);
        }

//      cout << "FILE IS OPEN " << endl;
        if (coordsfile.is_open()){


        int counter = 0;
        while (getline(coordsfile,line)){
                counter++;
//              cout << "\n\t LINE READING NUMBER " << counter << endl;
                size_t i = line.find("\t");

//              cout << "\t\t searched for tab found at i: " << i << endl;

                if (i != string::npos){
                        size_t y=0;
                        if (!line.empty()){
                                string first="";
                                string second="";
                                while (y!=i){
                                        first += line[y++];
                                }
                                //store name
                                name[rotid] = first;
//                              cout << "\n\t\t " << first << endl;
                                y++;

                                while (y!=line.length()){
                                        second += line[y++];
                                }
//                              cout << "\n\t\t " << second << endl;
                                //Parse numeric half
                                size_t r=second.find(",");
                                size_t s=second.rfind(",");
//                              cout << "\t\t r : " << r << endl;
//                              cout << "\t\t s : " << s << endl;

                                int z=0;

//                              cout << "\n\t\tbeforeif " << endl;
                                if (!second.empty()){
                                        strxrot1="";
                                        stryrot1="";
                                        strzrot1="";
```

```
//                              cout << "\t\t\t z < r "<< endl;

                                while (z<r){
                                        if (rotid==0) {
                                                strxrot1 += second[z++];
                                                rot1[0] = atof(strxrot1.c_str());
//                                              cout << "x1";
                                        }
                                        if (rotid==1) {
                                                strxrot2 += second[z++];
                                                rot2[0] =atof(strxrot2.c_str());
//                                              cout << "x2";
                                        }
                                        if (rotid>=2){
//                                              cout << "x";
                                                strxrot1 += second[z++];
                                                int coordid=rotid-2;
                                                coord[0][coordid]=atof(strxrot1.c_str());
//                                              cout << "x3";
                                        }
                                }
//                              cout << endl;
                                z++;

//                              cout << "\t\t\t z < r && z > r"<< endl;
                                while (z<s and z>r){

                                        if (rotid==0) {
                                                stryrot1 += second[z++];
                                                rot1[1] =atof (stryrot1.c_str());
//                                              cout << "y1";
                                        }
                                        if (rotid==1) {
                                                stryrot2 += second[z++];
                                                rot2[1] =atof(stryrot2.c_str());
//                                              cout << "y2";
                                        }
                                        if (rotid>=2){
                                                stryrot1 += second[z++];
                                                int coordid=rotid-2;
                                                coord[1][coordid]=atof(stryrot1.c_str());
//                                              cout << "y3";
                                        }
                                }
                                z++;
//                              cout << endl;
//                              cout << "\t\t\t z != second.length "<< endl;
                                while (z!=second.length()){

//                                      cout << "\n\t\t\t if 1 " << endl;

                                        if (rotid==0) {
                                                strzrot1 += second[z++];
                                                rot1[2] =atof (strzrot1.c_str());
//                                              cout << "z1";
                                        }
//                                      cout << "\n\t\t\t if 2 " << endl;

                                        if (rotid==1) {
                                                strzrot2 += second[z++];
                                                rot2[2] =atof (strzrot2.c_str());

                                        }
//                                      cout << "\n\t\t\t if 3 " << endl;

                                        if (rotid>=2){
                                                strzrot1 += second[z++];
                                                int coordid=rotid-2;
                                                coord[2][coordid]=atof(strzrot1.c_str());
```

```
//                                                      cout << "z3" << endl;
                                                }
//                                              cout << "\n\t\t\t end of three ifs " << endl;
                                        }
                                }
//                              cout << endl << "before : " <<         rotid;
                                rotid++;
//                              cout << " after : " << rotid << endl;
                        }
                }
                else {
                        string first=line;
                        string second="";
                }
//              cout << "\n\t LINE READ NUMBER " << counter << endl << endl;
        }
}


//      cout << "CLOSING FILE\n";

        coordsfile.close();


        deg = atoi(argv[2]);
//      cout << "SET the ANGLE OF ROTATION " << deg << "\n";
        assigntransmat(rot1, rot2, deg);


//translate coords
        for(int i=0; i < 3; i++){
                for (int j=0; j < atoms; j++){
                        coord[i][j] = coord[i][j] - rot1[i];
                }
        }



        // do matrix transformation (rotation) about line

//      cout << "PERFORMING MATRIX TRANSFORMATION\n";
        for (int i=0; i < 3; i++){
                for (int j=0; j < atoms; j++){
                        for (int k=0; k < 3 ; k++){
                                transf[i][j] += transmat[i][k]*coord[k][j];
                        }
                }
        }

//translate coords back
        for(int i=0; i <3; i++){
                for (int j=0; j < atoms; j++){
                        transf[i][j] = transf[i][j] + rot1[i];
                }
        }

//      printdata(allatoms ,name, rot1, rot2, transf);
        for (int i=0; i<allatoms; i++){
                if (i==0){
                        cout << name[i];
                        for (int r1=0; r1<3; r1++){
                                printf("  %*.*f",15,8,rot1[r1] );
                        }
                        cout << endl;
                }
                if (i==1){
                        cout << name[i];
                        for (int r2=0; r2<3; r2++){
                                printf("  %*.*f",15,8,rot2[r2] );
```

```
                        }
                        cout << endl;
                }
                if (i>1){
                        cout << name[i];
                        for (int r3=0; r3<3; r3++){
                                printf("  %*.*f",15,8,transf[r3][i-2] );
                        }
                        cout << endl;
                }
        }
        /*
        //cleanup

        delete[] name;
        delete[] coord;
        delete[] transf;

        //cout << "\nDone\n";
        */
        return 0;
}

double **createcoordarrays (int atoms){

        //cout << endl << "-------- start createcoordarrays (atoms: " <<  atoms << ") " <<
endl;

        double **j;
        j = new double*[3];
        for (int i=0;i<3;i++){
                j[i]=new double[atoms];
        }

        //cout << endl << "-------- end createcoordarrays " << endl;
        return j;
}

string *createnamearray (int allatoms){
        // create name array
        string *name;
        name = new string[allatoms];
        for (int i=0;i<allatoms;i++){
                name[i]="Some Literal string  ";
        }
        return name;
}


void assigntransmat(double rot1[], double rot2[],int deg){
        double u=(rot1[0]-rot2[0]);
        double v=(rot1[1]-rot2[1]);
        double w=(rot1[2]-rot2[2]);

        double l=sqrt(u*u+v*v+w*w);

        transmat[0][0]=(u*u+(v*v+w*w)*(cos(deg*pi/180)))/(l*l);
        transmat[0][1]=(u*v*(1-cos(deg*pi/180))-w*l*sin(deg*pi/180))/(l*l);
        transmat[0][2]=(u*w*(1-cos(deg*pi/180))+v*l*sin(deg*pi/180))/(l*l);
        transmat[1][0]=(u*v*(1-cos(deg*pi/180))+w*l*sin(deg*pi/180))/(l*l);
        transmat[1][1]=(v*v+(u*u+w*w)*(cos(deg*pi/180)))/(l*l);
        transmat[1][2]=(v*w*(1-cos(deg*pi/180))-u*l*sin(deg*pi/180))/(l*l);
        transmat[2][0]=(u*w*(1-cos(deg*pi/180))-v*l*sin(deg*pi/180))/(l*l);
        transmat[2][1]=(v*w*(1-cos(deg*pi/180))+u*l*sin(deg*pi/180))/(l*l);
        transmat[2][2]=(w*w+(u*u+v*v)*(cos(deg*pi/180)))/(l*l);
}

/*
void readfile(){
```

```
// read in the file
if (coordsfile.is_open()){
	while (getline(coordsfile,line)){
		size_t i = line.find("\t");
		if (i != string::npos){
			size_t y=0;
			if (!line.empty()){
				string first="";
				string second="";
				while (y!=i){
					first += line[y++];
				}
				//store name
				name[rotid] = first;

				y++;

				while (y!=line.length()){
					second += line[y++];
				}
				//Parse numeric half
				size_t r=second.find(",");
				size_t s=second.rfind(",");

				if  (i!=string::npos){
					size_t y=0;
					if (!second.empty()){
						strxrot1="";
						stryrot1="";
						strzrot1="";
						while (y<r){
							if (rotid==0) {
								strxrot1 += second[y++];
								rot1[0] = atof(strxrot1.c_str());
							}
							if (rotid==1) {
								strxrot2 += second[y++];
								rot2[0] =atof(strxrot2.c_str());
							}
							if (rotid>=2){
								strxrot1 += second[y++];
								int coordid=rotid-2;

	coord[0][coordid]=atof(strxrot1.c_str());
							}
						}

						y++;
						while (y<s){
							if (rotid==0) {
								stryrot1 += second[y++];
								rot1[1] =atof (stryrot1.c_str());
							}
							if (rotid==1) {
								stryrot2 += second[y++];
								rot2[1] =atof(stryrot2.c_str());
							}
							if (rotid>=2){
								stryrot1 += second[y++];
								int coordid=rotid-2;

	coord[1][coordid]=atof(stryrot1.c_str());
							}
						}
						y++;
						while (y!=second.length()){
							if (rotid==0) {
								strzrot1 += second[y++];
								rot1[2] =atof (strzrot1.c_str());
```

```
                                                        }
                                                        if (rotid==1) {
                                                                strzrot2 += second[y++];
                                                                rot2[2] =atof (strzrot2.c_str());
                                                        }
                                                        if (rotid>=2){
                                                                strzrot1 += second[y++];
                                                                int coordid=rotid-2;

        coord[2][coordid]=atof(strzrot1.c_str());

                                                        }

                                                }
                                        }
                                }
                                rotid++;

                        }
                }
                else {
                string first=line;
                string second="";
                }
        }
}
coordsfile.close();

}
*/

/*void printdata(int allatoms ,string* name[], double rot1[], double rot2[], double**
transf[][]){
        for (int i=0; i<allatoms; i++){
                if (i==0){
                        cout << name[i];
                        for (int r1=0; r1<3; r1++){
                                printf("  %*.*f",15,8,rot1[r1] );
                        }
                        cout << endl;
                }
                if (i==1){
                        cout << name[i];
                        for (int r2=0; r2<3; r2++){
                                printf("  %*.*f",15,8,rot2[r2] );
                        }
                        cout << endl;
                }
                if (i>1){
                        cout << name[i];
                        for (int r3=0; r3<3; r3++){
                                printf("  %*.*f",15,8,transf[r3][i-2] );
                        }
                        cout << endl;
                }
        }
}
*/
```

Name:

srn

Usage:

Description:

Tool that converts crepes naming into more conventional scheme

Script:

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

import sys
import itertools

lprog=['g','g-','t']
uprog=['G','G-','T']
global header
header = []

if len(sys.argv) == 1:
        repdef=raw_input('Enter definition: ')
else:
        repdef = sys.argv[1]

if len(sys.argv) == 2:
        input=raw_input('Enter input: ')
else:
        input = sys.argv[2]

inpcopy=input
repdef = repdef.split('.')
input = input.split('.')

inputcopysplit=inpcopy.split('.')
for piece in range(len(inputcopysplit)):
        if inputcopysplit[piece].isalpha():
                header.append(inputcopysplit[piece])
                input.pop(0)



if len(repdef)==len(input):
        for subs in range(len(repdef)):
#               print 'subs' + str(subs)
#               print 'input[subs]' + str(input[subs])
                if input[subs] == '0':
#                       print subs
                        input[subs] = repdef[subs]
                if input[subs] == '120':
                        if repdef[subs][0].isupper():
                                for ele in range(len(uprog)):
                                        if repdef[subs] == uprog[ele]:
                                                try:
                                                        output= uprog[ele+1]
                                                        input[subs]=output

                                                except:
```

```
                                                output= uprog[0]
                                                input[subs]=output
                        else:
                                for ele in range(len(lprog)):
                                        if repdef[subs] == lprog[ele]:
                                                try:
                                                        output=lprog[ele+1]
                                                        input[subs]=output
                                                except:
                                                        output= lprog[0]
                                                        input[subs]=output
                if input[subs] == '240':
                        if repdef[subs][0].isupper():
                                for ele in range(len(uprog)):
                                        if repdef[subs] == uprog[ele]:
                                                try:
                                                        output= uprog[ele+2]
                                                        input[subs]=output

                                                except:
                                                        try:
                                                                output= uprog[ele+1]
                                                                output= uprog[0]
                                                                input[subs]=output
                                                        except:
                                                                output=uprog[1]
                                                                input[subs]=output



                        else:
                                for ele in range(len(lprog)):
                                        if repdef[subs] == lprog[ele]:
                                                try:
                                                        output=lprog[ele+2]
                                                        input[subs]=output
                                                except:
                                                        try:
                                                                output= lprog[ele+1]
                                                                output= lprog[0]
                                                                input[subs]=output
                                                        except:
                                                                output=lprog[1]
                                                                input[subs]=output

#       print repdef
        print inpcopy+'\t' + '.'.join(input)

else:
        print "Definition does not match case length"
```

Name:

status

Usage:

N/A

Description:

lists the status of log files

Script:

```
#!/bin/tcsh

switch ($#)
        case 0:
echo -n  "Status of log files in "
pwd

echo -n  "1 Number of log files: "
 ls *.log |wc -l

echo -n  "2 Number with Located: "
grep -li located *.log |wc -l

echo -n  "3 Number out of steps: "
grep -li "TOO MANY STEPS TAKEN" *.log |wc -l

echo -n  "4 Number BKRNR: "
grep -li bkrnr *.log |wc -l

echo -n  "5 Number Corrupted: "
grep -li corrupt *.log |wc -l

echo -n  "6 Number Gradient out of range: "
grep -li "GRADIENT OUT OF RANGE" *.log |wc -l

echo -n  "7 Number of nospc: "
grep -li nospc *.log |wc -l

echo -n  "8 Number SCF not converged: "
grep -li "SCF HAS NOT CONVERGED" *.log |wc -l

echo -n  "9 Number Else: "
grep -Li located *.log |xargs -I {} grep -Li "TOO MANY STEPS TAKEN" {} | xargs -I {} grep -Li
bkrnr {}|xargs -I {} grep -Li corrupt {}|xargs -I {} grep -Li "GRADIENT OUT OF RANGE" {} |
xargs -I {} grep -Li nospc {}| xargs -I {} grep -Li "SCF HAS NOT CONVERGED" {} | wc -l


        breaksw
         case 1:
        if ($1 == "1") ls *.log |sed 's/.log//'
        endif
        if ($1 == "2") grep -li located *.log|sed 's/.log//'
        endif
        if ($1 == "3") grep -li "TOO MANY STEPS TAKEN" *.log|sed 's/.log//'
        endif
        if ($1 == "4") grep -li bkrnr *.log|sed 's/.log//'
        endif
        if ($1 == "5") grep -li corrupt *.log|sed 's/.log//'
```

```
        endif
        if ($1 == "6") grep -li "GRADIENT OUT OF RANGE" *.log|sed 's/.log//'
        endif
        if ($1 == "7") grep -li nospc *.log |sed 's/.log//'
        endif
        if ($1 == "8") grep -li "SCF HAS NOT CONVERGED" *.log|sed 's/.log//'
        endif

        if ($1 == "9") grep -Li located *.log |xargs -I {} grep -Li "TOO MANY STEPS TAKEN" {}
| xargs -I {} grep -Li bkrnr {}|xargs -I {} grep -Li corrupt {}|xargs -I {} grep -Li
"GRADIENT OUT OF RANGE" {} |xargs -I {} grep -Li nospc {} |xargs -I {} grep -Li "SCF HAS NOT
CONVERGED" {} |sed 's/.log//'
        endif

        breaksw
        case 2:
                echo    "Too many args"
                echo    "Usage: status none(print outline) Number(endifles of type denoted)"
                echo    "Example:  status 4"
                echo    "Prints list of files containing bkrnr"
        breaksw
endsw
```

Name:

theory

Usage:

theory filename(no extention)

Description:

Finds the level of theory from the given file name

Script:

```
#!/bin/tcsh

switch ($#)
        case 0:
        echo "Usage: theory filename"
        echo "Usage: Finds the level of theory from the given file name (no extention)"
        breaksw
        case *:
        if (-e "$1.log") then
                cat "$1.log" |sed -n -e '/$contrl/I {p;n;p}' -e '/$basis/Ip' -e '/$statpt/Ip'
-e '/$pcm/Ip' -e '/located/Ip' -e '/imaginary/Ip' |sed -e '/OPTIONS/d'
        endif
        if ( ! -e "$1.log" && -e "$1.inp") then
                cat "$1.inp" |sed -n -e '/$contrl/I,/$end/Ip' -e '/$basis/Ip' -e '/$statpt/Ip'
-e '/$pcm/Ip'
        endif
        breaksw
endsw
```

Name:

tkrtogam

Usage:

tkrtogam filename headerfilename

Description:

Convert tinker output files to gamess files

Script:

```
#!/bin/tcsh
switch ($#)
        case 0:
        echo "Usage: tkrtogam filename headerfilename"
        breaksw
        case 1:
        echo "Usage: tkrtogam filename headerfilename"
        breaksw
        case 2:
                sed -n '1,/C1/p' $2 > "$1.inp"
                tkrtogam.py $1 >> "$1.inp"
                echo ' $END' >> "$1.inp"
        breaksw
        case *:
        echo "To many arguments"
        breaksw
endsw
```

Name:

tkrtogam.py

Usage:

N/A

Description:

helper file for tkrtogam

Script:

```
#! /usr/bin/env python
#Usage
#arg1 = file name no extension

import sys
arg1 = sys.argv[1]

inf = open(arg1)
#outf = open(arg1+'.inp','w')
elements                                                =                                                       ["-
","H","He","Li","Be","B","C","N","O","F","Ne","Na","Mg","Al","Si","P","S","Cl","Ar","K","Ca",
"Sc","Ti","V","Cr","Mn","Fe","Co","Ni","Cu","Zn","Ga","Ge","As","Se","Br","Kr","Rb","Sr","Y",
"Zr","Nb","Mo","Tc","Ru","Rh","Pd","Ag","Cd","In","Sn","Sb","Te","I","Xe","Cs","Ba","La","Ce"
,"Pr","Nd","Pm","Sm","Eu","Gd","Tb","Dy","Ho","Er","Tm","Yb","Lu","Hf","Ta","W","Re","Os","Ir
","Pt","Au","Hg","Tl","Pb","Bi","Po","At","Rn","Fr","Ra","Ac","Th","Pa","U","Np","Pu","Am","C
m","Bk","Cf","Es","Fm","Md","No","Lr","Rf","Db","Sg","Bh","Hs","Mt"]
atoms = []
lines = 0
for line in inf:
        lines+=1
        if lines > 1:
                atoms.append(line[8:47])
#               print line[8:47]
#print lines
for atom in atoms:
        counter=-1
        for element in elements:
                counter+=1
                if atom[0:1] == element:
                        atom = atom[:4] + "\t" + str(round(counter,1)) + "\t" + atom[5:]
                        print atom

#       inf.readline()
#       outf.write(line)
```

Name:

ui (update input)

Usage:

updateinput $1 $2'
$1 = file list without extension(folder must contain both input and output decks)
$2 = extension to append between current filename and .inp (optional)'

Description:

batch tool that updates a group of files with the optimized structure and create new inp with extension

Script:

```
#!/bin/tcsh

switch ($#)
        case 0:
        echo 'Usage: updateinput $1 $2'
        echo '$1 = file list without extension(folder must contain both input and output
decks)'
        echo '$2 = extension to append between current filename and .inp (optional)'
        breaksw
        case 1:
        cat $1 |xargs -I {} cat2files "{}.inp" "{}.log"
        cat $1 |xargs -I {} inpupd "{}.inp"
        breaksw
        case 2:
        cat $1 |xargs -I {} cp "{}.inp" "{}.$2.inp"
        cat $1 |xargs -I {} cat2files "{}.$2.inp" "{}.log"
        cat $1 |xargs -I {} inpupd "{}.$2.inp"
        breaksw
endsw
```

CHAPTER 4. AB INITIO INVESTIGATION OF A MODELED APIOSE BORON APIOSE CROSSLINK

Abstract

Rhamnogalacturonan-II (RG-II) is an import pectic fraction polysaccharide. It forms a stable dimer (dRG-II-B) via a borate crosslink bridge at the apiosyl residue. The crosslink in RG-II has been modeled with β-D-di-methyl-apiose dimer to identify the expected stereochemistry of the borate diester's crosslink. The rotational conformation space has been thoroughly examined with CREPES. The potential energy surface has been mapped along the reaction pathway and thermodynamic data has also been computed.

Introduction

As noted in chapter 1, plant cell wall structure is essential for plant health. For many plants boron plays a critical role in the rigidity of plant cell walls. A lack of rigid structure will result in the plants exhibiting stunted growth, misshaped leaves, discoloration, and hollow stems. Brittle stems and leaves are produced, and fewer reproductive organs (i.e. buds and flowers) form in boron deficient plants.[1,2]

Rhamnogalacturonan II (RG-II) is an important, structurally complex polysaccharide. RG-II is one of three predominant pectic polysaccharides (homogalacturonan (HG), rhamnogalacturonan I (RGI), and RG-II)[3,4]. RG-II's back bone is made of 8 α-(1-4)-linked D-galacturonic acids. HG is made of long uniform α-(1-4)-linked D-galacturonic acids chains. RG-II binds covalently to HG and is linked to create a large macromolecule that composes the pectic infrastructure.

RG-II is highly conserved in high order plants and is a major component of the pectic fraction of pectin rich plants[5]. It has been found in all vascular plants examined to date[1]. RG-II has varying amounts in the plant cell walls across different types of plants. In dicots it accounts for 1%-4% of the primary cell wall in pectin rich plants where 0.1% has been found in pectin poor plant cell wall material[1] The diversity of plants in which the rhamnogalacturonan II crosslink mechanism occurs implies that it developed early in plant evolution and it is likely that it is fundamental to plant cell wall structure.

Rhamnogalacturonan II contributes significantly to cell wall structure in plant cell walls. Chemically there are a number of features of RG-II that contribute to its cellular function. RG-II is evenly distributed throughout the plant cell wall. This relatively low molecular weight polysaccharide typically exists as a dimer with a borate diester crosslink[6,7] (It is worth noting that this is not actually a ester but this type of system seems to be ubiquitously referred to as such. It is in fact a diol that forms a diether) The borate diester crosslink occurs between two apiosyl residues[8] Considerable effort has gone into identifying the glycosyl residues of RG-II[8–20] RG-II has 4 side chains labeled A-D. Side Chains A and B have apiosyl linkages to the back bone.

β-D-erythro-furanoses readily form borate esters[21]; β-D-Api*f*, the apiose found in RG-II, is the only component of RG-II with a β-D-erythro-furanose configuration. Glycosyl linkage analysis (GLA) suggested a borate crosslink at some combination of A and B side chain apiosyl residues.[7,22] GLA indicated that the crosslink on apiose occurred in a 2,3 configuration. Which combination of A and B side chain was borate crosslinked was not able to be determined via NMR but a chemical method was developed to show that the crosslink

occurs at the 2, 3 position on the A side chain and does not connect to the B side chain in any way[23].

There are two possible configurations for the crosslink. The boron crosslink is a chiral center. In a simplified system ^{11}B NMR spectroscopy is not able to distinguish between two possible diastereomers of methyl apiofuranosides. This is because the chemical shift difference is less than the line width of the ^{11}B signal. ^{1}H and ^{13}C NMR has confirmed that two conformations exist however the peaks are not assignable. ^{1}H and ^{13}C NMR has also shows nearly equal distribution of each conformer.[24]

Some factors that control the regulation of RG-II crosslink that have been shown are pH and cation dependence. Borate-diol esters are hydrolyzed in vitro at pHs between 3 and 4.[7,25] It is believed that this is the primary mechanism for regulating the borate crosslink. This process is facilitated by divalent cations with large radii.

The dimers of apiose that have been modeled are bis[1-*O*-,4-*O*-dimethyl-apio-β-D-furanoside] –(R)-2,3:2’3’- and (S)-2,3:2’3’-borate hereafter referred to R and S dimethyl-apiose (DMA)(Figure 4.1). Calculated thermodynamic data will be presented to suggest a mechanism for the formation of the crosslink. The energy differentiation between R and S configurations will be presented to correlate to reported values.

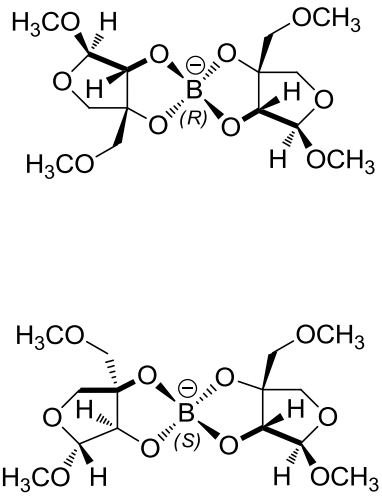


Figure 4.1 R (Top) & S (Bottom) dimethyl-apiose (Shown in wedge dash projection to emphasize stereochemistry)

Methods

For computational tractability at ab initio levels of theory it is not feasible to use polysaccharide residues to represent RG-II. Instead, a representative system was modeled. For this study we will start with just the β-D-Api*f* residue. Apiose 1,4 hydroxyls are capable of hydrogen bonding and other intra molecular interactions. To mitigate the interaction of these groups would that typically not be available in the system we are attempting to model, methyl groups have been added to β-D-Api*f* at the 1 and 4 positions. The methyl groups also, although to a lesser extent, simulate some of the steric hindrance along the chain of saccharides.

The dimerization of DMA by boron crosslink is not expected to be a one step process. The stepwise dehydration reactions will be modeled here. For such highly variable systems conformational searching is impetrative in finding low energy configurations.

Conformational searching was utilized to assure that the lowest energy configurations were found. Complete Rotation for the Evaluation of the Potential Energy Surface (CREPES) is a tool used to generate the iterative combinations of rotations of functional groups. (A description of the CREPES method can be found in chapter 3). A complete CREPES evaluation of DMA was performed. The minima structure was used for the subsequent stepwise configurations. Optimizations on the subsequent reaction steps were iterated over the rotations associated with the added boron containing moiety.

Coarse screening was completed at RHF/3-21G(d) PCM with 20 iteration steps and step sizes of 0.1 bohr. Fine screening was completed for structures within a range of 0.165-7 RT of the minimum structure. Fine screening energies were calculated at MP2/6-31G(d) PCM. Final optimizations were done at MP2/6-31++G(d) (and MP2/cc-pTVZ). CREPES was utilized to identify the boric and borate binding minima. The configurations for the single bound (DMA-X-BR), chelated (DMA=BR), and dimer (dDMA-X-BR) minima were identified using the CREPES methodology (where R= OH or $(OH)_2$ and X = the binding site O(2) or O(3)).

Scatter plots of the conformational energy profile are shown in the figures throughout the results section. Most of these figures are the truncated representations of the complete CREPES determined conformational energy profile. Very high energy configurations are often determined by CREPES. For the sake of visibility (afforded by limiting the range of energies displayed) of the more pertinent data these data points are typically excluded. The maximum energy for selected configurations for fine screening is shown with the red line in the density of state diagrams.

All stationary state structures derived from optimization calculations were determined to be minima by positive definite hessian matrix. Transition states (saddle point calculations) Hessian matrices were confirmed to have only one negative. Polarizable continuum mode (PCM) was used as a water solvent at all levels of calculation; this was done to mimic the in vivo environment**.** Thermodynamic properties were generated from the full vibrational frequency analysis. All calculations were performed using the GAMESS software package. Hartree-Fock and MP2 thermodynamics were scaled by the normal frequency scaling factors of 0.89 and 0.95 respectively. Estimates for equilibrium constants were computed from free energy using $\Delta G = -RT \ln K_{eq}$.

Complete Translation for the Evaluation of the Potential Energy Surface

Identifying transition states for large systems (the dimers have as many as 26 heavy atoms and 26 hydrogens) can be computationally very expensive. Complete hessians need to be calculated several times for each transition state search. The large number of atoms and because the possible geometries that must be searched are in an area where PES curvature is complicated it's not just expensive, it's not always fruitful. In addition in GAMESS there is no analytic hessian available for either MP2 or PCM so numerical methods increase the computational time required. Consequently, simply allowing GAMESS to search for a transition state without careful consideration is a poor use of resources.

Complete Translation for the Evaluation of the Potential Energy Surface (CTEPES) is a tool created to help automate the generation of GAMESS input files. CTEPES allows users to define parameters for the translation of functional groups. The use of the z-matrix coordinate system is required. The input parameters are, moving atom, step length, and

length of total translation. A properly formatted z-matrix is critical to the translation of atoms to ensure no unexpected atom movements are occurring. All iterative combinations of translations are computed. Output files are labeled by explicit inter-nuclear distances as generated by CTEPES. CTEPES like CREPES only works on the $DATA group of the GAMESS input file and consequently is capable being used at any desired level of theory.

For the purposes of this investigation CTEPES was used to assist in the mapping of the potential energy surface to attempt to locate transition states. CTEPES calculations were run at RHF/6-31++G(d). CTEPES is useful for rapidly mapping two dimensions of the PES simultaneously.

Nomenclature

The wide variety of available configuration of saccharides causes a difficulty with nomenclature in describing configurations of DMA. Chapter 3 nomenclature section contains a more complete description of the nomenclature that will be used. A brief recapitulation will be outlined here.

Saccharide hydroxyl groups each have three expected configurations gauche(-), gauche(+), and trans (**ḡ**, **g**, and **t** respectively). The symbol indicates the dihedral angle H-O-C(n)-C(n-1) relationship (or the ring oxygen in the anomeric hydroxyl case). **ḡ** denotes a clockwise (viewed O to C) rotation; **g** indicates a counter clockwise rotation and **t** the anti position. The hydroxylated methyl group is indicated with a capital representation of this symbol (**Ḡ**, **G** and **T** they are sometimes referred to as gg, gt, and tg respectively). As the capping methyl groups are highly symmetric the only two configurations described are eclipsed and staggered referred to hereafter as **e** and **s** respectively. DMA configuration

naming will be ordered, Methyl(O(1)), O(1), O(2), O(3), C(3'), O(3'),Methyl(O(3')) (Figure 4.2)

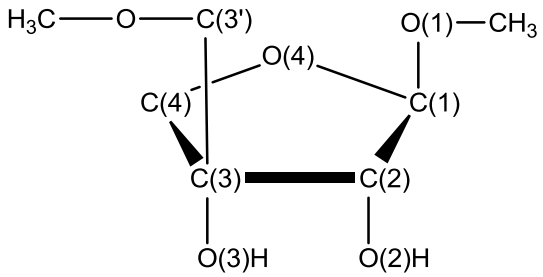


Figure 4.2 Configurationally important elements of dimethyl-apiose

Results

Total Reaction

The reported structure of Rhamnogalacturonan –II dimer contains a four-coordinate borate crosslink. While the final product has with a four-coordinate borate cross-link, it is not immediately apparent at which point in the process the boron adds the fourth bond. Therefore the dimerization of RG-II would be expected to dimerize via one of two overall reactions. Either binding to boric acid or the borate ion. Using dimethylapiose to represent this one would expect the following over all reactions. A one step reaction is not expected. The individual steps towards the final products will be described here with a summary of the complete reaction at the end.

$2\ DMA + B(OH)_3 \leftrightarrow DMA{=}B{=}DMA^- + 2H_2O + H_3O^+$

$2\ DMA + B(OH)_4^- \leftrightarrow DMA{=}B{=}DMA^- + 4H_2O$

Boric Acid/Borate

Boric acid (Figure 4.3) has been shown to be a planar molecule; the low energy configuration has all hydroxyls rotated anti to each other. Borate ion's low energy configuration of is shown in Figure 4.4; over 20,000 (12^4) configurations were computed to confirm that this is low energy configurations. Figure 4.5 displays the conformational energy profile of all the configurations generated. The density of states was so high that the lowest 30 configurations were selected for fine screening. This results in the selection of configurations within 0.165 of RT. Any configuration within RT is readily available, however the large number of conformers within RT made calculating all within RT impractical. As so many configurations are low lying no one geometry is experimentally known. Many of the CREPES generated structures for the borate ion are symmetrically equivalent. A non-symmetric starting point helps ensure that a larger portion of the PES is sampled while using CREPES. In addition borate ion is the only structure being studied that is small and is this symmetric, so thoroughness is of value. The geometry of the minimum energy configuration for boric acid and borate were used as starting point geometries for all subsequent optimizations.

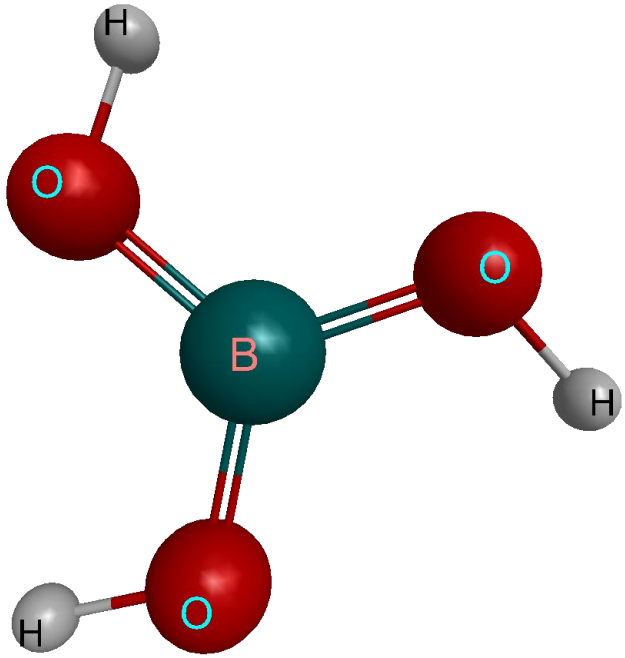


Figure 4.3 Boric acid, $B(OH)_3$

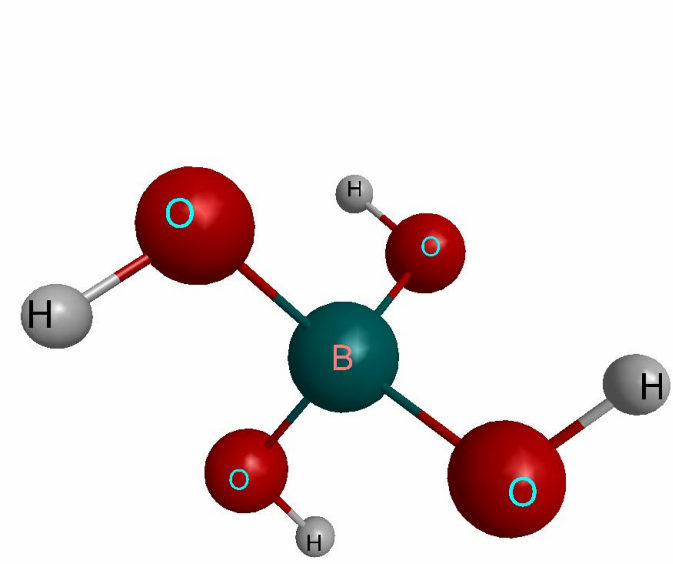


Figure 4.4 Borate ion, $B(OH)_4^-$

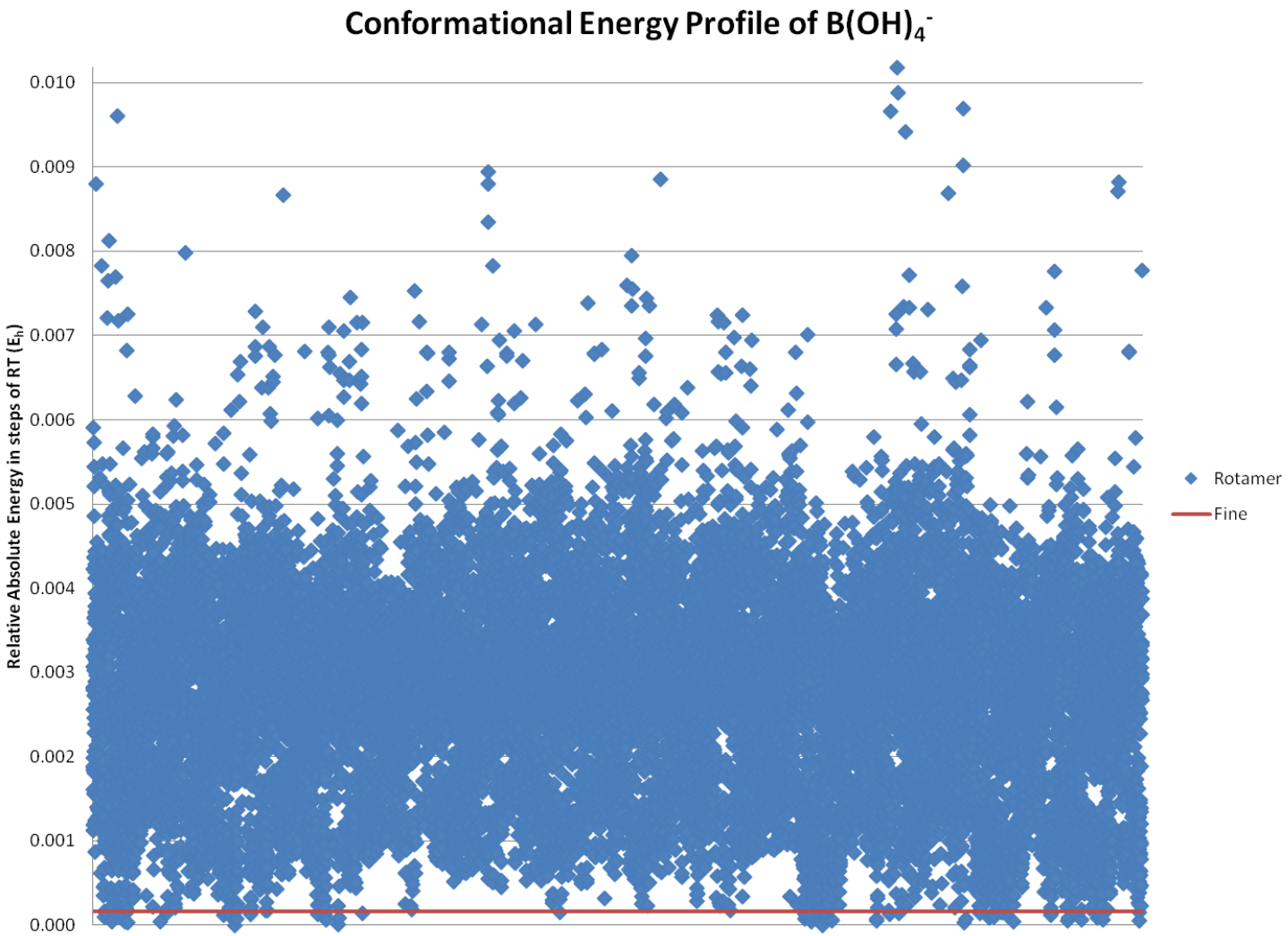


Figure 4.5 Conformational energy profile of $B(OH)_4^-$ modeled by bulk screening energies
(RT is approximately equal to $1mE_h$)

Dimethyl-Apiose

The Dimethyl-D-β-Api*f* conformational energy profile is shown in Figure 4.6. The conformers appear to have clustering due to the method for which they are plotted. In this figure the second half of the configurations indicates that a methyl group is rotated to the eclipsed position. As one moves from left to right comparing halves a mirroring effect can be observed with small variations and an upward shift to the configurations with the added energy of the eclipsed position. With careful visual inspection other trends about other rotations can be observed.

Dimethyl-D-β-Api*f* minimum energy configuration found by CREPES is **sḡtgḠts (**Figure 4.7**)**. In this configuration the methoxy group on C(3') interferes with boric and borate interactions with O(3). The configuration that was used in all subsequent calculations was **sḡttTts (**Figure 4.8**)**. This structure was used as a starting geometry for all further geometry searching.. **sḡttTts** while higher in energy by 6.75kJ·mol^{-1} than **sḡtgḠts** it is more open to the interactions we would expect. In addition, when considering free energy, **sḡttTts** is only higher than **sḡtgḠts** by 1.858 kJ·mol^{-1}, which is within RT at 298K (2.48 kJ·mol^{-1}).

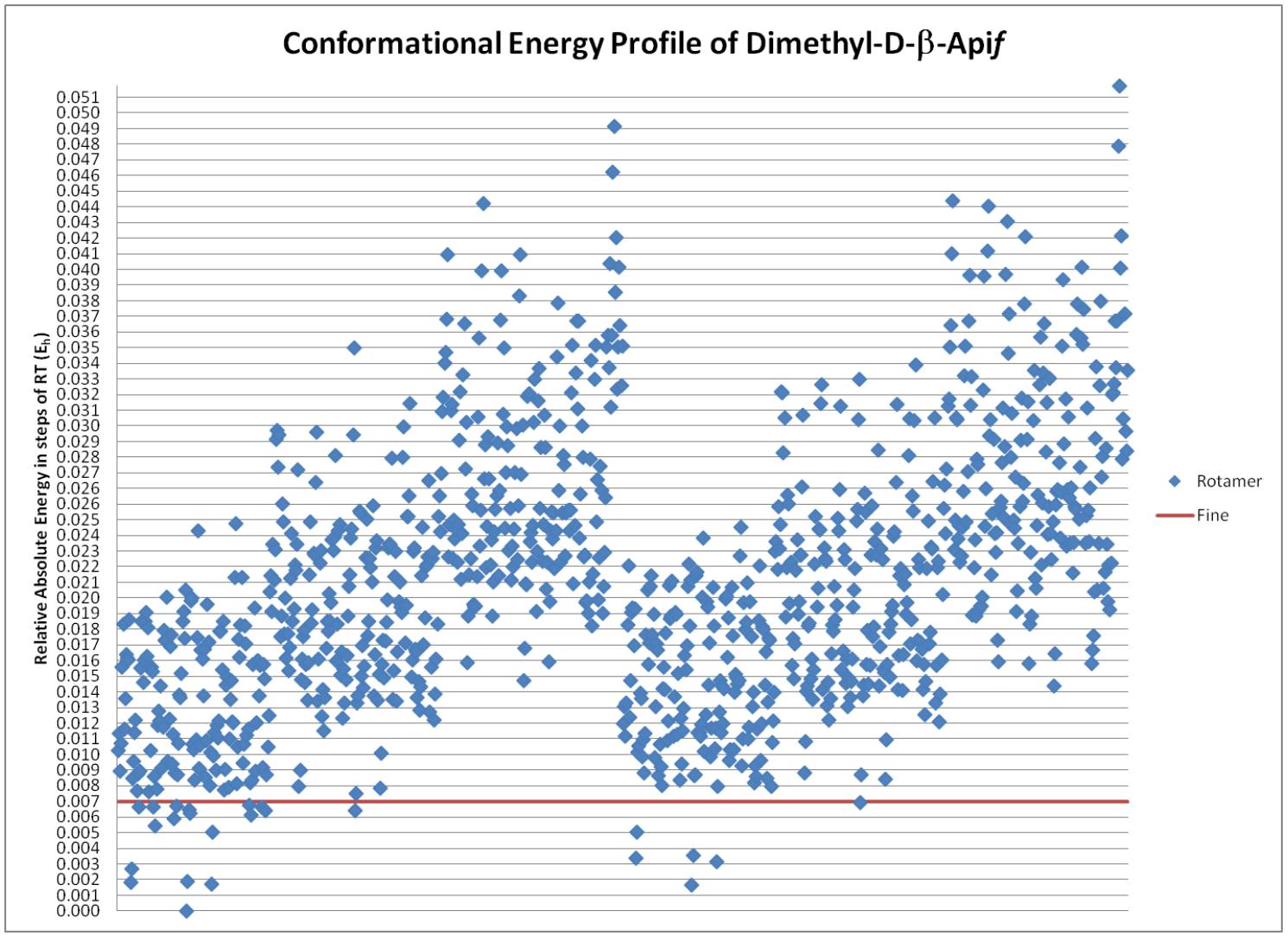


Figure 4.6 Conformational energy profile of Dimethyl-D-β-Api*f* modeled by bulk screening energies
(RT is approximately equal to $1mE_h$)

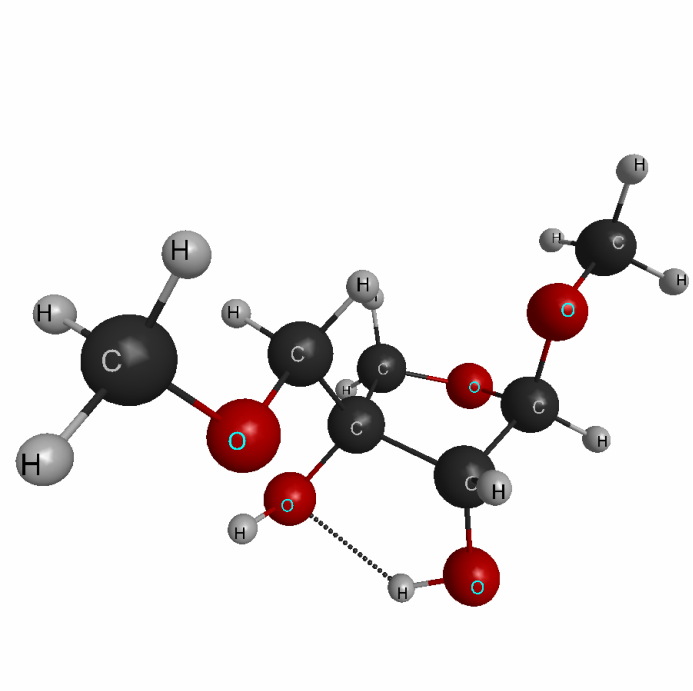


Figure 4.7 Dimethyl-D-β-Api*f* sḡtgḠts

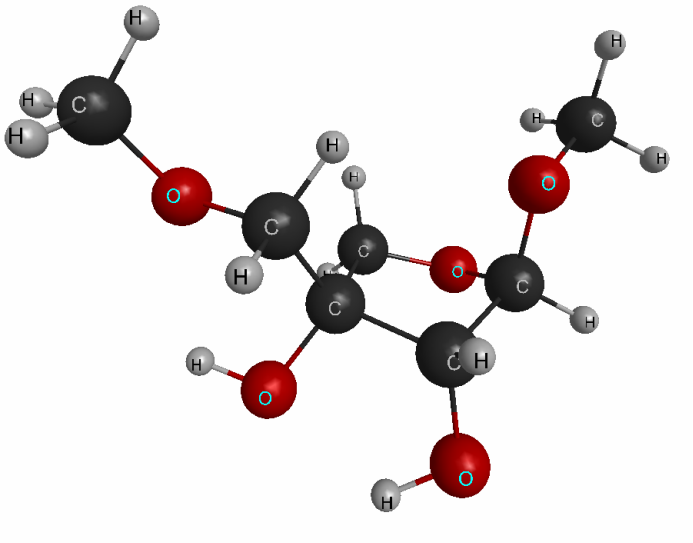


Figure 4.8 Dimethyl-D-β-Api*f* sḡttTts

Boron Binding

The first step towards the boron crosslink is a dehydration reaction binding boric acid or borate at either position O(2) or O(3) to DMA. DMA-O(2)-$B(OH)_2$ and DMA-O(3)-$B(OH)_2$ are the resulting products, their minimum energy conformers are diagramed in Figure 4.9 and Figure 4.10 respectively. Boron bound at the O(3) position is stabilized by intramolecular interaction namely a hydrogen bond as can be seen in the figure. Borate ion can also bind at either position O(2) or O(3) to DMA. Figure 4.11 and Figure 4.12 are the minimum energy configurations from the CREPES of DMA-O(2)-$B(OH)_3^-$ and DMA-O(3)-$B(OH)_3^-$. Again intermolecular inaction stabilizes the O(3) bound boron compound , this time with hydrogen bonding to the adjacent –O(2)H Reaction energies are shown in Table 4.1 Only one series of transition states was calculated. Overall energetic results and barrier heights will be discussed more in the discussion section.

Table 4.1 First bond, Boron binding, Reaction Energetics, Internal energy barrier, Absolute energy reaction energy, Enthalpy, Free energy ($kJ \cdot mol^{-1}$)

First bond, Boron binding						Barrier	ΔU_e	ΔH^{298}	ΔG^{298}
DMA	+	$B(OH)_3$	↔	DMA-O2-$B(OH)_2$	+ H_2O		-8.02	-9.2	-2.2
DMA	+	$B(OH)_3$	↔	DMA-O3-$B(OH)_2$	+ H_2O	192.73	-13.34	-13.7	-4.9
DMA	+	$B(OH)^-_4$	↔	DMA-O2-$B(OH)^-_3$	+ H_2O		-19.65	-18.6	-3.2
DMA	+	$B(OH)^-_4$	↔	DMA-O3-$B(OH)^-_3$	+ H_2O		-30.52	-29.8	-13.1

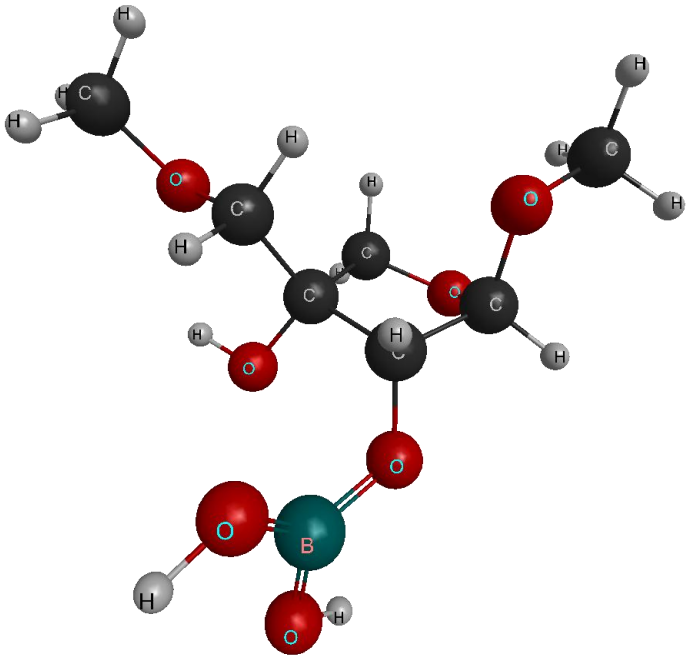


Figure 4.9 DMA-O(2)-B(OH)$_2$

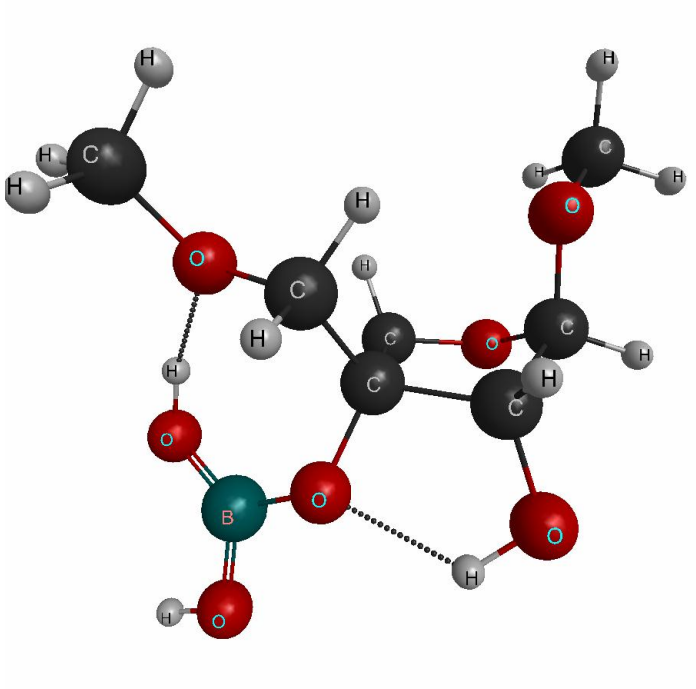


Figure 4.10DMA-O(3)-B(OH)$_2$

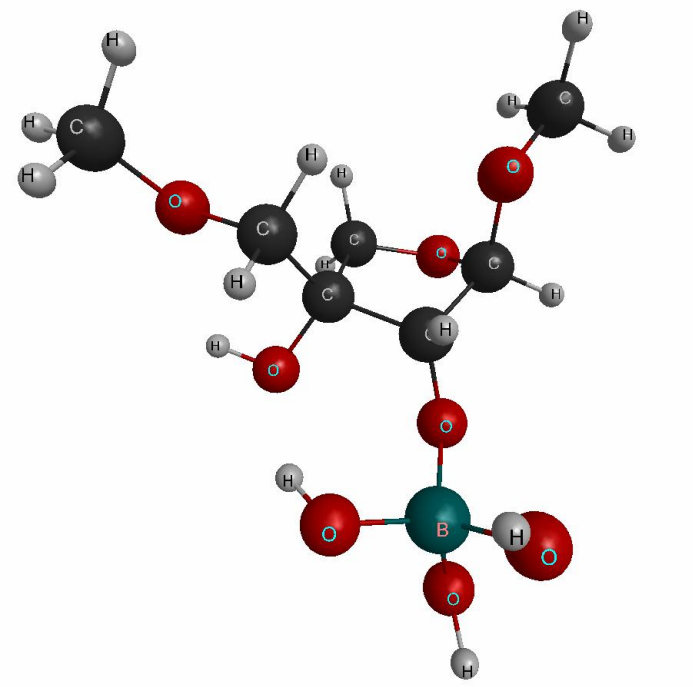


Figure 4.11 DMA-O(2)-B(OH)$_3$

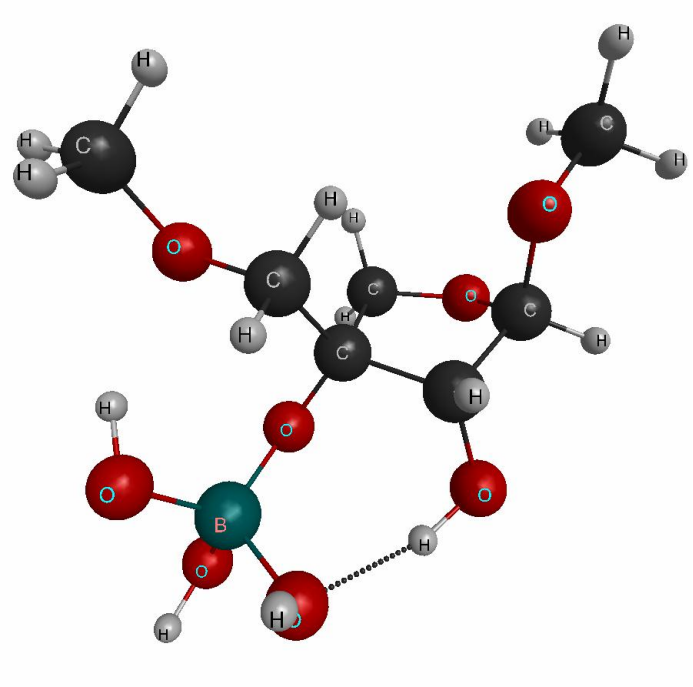


Figure 4.12 DMA-O(3)-B(OH)$_3$

First Chelation

After the boron moiety is bound to DMA a dehydration chelation reaction occurs to the adjacent hydroxyl group. Boron makes strong bonds to 2-3 erythro-furanoses. It is readily apparent from the free energies presented here that apiose is no exception (Table 4.2). This reaction is has a large negative free energy and is likely a driving force in boric/borate bonding. Figure 4.13(DMA-O(2,3)-B(OH)) and Figure 4.14(DMA-O(2,3)-B(OH)$_2$)show the structures of the chelated species.

In organic chemistry when one atom deviates from planarity in a five member ring the conformation is referred to as envelope. To investigate the possible occurrence of this structure in boronated-DMA, DMA-O(2,3)-B(OH), the boron containing five member ring was bent to both envelop conformers (boron being the moving portion) and the planar configuration. Whenever one of these envelope conformers is allowed to optimize it invariably will return to a planar configuration. As a result of this finding the planar ring configuration was chosen for the borate case. Hydroxyl configurations bound to the boron were chosen to match optimized boric and borate structures.

Table 4.2 Second Bond, First Chelation, Reaction Energetics, Absolute energy barrier, Absolute energy reaction energy, Enthalpy, Free energy (kJ·mol^{-1})

First chelation				Barrier	ΔU_e	ΔH^{298}	ΔG^{298}
DMA-O2-B(OH)$_2$	↔	DMA=B(OH)	+ H_2O		1.12	-6.2	-48.4
DMA-O3-B(OH)$_2$	↔	DMA=B(OH)	+ H_2O	197.33	6.43	-1.7	-45.8
DMA-O2-B(OH)$_3$	↔	DMA=B(OH)$^-_2$	+ H_2O		-3.32	-12.4	-59.1
DMA-O3-B(OH)$_3$	↔	DMA=B(OH)$^-_2$	+ H_2O		7.55	-1.3	-49.1

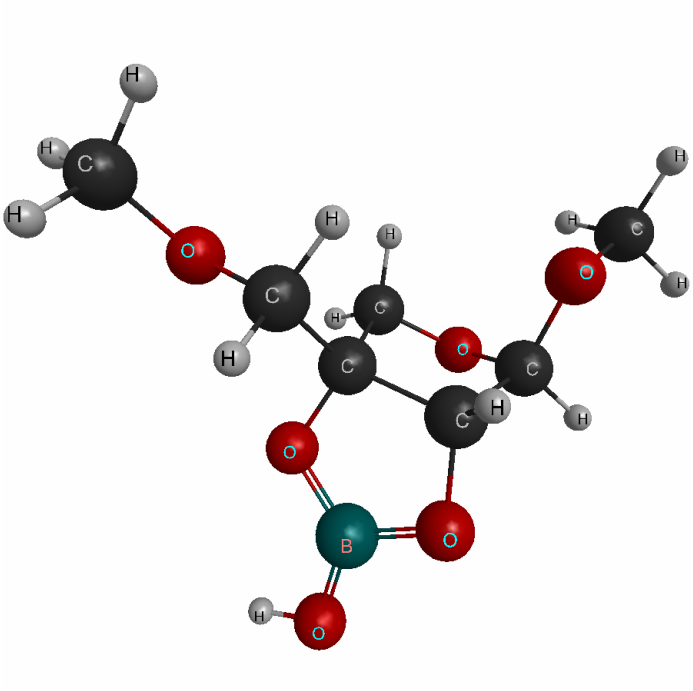


Figure 4.13 DMA-O(2,3)-B(OH)

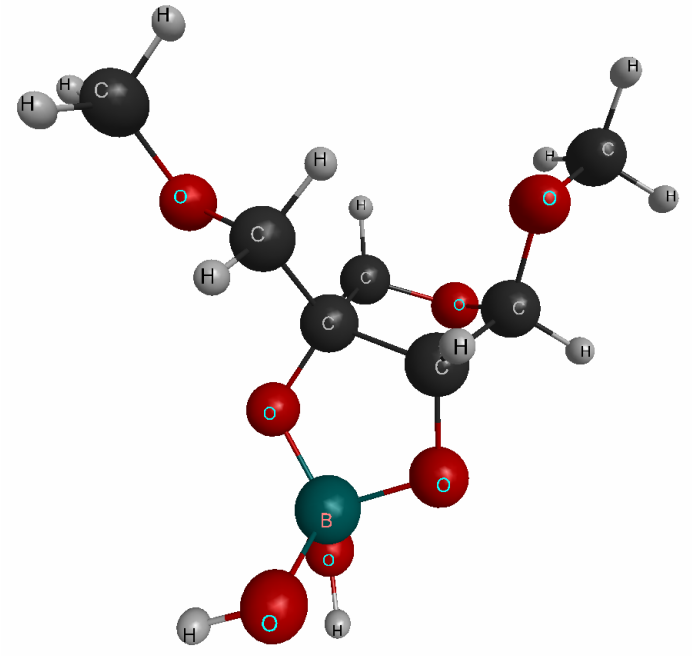


Figure 4.14 DMA-O(2,3)-B(OH)$_2$

Dimerization

The formation of the third boro-ester creates the dimer. A structure that was different from the expected configuration was found for DMA-O(2,3)-B-O(2'')-DMA. Optimization at low levels of theory of DMA-O(2,3)-B-O(2'')-DMA identified a barrier free transition to DMA-O(2,3)-B-O(2'',3'')-H-DMA (a protonated double chelated structure)(Figure 4.15) this structure maintained stability when fine screening levels of theory are applied. To try and observe the transition state for deprotonation of this species CTEPES was applied to monitor the PES as deprotonation occurred via H_2O. The PES is displayed in Figure 4.16. This is a barrier free transition to a higher energy state.

Postulating that the second chelation was a barrier free process, similar structures were sought for the O(3'') configuration, however no stationary state structures could be found for as protonated second chelation for the O(3'') bound dimer. Additional structural stability for the O(2'') H conformation appears to be dependent on the two proximal oxygens in the local environment.

A search for stable 3-coordinate dimers was performed on to find a consistent O(2'') configuration. Figure 4.17 and Figure 4.18 show the optimized 3 coordinate structures found for boric binding at O(2'') and O(3'').

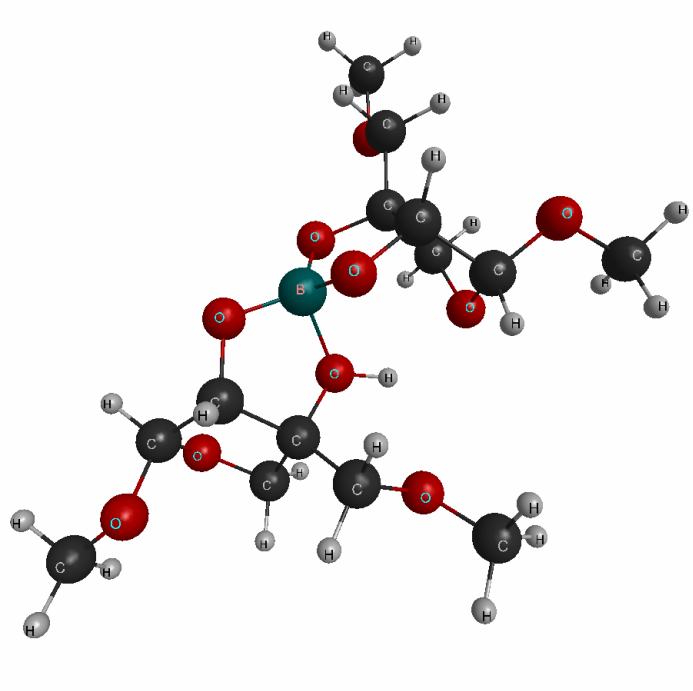


Figure 4.15 DMA-O(2,3)-B-O(2'',3'')-H-DMA

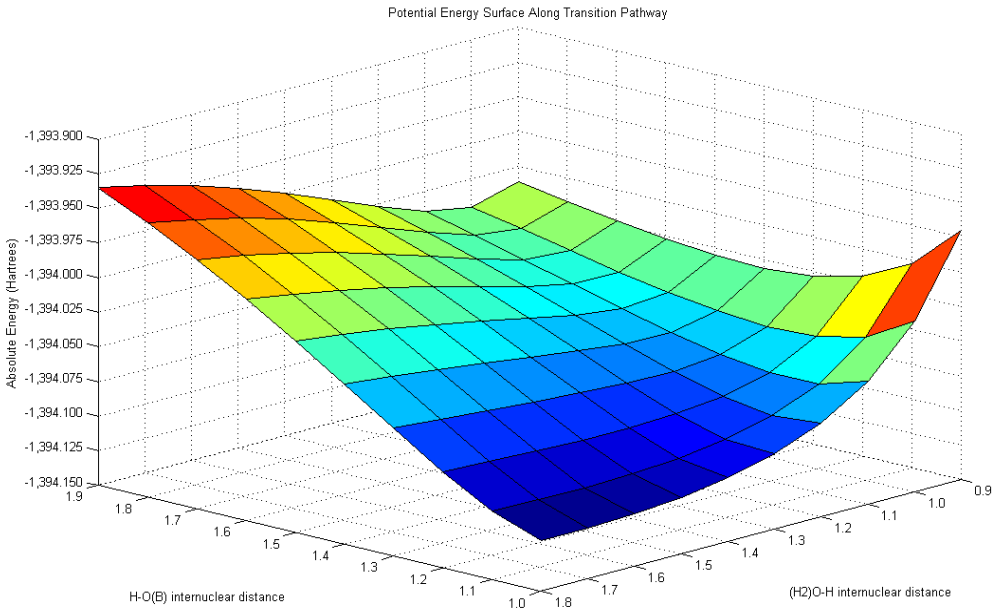


Figure 4.16 DMA-O(2,3)-B-O(2'',3'')-H-DMA Potential Energy surface to deprotonation

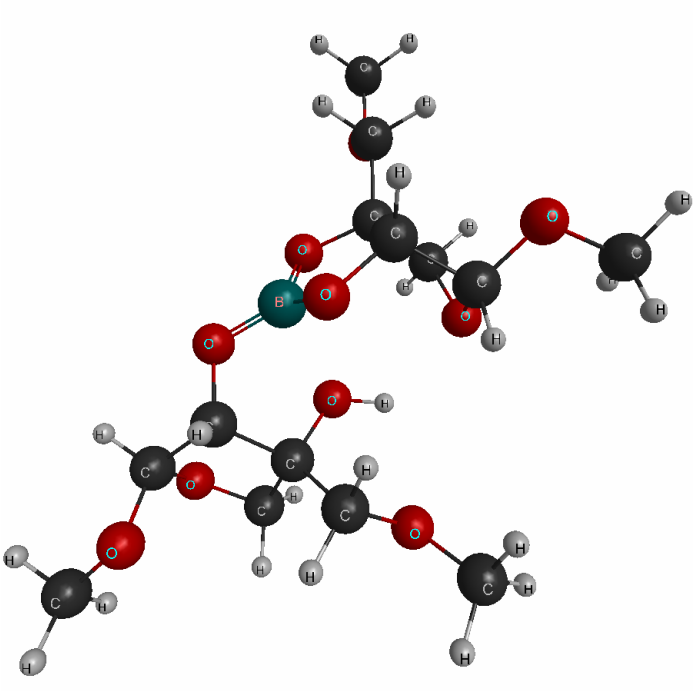


Figure 4.17 DMA-O(2,3)-B-O(2'')-DMA

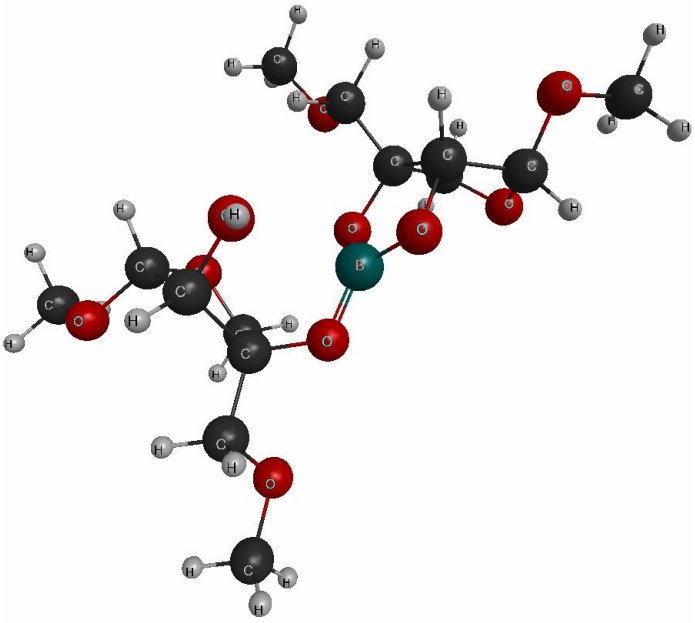


Figure 4.18 DMA-O(2,3)-B-O(3'')-DMA

DMA-O(2,3)-B(OH) -O(2'')-DMA and –O(3'')-DMA have low energy configurations with a rotation of the second DMA anti to a configuration that would be able to perform the second chelating dehydration reaction. In other words, the most stable configuration with 3 B-O bond borate is stereochemically hindered from forming a 4th B-O bond. Scripts were generated to identify structures with the stereochemically appropriate orientation for further reactions. Configurations with this orientation rather than the lowest lying energetically were selected as fine screening configurations. They are shown in Figure 4.19 and Figure 4.20 respectively. The minimum structures in each case were within 4.2kJ·mol^{-1} of each other (at fine screening conditions) the appropriate orientation models have one less intermolecular hydrogen bond (Table 4.3).

CREPES screening of rotational conformer of dimers is very resource intensive so only one CREPES screening was done for each of boric and borate binding. The identified low energy conformers were not the stereochemically appropriate conformer. Therefore the alternate configurations we simply selected for their similarity to the stereochemically appropriate counterparts on the opposite hydroxyl.

Table 4.3 Third bond, Dimerization, Reaction Energetics, Absolute energy barrier, Absolute energy reaction energy, Enthalpy, Free energy (kJ·mol^{-1})

Dimerization					Barrier	ΔU_e	ΔH^{298}	ΔG^{298}
DMA=B(OH)	+ DMA	↔	DMA=B-O2-DMA	+ H_2O		-25.22	1.12	-6.2
DMA=B(OH)	+ DMA	↔	DMA=B-O3-DMA	+ H_2O	207.33	-6.98	6.43	-1.7
DMA=B(OH)$^-_2$	+ DMA	↔	DMA=B(OH)-O2-DMA$^-$	+ H_2O		-35.33	-3.32	-12.4
DMA=B(OH)$^-_2$	+ DMA	↔	DMA=B(OH)-O3-DMA$^-$	+ H_2O		-36.90	7.55	-1.3

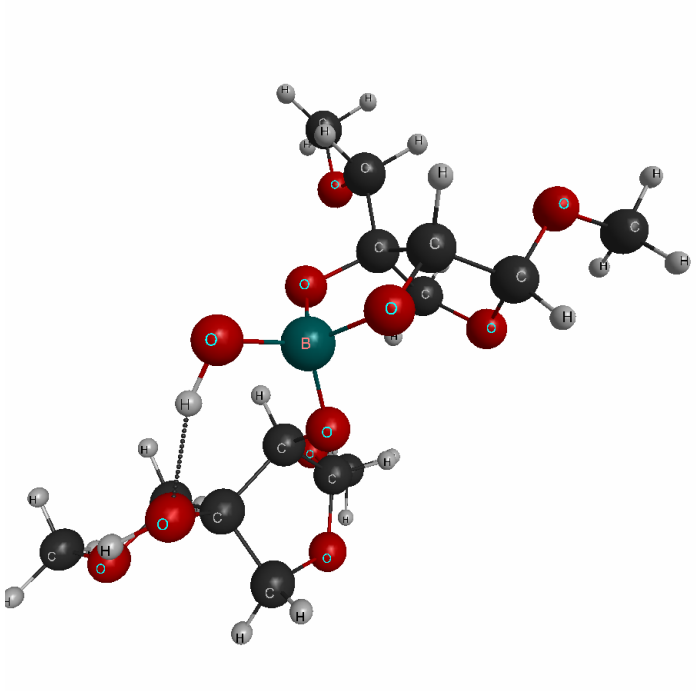


Figure 4.19 DMA-O(2,3)-B(OH)-O(2'')-DMA

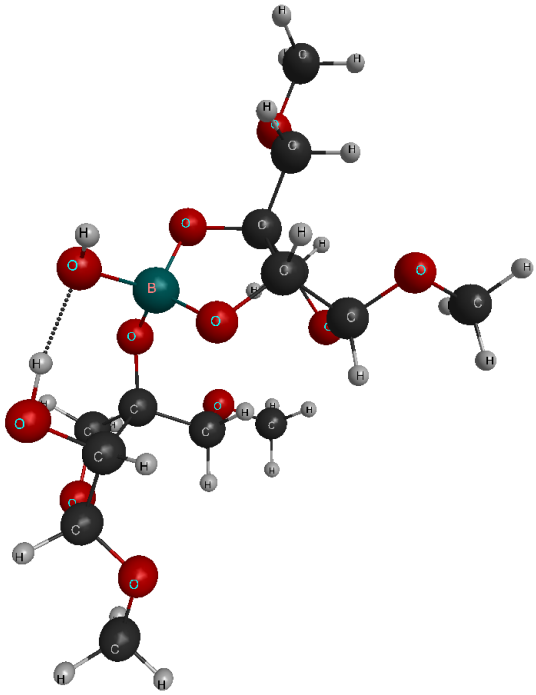


Figure 4.20 DMA-O(2,3)-B(OH)-O(3'')-DMA

Since DMA-O(2,3)-$B(OH)_2$ can bond at either –OH to either O(2'') or O(3'') and both DMA-O(2,3)-BOH and DMA-O(2,3)-$B(OH)_2$ can rotate freely, the number of iterative combinations of binding sites and R and S configurations is too large to completely explore. An arbitrary selection has been chosen to represent the system.

Second Chelation

The final step in dimerization is the second chelation reaction. For the boric acid this is a deprotonation reaction and for the borate this is a dehydration reaction. The structures of the R and S configurations of dDMA-B are displayed in Figure 4.21 and Figure 4.22. Reaction energies for the creation of the fourth bond are presented in Table 4.4

Table 4.4 Fourth Bond, Second chelation reaction energetics, Absolute energy reaction energy, Enthalpy, Free energy (kJ·mol^{-1})

Fourth Bond, Second chelation					ΔU_e	ΔH^{298}	ΔG^{298}
DMA=B-O(2)-DMA	+ H2O	↔	DMA=B(R)=DMA	+ H_3O^+	140.30	143.0	151.1
DMA=B-O(2)-DMA	+ H2O	↔	DMA=B(S)=DMA	+ H_3O^+	138.64	140.0	144.7
DMA=B-O(3)-DMA	+ H2O	↔	DMA=B(R)=DMA	+ H_3O^+	122.06	123.7	133.3
DMA=B-O(3)-DMA	+ H2O	↔	DMA=B(S)=DMA	+ H_3O^+	120.40	120.7	126.9
DMA=B(OH)-O2-DMA-		↔	DMA=B(R)=DMA	+ H_2O	-7.49	-5.5	-7.49
DMA=B(OH)-O2-DMA-		↔	DMA=B(S)=DMA	+ H_2O	-9.14	-6.8	-9.14
DMA=B(OH)-O3-DMA-		↔	DMA=B(R)=DMA	+ H_2O	-5.92	-6.1	-5.92
DMA=B(OH)-O3-DMA-		↔	DMA=B(S)=DMA	+ H_2O	-7.58	-7.4	-7.58

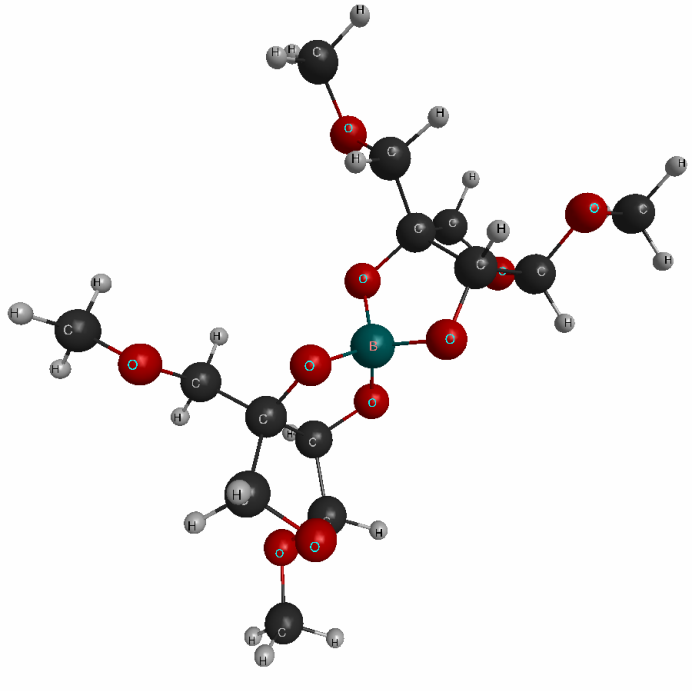


Figure 4.21 DMA-O(2,3)-B(R)-O(2’’,3’’)-DMA

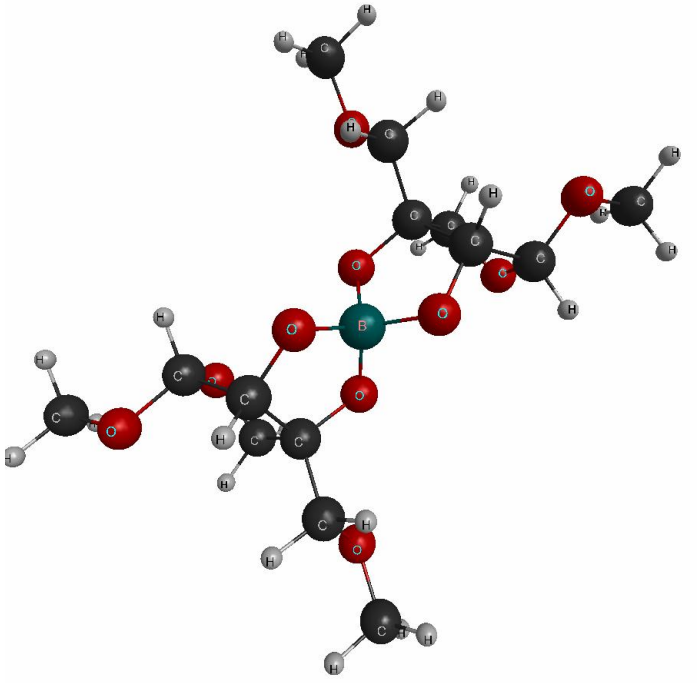


Figure 4.22 DMA-O(2,3)-B(S)-O(2’’,3’’)-DMA

Total reaction summary

The total reaction pathway is described in the diagrams below. Experimentally mRG-II (monomer) and dRG-II-B (dimer) have been observed. Utilizing our modeled system one would expect a stepwise reaction; first, a one sided bonding to boron by DMA at either O(2) or O(3) (Figure 4.23), second, the first chelation (Figure 4.24), third, dimerization at either O(2'') or O(3'') (Figure 4.25) finally the second chelation (Figure 4.26) to either R or S dimer. Products and reactants from these figures are color coordinated to assist in corroboration from diagram to diagram. The individual reactant absolute energies are shown in Table 4.5 they are presented for completeness. The stepwise reaction steps energetics are summarized in Table 4.6.

Table 4.5 Absolute energies of stationary state molecules

Reactants	U_e (E_h)	Dimerization	U_e (E_h)
H_2O	-76.2215275381	DMA=B-O(2)-DMA	-1321.8809728281
H_3O^+	-76.6094564224	DMA=B-O(3)-DMA	-1321.8740249654
$B(OH)_3$	-251.8274496443	DMA=B(OH)-O(2)-DMA$^-$	-1397.6582838350
$B(OH)^-_4$	-327.5947890960	DMA=B(OH)-O(3)-DMA$^-$	-1397.6588800063
DMA	-649.3529351622		
Boron binding		Boric transition states	
DMA-O(2)-$B(OH)_2$	-824.9619137288	TS1	-901.1069790159
DMA-O(3)-$B(OH)_2$	-824.9639381935	TS2	-824.8887798792
DMA-O(2)-$B(OH)^-_3$	-900.7336827183	TS3	-1398.0139283242
DMA-O(3)-$B(OH)^-_3$	-900.7378205430		
Chelation		Double chelation	
DMA=B(OH)	-748.7399601919	DMA=B(R)=DMA	-1321.4396072906
DMA=$B(OH)^-_2$	-824.5134188477	DMA=B(S)=DMA	-1321.4402386436

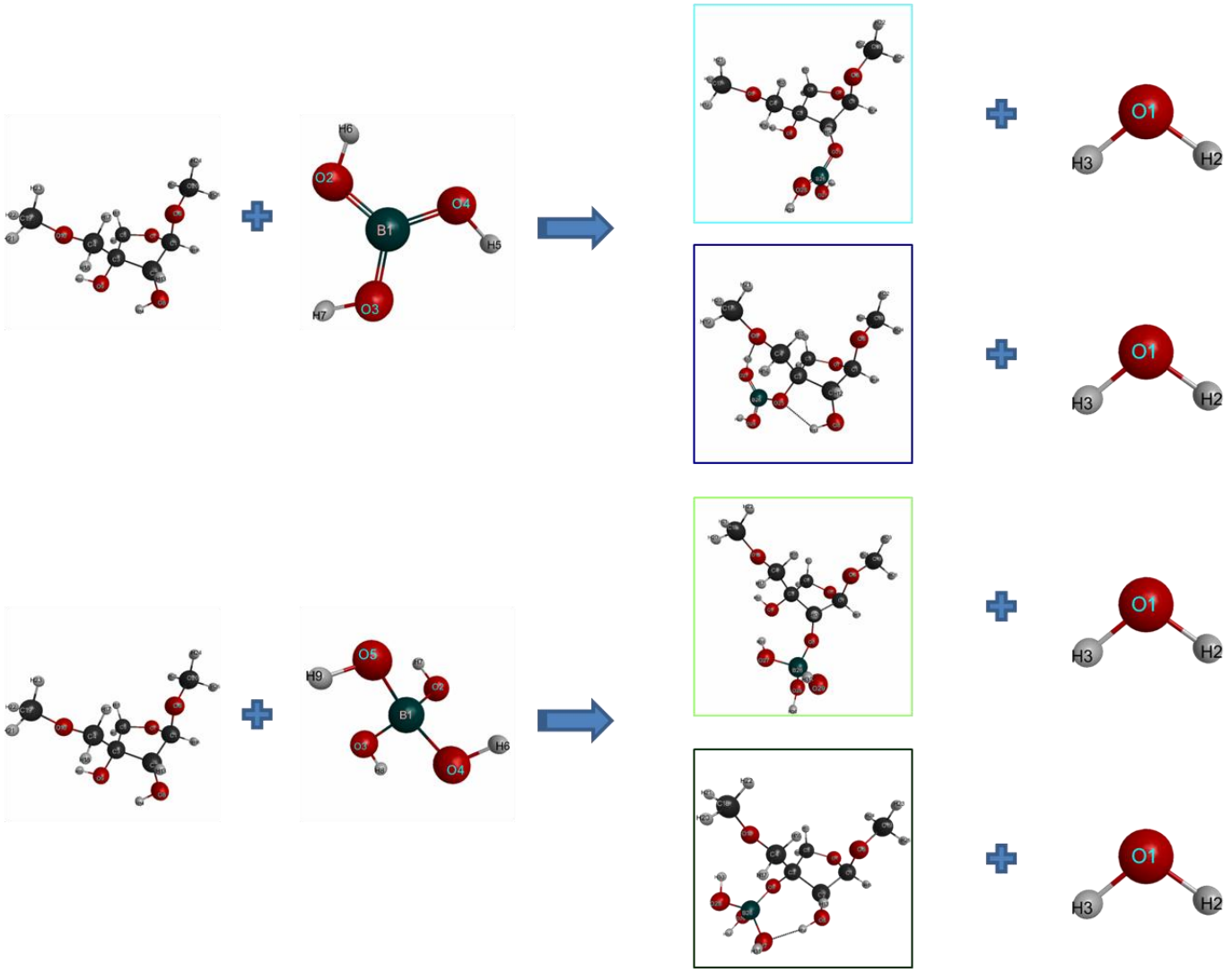


Figure 4.23 Bond 1, Boron binding

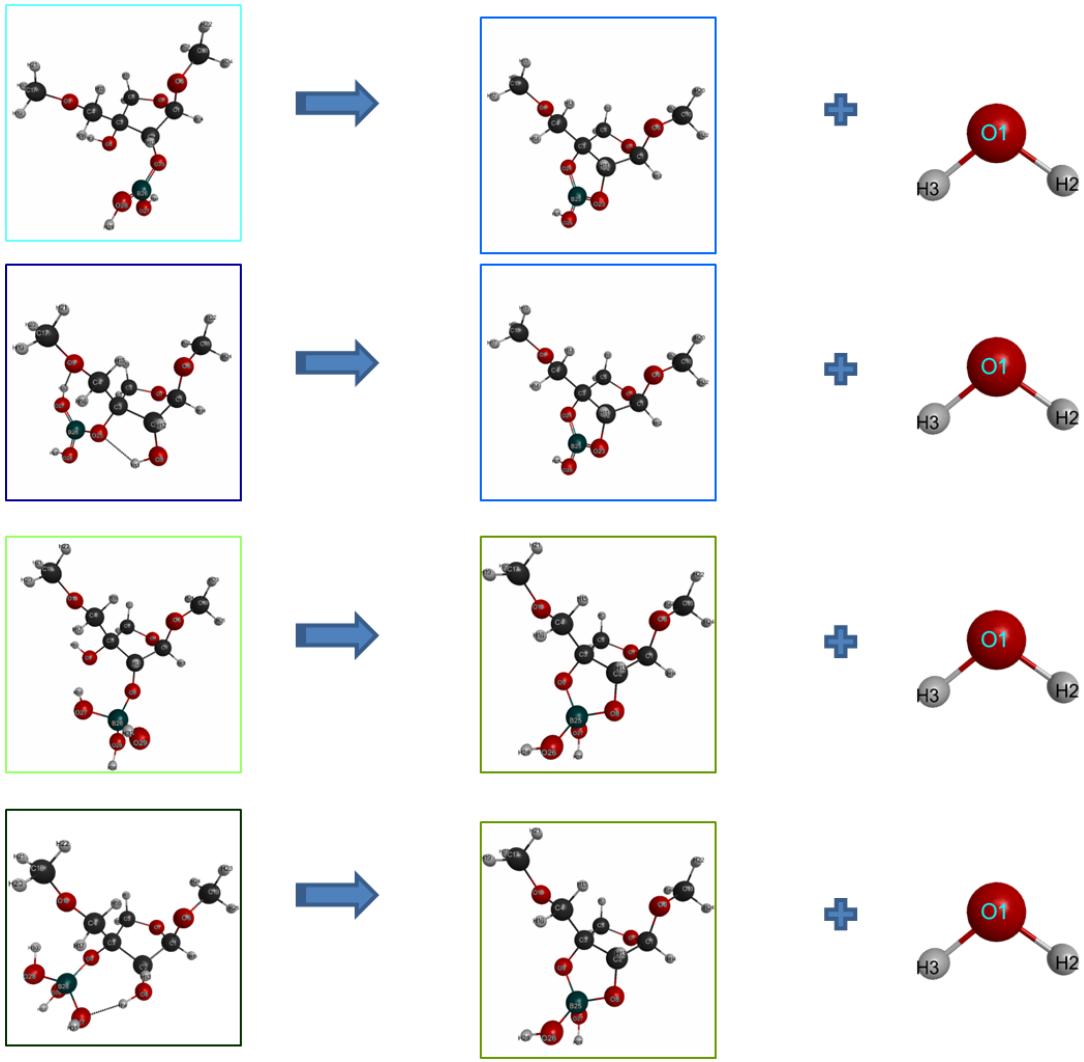


Figure 4.24 Bond 2, First chelation

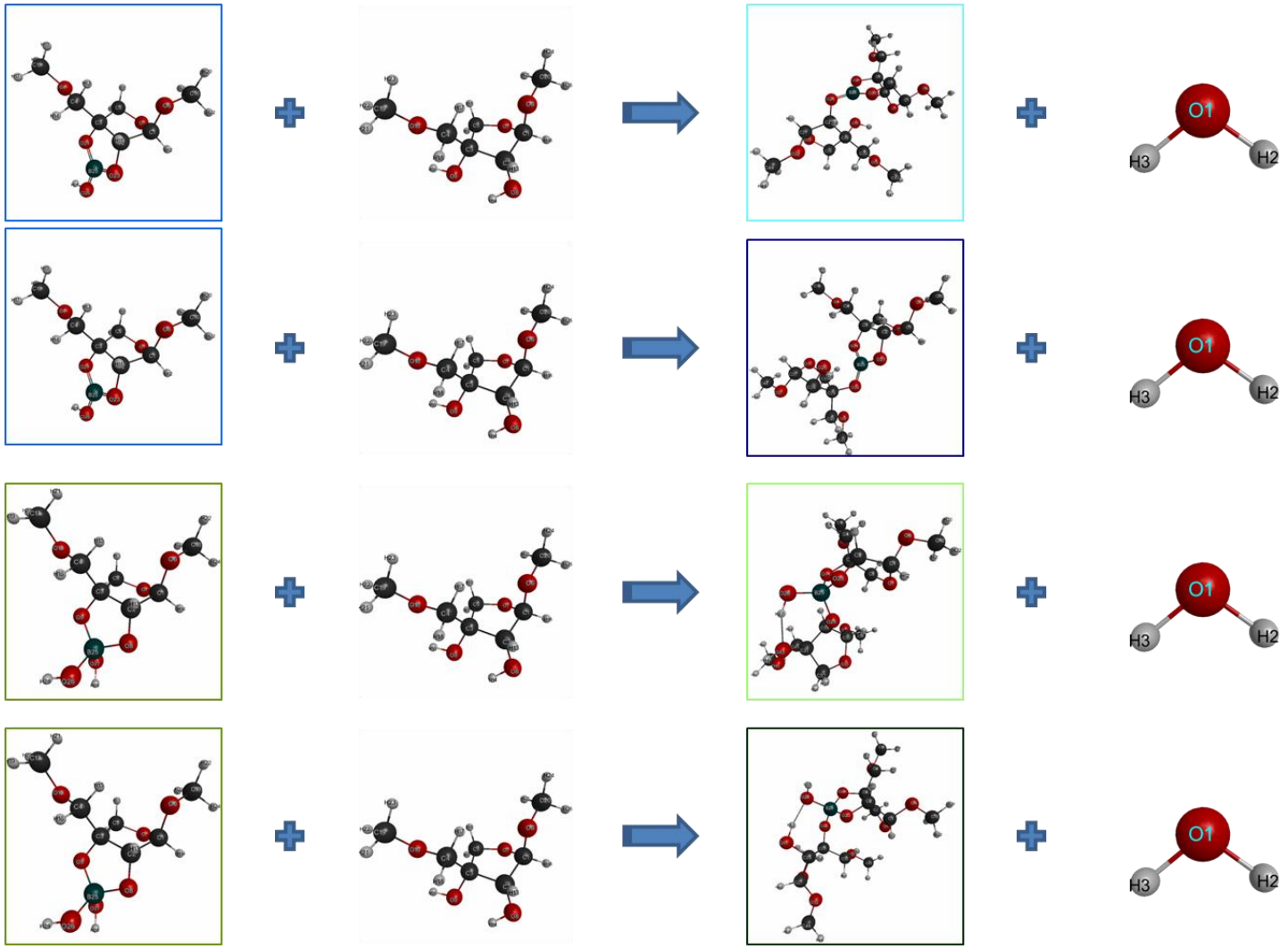


Figure 4.25 Bond 3, Dimerization

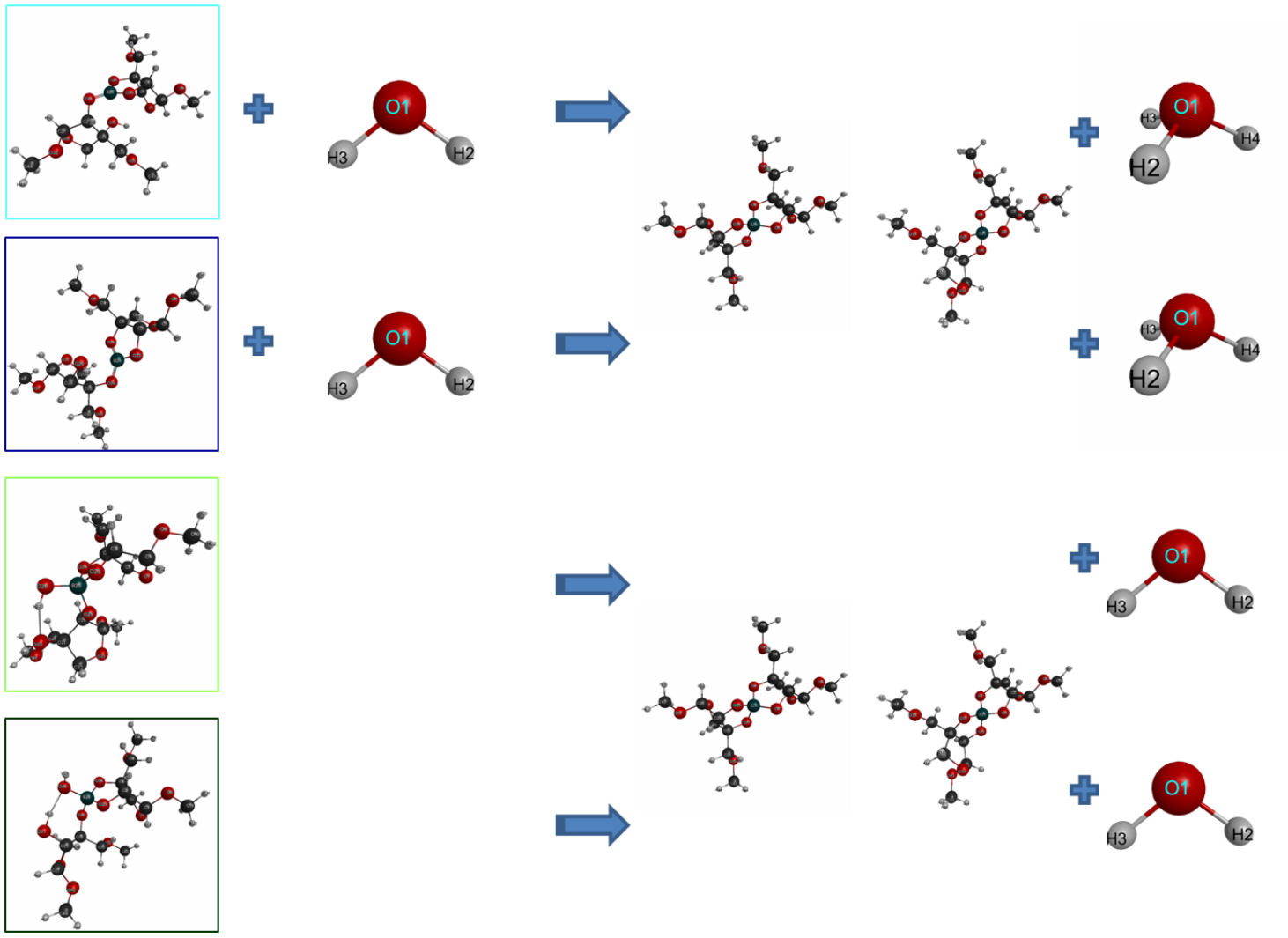


Figure 4.26 Bond 4, Second chelation

Table 4.6 Overall reaction energetics summary, absolute energy barrier, absolute energy reaction energy, enthalpy, free energy (kJ·mol^{-1})

Boron binding					Barrier	ΔU_e	ΔH^{298}	ΔG^{298}
DMA	+ $B(OH)_3$	↔	DMA-O2-$B(OH)_2$	+ H_2O		-8.02	-9.2	-2.2
DMA	+ $B(OH)_3$	↔	DMA-O3-$B(OH)_2$	+ H_2O	192.73	-13.34	-13.7	-4.9
DMA	+ $B(OH)^-_4$	↔	DMA-O2-$B(OH)^-_3$	+ H_2O		-19.65	-18.6	-3.2
DMA	+ $B(OH)^-_4$	↔	DMA-O3-$B(OH)^-_3$	+ H_2O		-30.52	-29.8	-13.1
First chelation								
DMA-O2-$B(OH)_2$		↔	DMA=B(OH)	+ H_2O		1.12	-6.2	-48.4
DMA-O3-$B(OH)_2$		↔	DMA=B(OH)	+ H_2O	197.33	6.43	-1.7	-45.8
DMA-O2-$B(OH)_3$		↔	DMA=$B(OH)^-_2$	+ H_2O		-3.32	-12.4	-59.1
DMA-O3-$B(OH)_3$		↔	DMA=$B(OH)^-_2$	+ H_2O		7.55	-1.3	-49.1
Dimerization								
DMA=B(OH)	+ DMA	↔	DMA=B-O2-DMA	+ H_2O		-25.22	1.12	-6.2
DMA=B(OH)	+ DMA	↔	DMA=B-O3-DMA	+ H_2O	207.33	-6.98	6.43	-1.7
DMA=$B(OH)^-_2$	+ DMA	↔	DMA=B(OH)-O2-DMA^-	+ H_2O		-35.33	-3.32	-12.4
DMA=$B(OH)^-_2$	+ DMA	↔	DMA=B(OH)-O3-DMA^-	+ H_2O		-36.90	7.55	-1.3
Second chelation								
DMA=B-O(2)-DMA	+ H2O	↔	DMA=B(R)=DMA	+ H_3O^+		140.30	143.0	151.1
DMA=B-O(2)-DMA	+ H2O	↔	DMA=B(S)=DMA	+ H_3O^+		138.64	140.0	144.7
DMA=B-O(3)-DMA	+ H2O	↔	DMA=B(R)=DMA	+ H_3O^+		122.06	123.7	133.3
DMA=B-O(3)-DMA	+ H2O	↔	DMA=B(S)=DMA	+ H_3O^+		120.40	120.7	126.9
DMA=B(OH)-O2-DMA-		↔	DMA=B(R)=DMA	+ H_2O		-7.49	-13.0	-51.2
DMA=B(OH)-O2-DMA-		↔	DMA=B(S)=DMA	+ H_2O		-9.14	-15.9	-57.6
DMA=B(OH)-O3-DMA-		↔	DMA=B(R)=DMA	+ H_2O		-5.92	-12.0	-53.2
DMA=B(OH)-O3-DMA-		↔	DMA=B(S)=DMA	+ H_2O		-7.58	-15.0	-59.5

Transition states

As has been mentioned, transition state searching is quite computationally expensive. For this reason, transitions states were found along only one complete pathway from reactants to products. Four transition states are expected between the five stationary point steps. Transition states for borate complexes have not been observed.

Each of the first three transition states have been found in the boric acid binding case. Transition state 1 between DMA and Boric Acid bind at O(3) is shown in Figure 4.27. The second boron bond transition state is between DMA-O(3)-$B(OH)_2$ and O(2), it is found in Figure 4.28. Figure 4.29contains the dimerization transition state between DMA=B-OH and O(3')-DMA

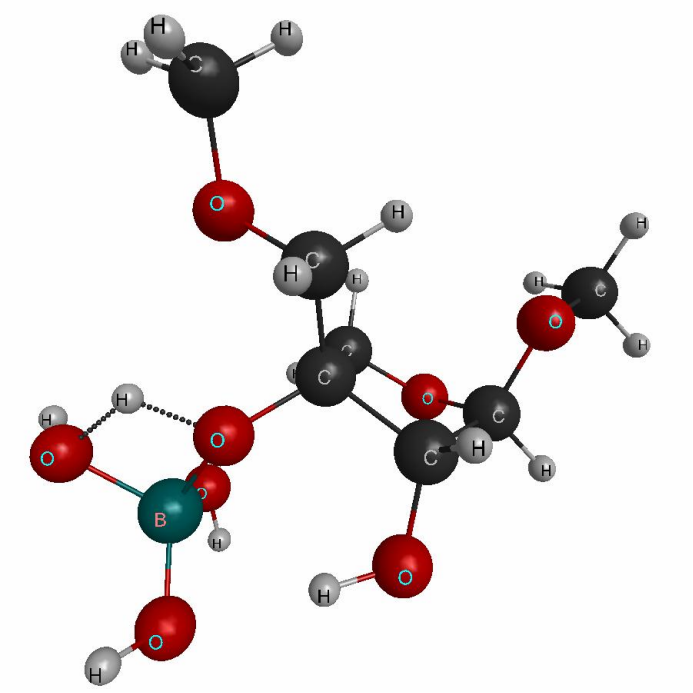


Figure 4.27 $B(OH)_3$-O(3)-DMA First boron bond transition state

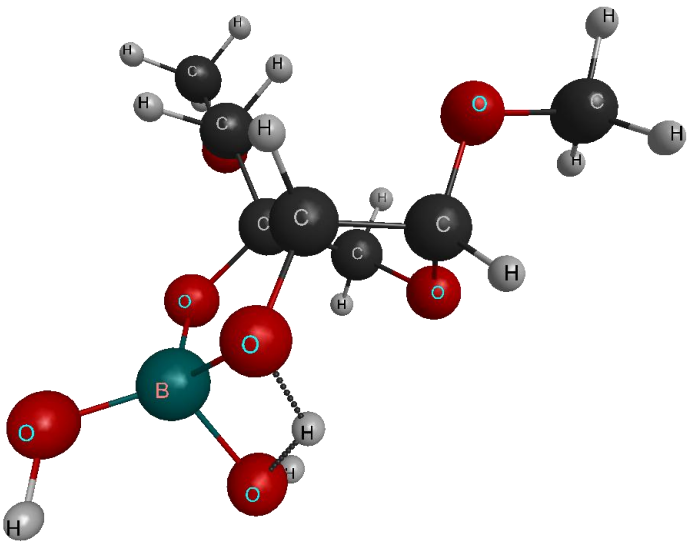


Figure 4.28DMA-B(OH)$_2$-O(2) Second boron bond first chelation transition state

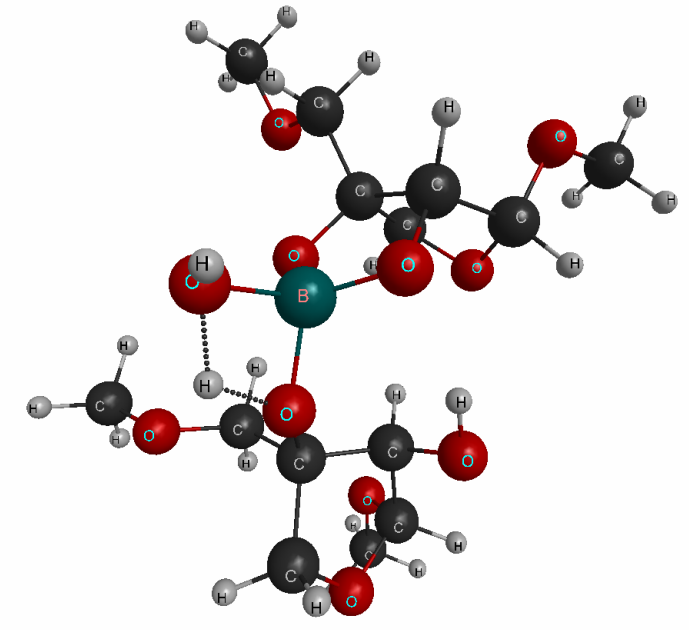


Figure 4.29 DMA-B-O(3'')-DMA Third boron bond dimerization transition state

The final transition state is for the second chelation. For boric acid this is a deprotonation reaction and is assisted by a water molecule. Several intuitive searches were

completed in an attempt to find this transition state, in all cases the saddle point optimization failed to locate a stationary point. This failure to find a stationary point indicated that either there is no transition state, that the saddle was too shallow to find with the defined parameters or that the area of the potential energy surface that we were looking at was the wrong place. CTEPES allows us to scan the PES in order help us identify the problem.

CTEPES was utilized at steps along a closing pathway for DMA=B-O(3')-DMA. Slices of the PES were taken at varying B-O(2'') bond lengths. A diagram of the moving components is pictures in Figure 4.30. The PES for the slices are modeled in Figure 4.31-Figure 4.34. The contour of the PES at each slice retains its basic shape. For all B-O bond lengths tried, there is no barrier, which tends to confirm the initial hypothesis that this reaction is barrier free.

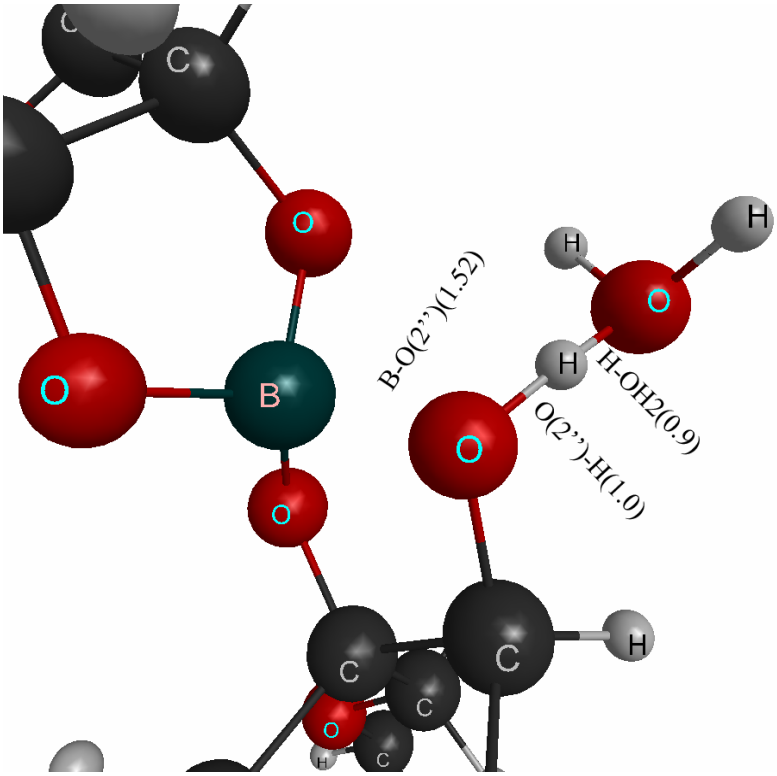


Figure 4.30 Structure of transition state four

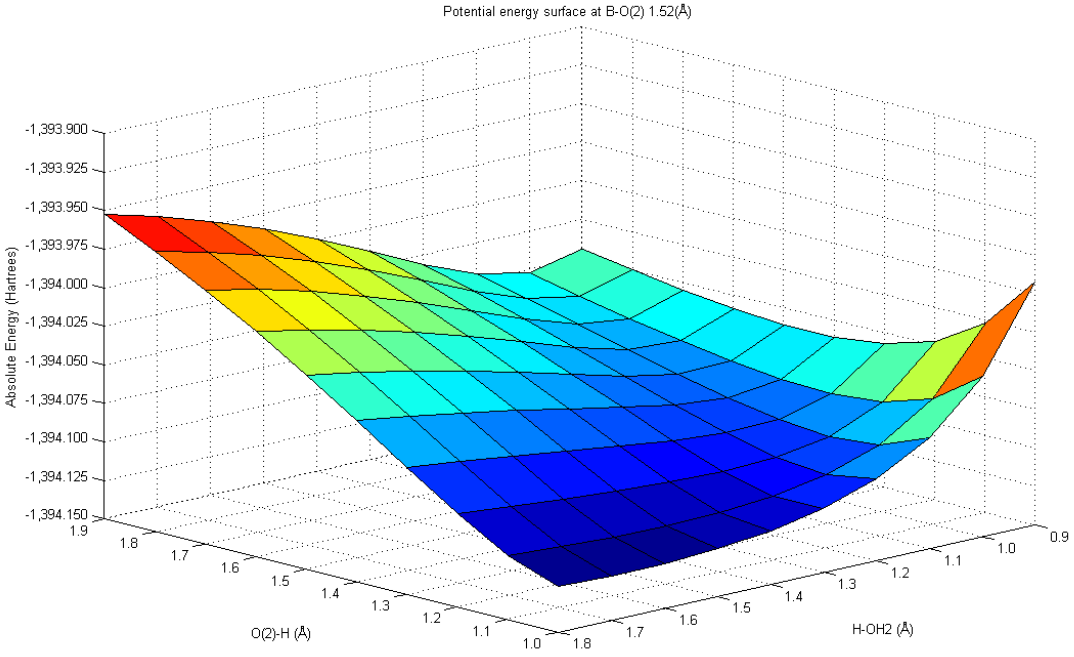


Figure 4.31 PES at B-O(2'') 1.52 Å

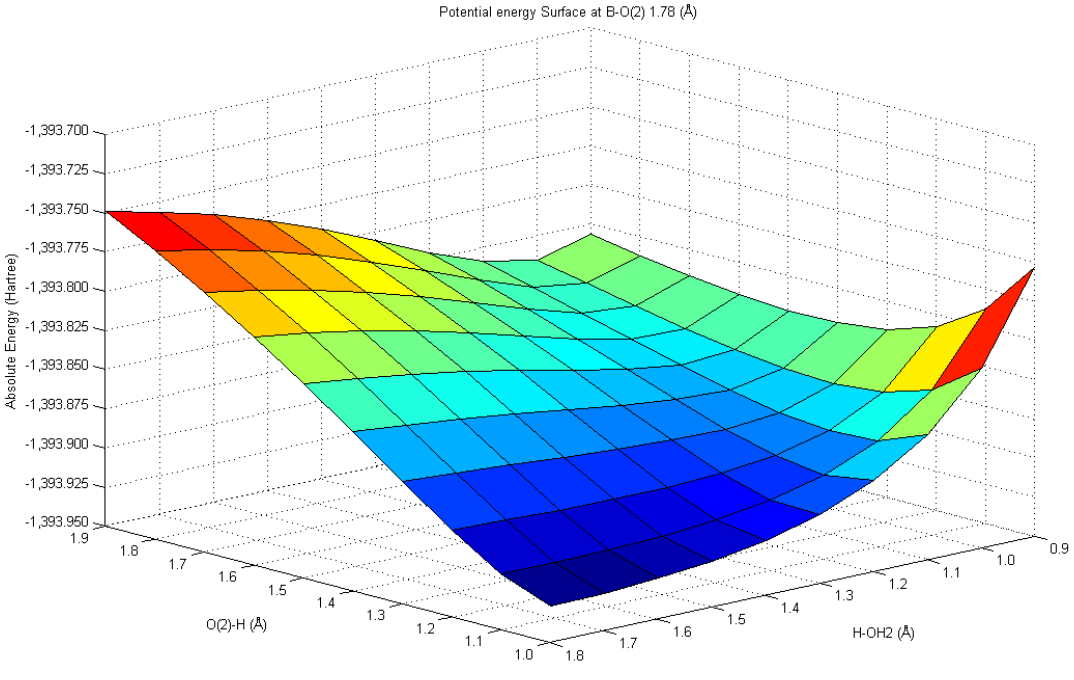


Figure 4.32 PES at B-O(2'') 1.78 Å

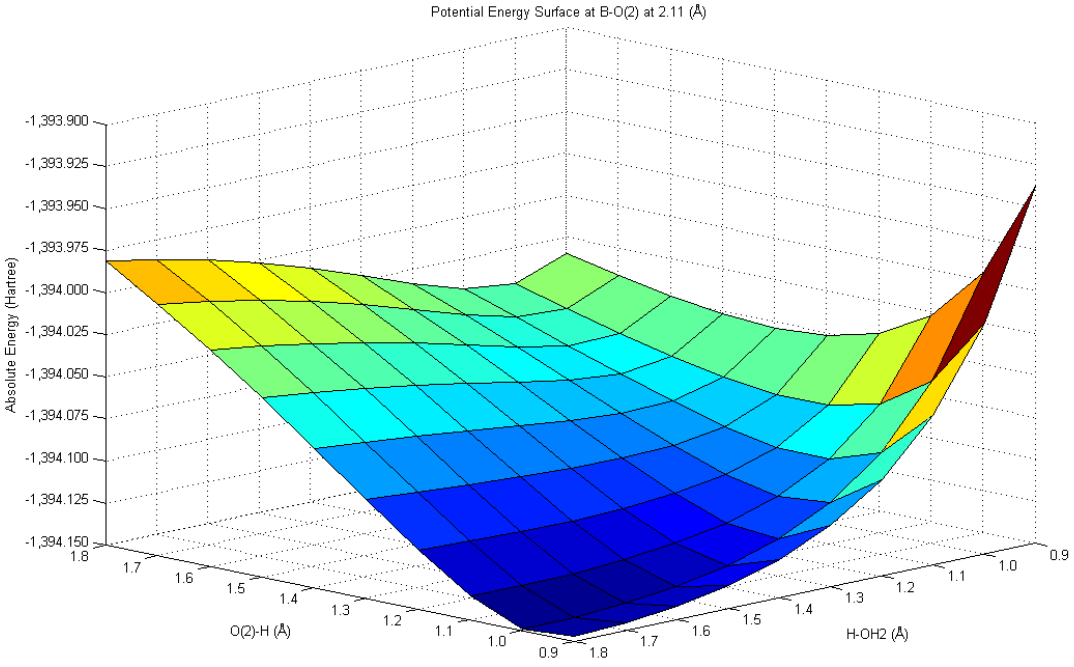


Figure 4.33 PES at B-O(2'') 2.11 Å

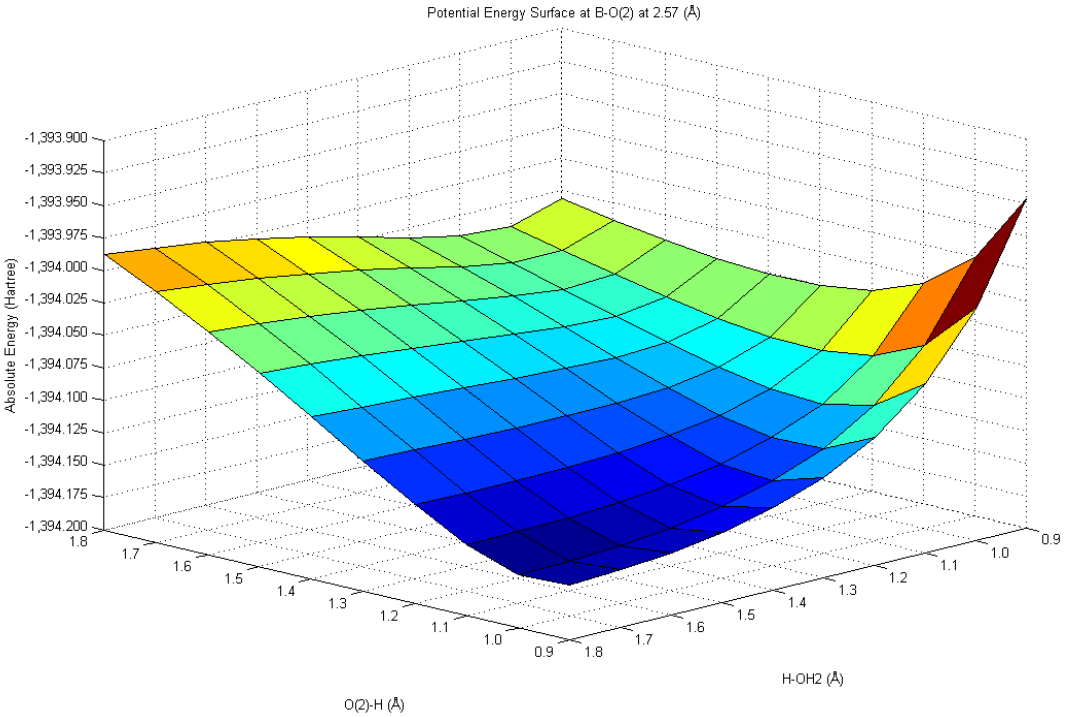


Figure 4.34 PES at B-O(2'') 2.57 Å

Boric Acid to Borate

In order for both reaction path ways to be described we must suggest that both reactions have the same starting and ending materials. This necessarily incorporates the boric acid/borate ion equilibrium. This reaction offsets the borate ion reaction energetics.

Several solvent models were used to attempt to understand the boric acid equilibrium. The methods used were MP2/6-31++G(d) in the gas phase, MP2/6-31++G(d) with a PCM solvent (ICOMP=0 ICAV=0), MP2/6-31++G(d) with an PCM solvent model (ICOMP=2 ICAV=1). The Surface Volume Polarization for Electrostatics (SVPE) and Simulation of Volume Polarization for Electrostatics (SS(V)PE) were utilized to estimate the amount of escaped charge that was occurring[26–29]. SS(V)PE and SVPE with 1-5 cavities were run these methods better approximate the electrostatic interaction. With increasing accounting for escaped charge the volume polarization method approached a limit that was approximately equal to the improved PCM electrostatic interaction term.

Thermal corrections to the internal energy are poor approximations as translations and rotations can collide with the continuum shell. Vibrational component is a better approximation but caution should be used here as well as the solvent shell is static.

$$B(OH)_3 + 2H_2O \leftrightarrow B(OH)^-_4 + H_3O^+$$

Table 4.7 Boric Acid Reaction energies (kJ·mol^{-1})

	ΔU_e	ZPE	ΔH^{298}	ΔG^{298}
Gas	705.3	715.7	711.0	755.5
PCM	173.96	189.9	183.7	231.1
Improved PCM	154.3	169.1	162.5	210.9

Discussion

R versus S configuration

Using zero point energy (ZPE) corrected energies for the transition from S→R we find an energy gap of 4.02 kJ·mol-^{1} this corresponds to a K_{eq} of 0.198. Equilibrium constants are very susceptible to small changes in free energy. However this does suggest that the structure with a slightly higher relative proportion of the R and S configurations reported by Ishii and Ono should be assigned to the S configuration.

Boric/Borate

From Table 4.7 the calculated gas phase ΔH^{298} for boric acid hydrolyzing water is 711.0 kJ·mol^{-1} this matched well with the calculated from standard heats of formation the approximate gas phase enthalpy of 713.061 kJ·mol^{-1}. It is apparent that the gas phase model appropriately models the physical system.

$$\mathbf{\Delta H^{298} = [(-1344.026)(aq) + (581.158)(g)]-[(-992.277)(g) +2*(-241.826)(g)]}$$

There are however, some results that one would not expect when comparing to the experimental system. The pKa of boric acid is 9.237 which is a ΔG^{298} of 52.725. This is a significant deviation from the PCM model used herein and even from an improved model. One would expect that for a system with so many –OH groups that an explicit solvent model would significantly improve the results of a solvation. However, for our complete system even the first solvent shell would tedious to build and would most likely be prohibitively

expensive. Previous attempts at global optimization of a quantum boric acid in 64 EFP water molecules across the half the cluster with even cursory simulated annealing parameters have taken as long as a week to complete. Creating a solvent shell for the dimer of DMA would involve hundreds of waters molecules and possibly thousands for a second solvent shell.

Charged Species

When one looks at the RG-II system it is important to remember that the reported dimer structure is identified as a charged molecule. Its dimerization is regulated by large divalent cations and it is spontaneous at a neutral pH. The overall reaction energies for the DMA modeled system are provided in Table 4.8

Table 4.8 Overall reaction, absolute energy reaction energies, Free energy, equilibrium constants (kJ·mol^{-1})

						ΔU_e	ΔH^{298}	ΔG^{298}	K_{eq}
2 DMA	+ $B(OH)_3$	↔	DMA=B(R)=DMA$^-$	+ $2H_2O$	+ H_3O^+	106.52	96.1	76.2	4.40E-14
2 DMA	+ $B(OH)_3$	↔	DMA=B(S)=DMA$^-$	+ $2H_2O$	+ H_3O^+	108.17	99.1	82.9	3.04E-15
2 DMA	+ $B(OH)^-_4$	↔	DMA=B(R)=DMA$^-$	+ 4 H_2O		-67.45	-172.1	-127.9	2.60E+22
2 DMA	+ $B(OH)^-_4$	↔	DMA=B(S)=DMA$^-$	+ 4 H_2O		-65.79	-170.4	-126.3	1.33E+22
$B(OH)_3$	+ $2H_2O$	↔	$B(OH)^-_4$	+ H_3O^+		173.96	183.7	231.1	3.20E-41

Using Table 4.6, two DMA make a crosslink that is free energy favored to form three bonds for our system. The fourth bond appears to have similar unexpected results as occurred in the boric acid case. This was a surprising result as all reported results discuss a four-coordinate boron crosslink. We hypothesize that there are reasons why the current model study may not match expected results. They include (1) an inability to fully model solvent effects and (2) the inability to include the divalent cation in our model. Currently, too little is known about how the cations (or even which cations) are incorporated into the system to

allow for computational modeling at this point. It is of note that the cations tend to stabilize better with large radii. It is unknown why the cationic moiety needs to be divalent or why the large radii cations are so much more effective at stabilizing the dimer than small cations.

If we assume that some correction to solvated charged species exists and that it is relatively consistent then in Figure 4.35 and Figure 4.36 all of the green lines(borates) would likely drop by the same amount, including the final product (R and S dimer). The borate system maintains the same charge for the whole reaction it is therefore rational to approximate that the error is relatively consistent across the entire calculations.

The borate ion binding is a spontaneous reaction for the overall reaction. The aforementioned correction would presumably make the final binding a spontaneous binding and if it is a consistent across all the steps and is consistent with the borate free energy changes this would be a strongly favored reaction that could be modulated by pH and described simply with Le Chatelier's principle.

Borate availability

Typically the RG-II system is described as a borate binding. This description, however, raises a few questions. Using the Henderson-Hasselbalch equation, at the pH's relevant to the formation of the dimer (4-5);

$$pH = pK_a + \log([B(OH)_4^-]/[B(OH)_3])$$

the ratio of borate to boric is $1.8x10^{-6}$, i.e. the availability of borate ion is 55000 times less likely (for an already minimal amount of boron). This fact is also demonstrated in a different

way in the DMA model with the energy gap involved in making the borate ion.

Transition state barriers

Significant effort went into finding borate transition states, however none have been found. The borate ion is a crowded system. While in chapter 2 five-coordinate boron is readily observed that is a three-center, two-bond transition state. Borate ion transition states are four-center, four-bond transition state with large nearby steric groups that are capable of hydrogen bonding throughout the system.

Searches for transitions are immediately repulsive and readily formed boric acid as commonly as quickly pushing borate away. Intuitive transition states for borate ion were typically un able to converge. This does not exclude the possibilty of borate transition state however it does suggest that borate's will have higher reaction barrier heights than the boric case. Barriers for boric acid pathway are already high so this suggests that borate ion is the less likely pathway to borate crosslink.

Figure 4.35 diagrams the total reaction path way on the absolute energy surface. Figure 4.36 corrects the absolute energy to describe the free energy. The surfaces are clearly delineated with high energy transition states. The transition states were determined only for the entire O(3) bound boric acid path way. The fourth B-O bond in boric acid was shown to be barrier free so the median structure energy was plotted as the transition state. The transition state searches are computationally expensive. The O(2) boric acid pathway is expected to have similar barrier heights to the O(3) case. Borate transition states were unable to be found.

The transition state barriers determined for the O(3) were therefore, used as a model for the other transition states. The O(2) boric states (light blue) one would expect to have similar barrier heights. The borate transition states (both greens) one would expect to have higher barriers than the boric case (the boric O(3) barrier heights are shown for perspective). The final boric acid transition state was approximated to be equal to the mean of the other boric O(3) dehydration reaction barriers. A relative scale was used for comparison (Figure 4.35 , Figure 4.36). The initial configuration (Far left) is 2 DMA with boric acid. The second point is the same energies for the boric case but add the energy gap for the conversion of boric acid to borate. The astute observer will note that after the first chelation the products are the same again and any combination of pathways is possible. The pathways are diagramed for ease of visibility. Finally comparing Figure 4.35 and Figure 4.36 shows some subtle differences in spontaneity.

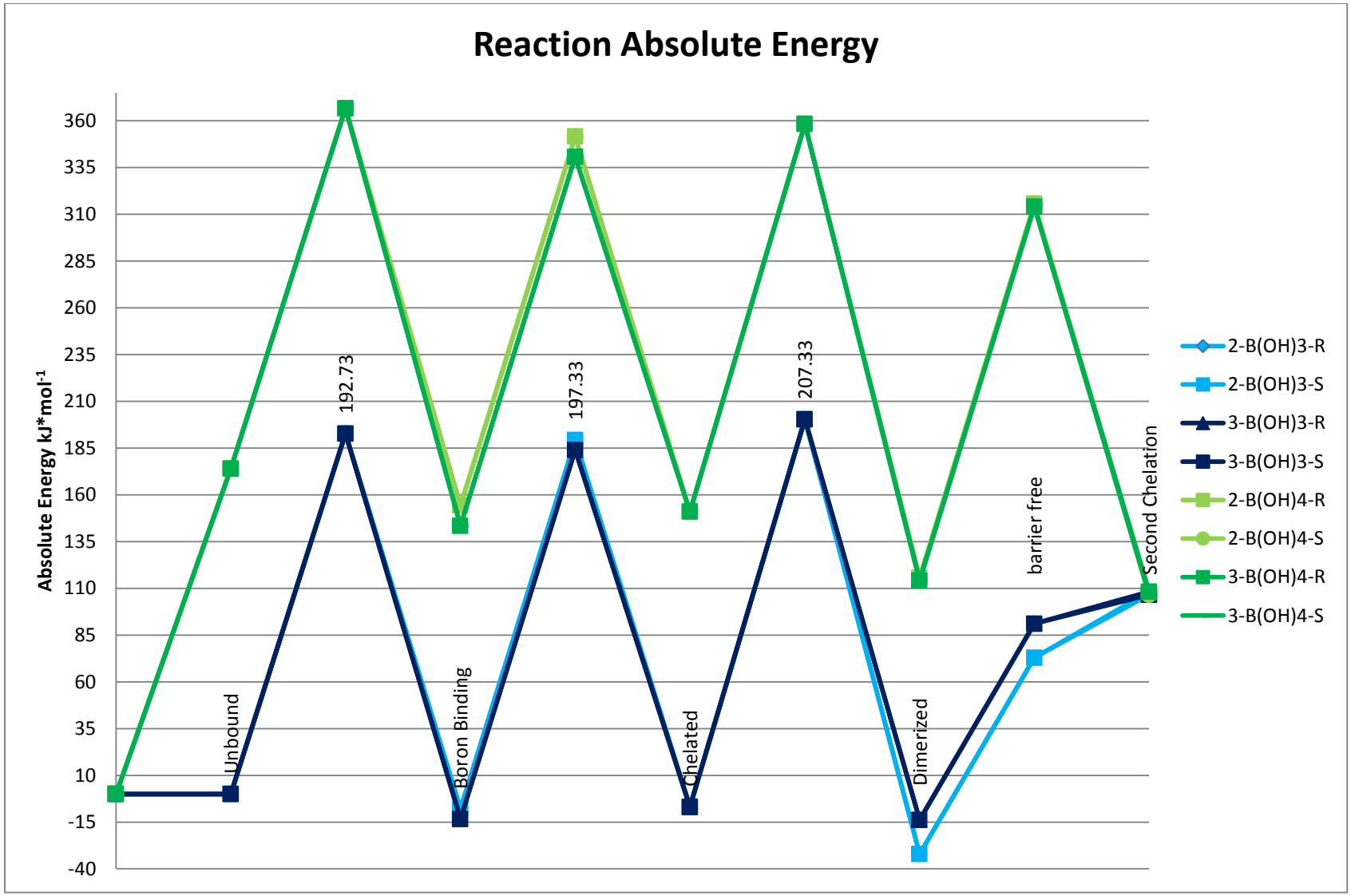


Figure 4.35 Relative Absolute energy dimerization pathway

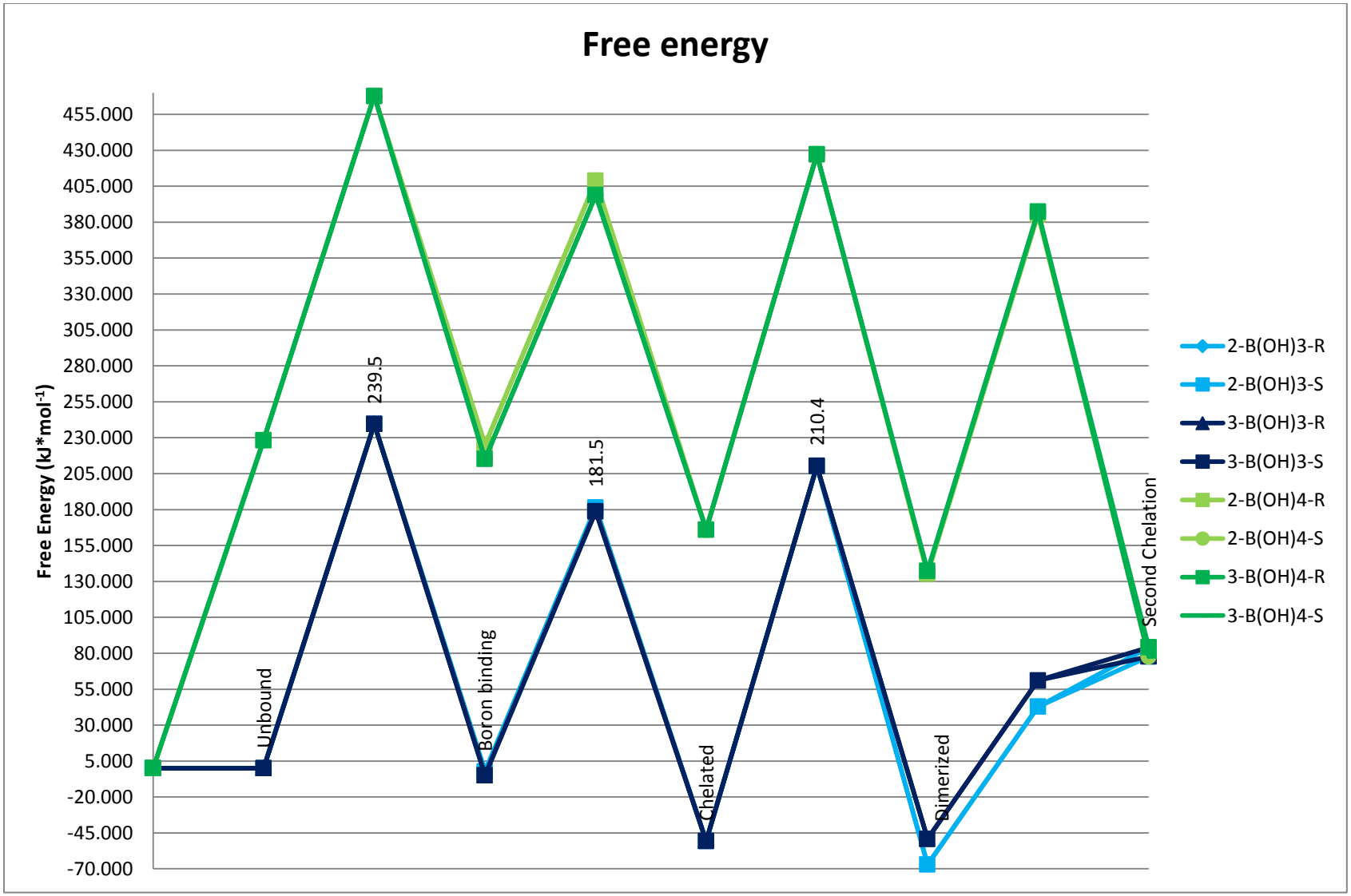


Figure 4.36 Relative free energy along dimerization pathway

Conclusion

Rhamnogalacturonan-II forms a dimer dRG-II-B at pH's above 5 with the assistance of large radii divalent cations. This dimer has been reported to be bridged with a borate crosslink at the 2, 3 position of the A side chain apiosyl residue. The apiosyl crosslink has been modeled with dimethyl-apiose. The RG-II crosslink pathway is expected to be a stepwise reaction. Each step of the pathway is a dehydration reaction. In the modeled system those barriers have been found to have large barriers.

The R and S ratio of the dimer of DMA are nearly equal in free energy. The S enantiomer appears to be the more stable conformer; therefore it appears that the reported methyl-apiose dimer peak with slightly higher proportion can be assigned to S. The borate ion binds spontaneously and tightly as expected. The Le Chatelier's principle readily explains the dissociation mechanism that is pH modulated.

Even at neutral pH borate concentrations are small. In addition a substantial amount of energy must be invested in initial conversion to borate makes it energetically less favorable. It was not possible to locate the borate dehydration transition state barriers; these barriers might be expected to be greater than boric barriers as they are more significantly sterically constrained. The boric barriers are already high, therefore in addition to the other effects we predict that the borate pathway is not a major pathway to product.

The charged species studied here in seem to incompletely represent the energetics of the system despite finding geometries that do not significantly change with variation in solvent methodologies. Further work to better approximate the solvent would be a good avenue for future work.

References

(1) Mengel, K.; Kirkby, E. *Principles of Plant Nutrition*; 5th ed.; Kluwer Acedemic Publishers, 2001.

(2) Foroughi, M.; Marschner, H.; Döting, H. - W *Zeitschrift für Pflanzenernährung und Bodenkunde* **1973**, *136*, 220–228.

(3) O'Neill, M. A. *Methods in plant biochemistry*; Academic Press: London ; San Diego, 1990; Vol. 2.

(4) Ridley, B.; O'Neill, M.; Mohnen, D. *Phytochemistry* **2001**, *57*, 929–967.

(5) O'Neill, M. A.; Ishii, T.; Albersheim, P.; Darvill, A. G. *Annual Review of Plant Biology* **2004**, *55*, 109–139.

(6) Kobayashi, M.; Nakagawa, H.; Asaka, T.; Matoh, T. *Plant Physiol.* **1999**, *119*, 199.

(7) ONeill, M.; Warrenfeltz, D.; Kates, K.; Pellerin, P.; Doco, T.; Darvill, A.; Albersheim, P. *J. Biol. Chem.* **1996**, *271*, 22923–22930.

(8) PUVANESARAJAH, V.; DARVILL, A.; ALBERSHEIM, P. *Carbohydr. Res.* **1991**, *218*, 211–222.

(9) WHITCOMBE, A.; ONEILL, M.; STEFFAN, W.; ALBERSHEIM, P.; DARVILL, A. *Carbohydr. Res.* **1995**, *271*, 15–29.

(10) MELTON, L.; MCNEIL, M.; DARVILL, A.; ALBERSHEIM, P.; DELL, A. *Carbohydr. Res.* **1986**, *146*, 279–305.

(11) SPELLMAN, M.; MCNEIL, M.; DARVILL, A.; ALBERSHEIM, P.; DELL, A. *Carbohydr. Res.* **1983**, *122*, 131–153.

(12) Vidal, S.; Doco, T.; Williams, P.; Pellerin, P.; York, W.; O'Neill, M.; Glushka, J.; Darvill, A.; Albersheim, P. *Carbohydr. Res.* **2000**, *326*, 277–294.

(13) Glushka, J.; Terrell, M.; York, W.; O'Neill, M.; Gucwa, A.; Darvill, A.; Albersheim, P.; Prestegard, J. *Carbohydr. Res.* **2003**, *338*, 341–352.

(14) STEVENSON, T.; DARVILL, A.; ALBERSHEIM, P. *Carbohydr. Res.* **1988**, *182*, 207–226.

(15) Reuhs, B. L.; Glenn, J.; Stephens, S. B.; Kim, J. S.; Christie, D. B.; Glushka, J. G.; Zablackis, E.; Albersheim, P.; Darvill, A. G.; O'Neill, M. A. *Planta* **2004**, *219*, 147–157.

(16) STEVENSON, T.; DARVILL, A.; ALBERSHEIM, P. *Carbohydr. Res.* **1988**, *179*, 269–288.

(17) YORK, W.; DARVILL, A.; MCNEIL, M.; ALBERSHEIM, P. *Carbohydr. Res.* **1985**, *138*, 109–126.

(18) Rodriguez-Carvajal, M.; du Penhoat, C.; Mazeau, K.; Doco, T.; Perez, S. *Carbohydr. Res.* **2003**, *338*, 651–671.

(19) THOMAS, J.; DARVILL, A.; ALBERSHEIM, P. *Carbohydr. Res.* **1989**, *185*, 261–277.

(20) Perez, S.; Rodriguez-Carvajal, M.; Doco, T. *Biochimie* **2003**, *85*, 109–121.

(21) MAZUREK, M.; PERLIN, A. *Can. J. Chem.-Rev. Can. Chim.* **1963**, *41*, 2403–&.

(22) Pellerin, P.; Doco, T.; Vidal, S.; Williams, P.; Brillouet, J.; ONeill, M. *Carbohydr. Res.* **1996**, *290*, 183–197.

(23) Ishii, T.; Matsunaga, T.; Pellerin, P.; O'Neill, M.; Darvill, A.; Albersheim, P. *J. Biol. Chem.* **1999**, *274*, 13098–13104.

(24) Ishii, T.; Ono, H. *Carbohydr. Res.* **1999**, *321*, 257–260.

(25) Kobayashi, M.; Matoh, T.; Azuma, J. *Plant Physiol.* **1996**, *110*, 1017–1020.

(26) Chipman, D. *J. Chem. Phys.* **1997**, *106*, 10194–10206.

(27) Zhan, C.; Bentley, J.; Chipman, D. *J. Chem. Phys.* **1998**, *108*, 177–192.

(28) Chipman, D. *J. Chem. Phys.* **2006**, *124*.

(29) Chipman, D. *J. Chem. Phys.* **2000**, *112*, 5558–5565.

CHAPTER 5. CONCLUSIONS AND FUTURE WORK

Boron participates in biochemical systems apparently because of its ability to act as a Lewis acid. In order to better understand boron's role in these large systems the research motif has been to build understanding with small calculations then to scale to progressively larger systems.

The Lewis acid/base aspect was considered in animal physiology systems with B-N bonds. In chapter two boron's role in dative bonding was examined to support an explanation for physiologically active boranes. This helped us further our understanding of the small boron containing system. Using Lewis acid/base context it was demonstrated that four-coordinate boron molecules have the capacity to transfer to Lewis base sites in the body and thereby perhaps interrupt the natural physiological pathways.

The plant physiological system is an intuitive progression of the concept of scaling small systems to large physiologically relevant systems. In plant physiology, B-O dative bonds are the primary mechanism that boron demonstrates its Lewis acidity. It became necessary, due to the intrinsic difficulties in studying saccharides in plants, to develop a methodology by which to investigate them. In chapter three we discuss the CREPES tool which allows rapid sampling of an entire potential energy surface that helps us inform our intuition of these systems and identify global minima.

Finally, a model for the rhamnogalacturonan-II complex was studied to investigate B-O bonds in physiologically relevant plant systems. In chapter four the nature of B-O dative bonding in saccharides was explored. The gleaned information will further instruct models that will be used to scale to ever increasing calculations and has suggested some assignments to experimental data.

Future work in boron chemistry will become more common as its utility becomes more apparent and its research history grows. The incomplete knowledge of how sugars interact with boron is ideal for study with ab initio methods as this level of calculation helps to more readily identify rationalizations for trends.

Although the model studies to date have provided important insights into bonding, there are challenges that remain. Indeed, these challenges are also present in model experimental studies about boron cross links. Three components to the currently accepted descriptions of boron crosslinking have been reported.

First, there are 4 apiosyl residues in dRG-II-B one at side chain A and one at side chain B. In glycosyl linkage analysis (GLA)[1,2] it is possible to identify the in linkages in saccharides. In a GLA of mRG-II one would expect all Apiose to have 1,3' linkage and none with 1,2,3,3'. Indeed O'Neill et al found this ratio to be 8:0 (mol %)[3] in the dimer case you would expect both side chain A Api to show a 1,2,3,3' and both side chain B to show a 1,3' linkage or a 1:1 ratio, however the experiments show a 6:3 (mol %) ratio.

Second, GLA identifies linkages by a process of methylation of hydroxyls, followed by hydrolysis. The RG-II is methylated using sodium methyl sulfinyl carbanion and methyl iodide. Sodium methyl sulfinyl carbanion is the cation of DMSO and has a pKa of 35.1. This suggests that it may be able to attack a 3-coordinate boron and methylate at that site (Figure 5.1). Hypothetically, this reaction analysis would make an Apiose that was bound singly to a 3-coordinate boron appear after GLA to be a 1,3' linked compound.

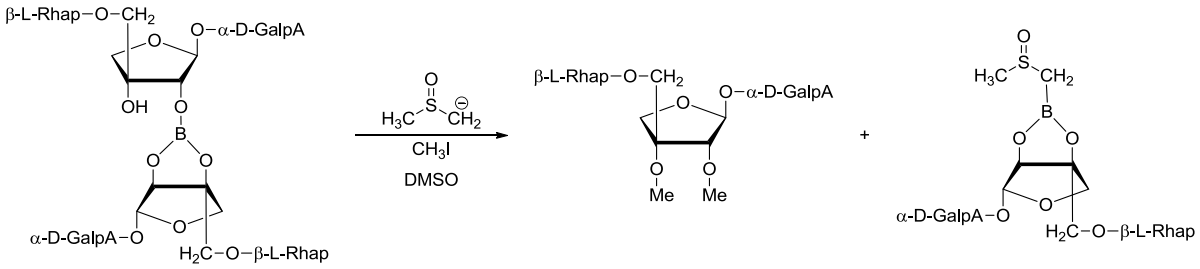


Figure 5.1 Proposed GLA methylation of a putative 3-coordinate crosslink

Third, in 1984 Van Duin et al. presented ^{11}B NMR data of borate esters[4]. This appears to be the primary work cited when any boron-diol NMR work is presented (134 citations as of March 2012 according to web of knowledge). These authors presented a figure much like Figure 5.2 with the exception that this figure adds $\mathbf{B^0L_2}$ which the authors did not consider. Despite the plethora of work done since that time, no investigation of $\mathbf{B^0L_2}$ has been reported. We hypothesize that the role of $\mathbf{B^0L_2}$ may have been missed. No experimental data to suggest what an NMR shift for $\mathbf{B^0L_2}$ would be has been reported. Shifts of $\mathbf{B^0L_2}$ may be occluded or ignored due to overlap with other ^{11}B signals. Given the results of the computational work reported here, this omission in the study of cross-linking may be unwarranted. Specifically, the stereochemistry of the crosslink reaction may cause the reaction labeled Figure 4.26 to not occur, and a relatively stable $\mathbf{B^0L_2}$ structure might be possible.

These three components in addition to the computed results presented in chapter 4 suggest that further investigation into borate crosslinking is needed to more exhaustively explore whether experiments reported to date have fully described the system or whether ab initio methods on modeled for this chemistry are improperly describing the system.

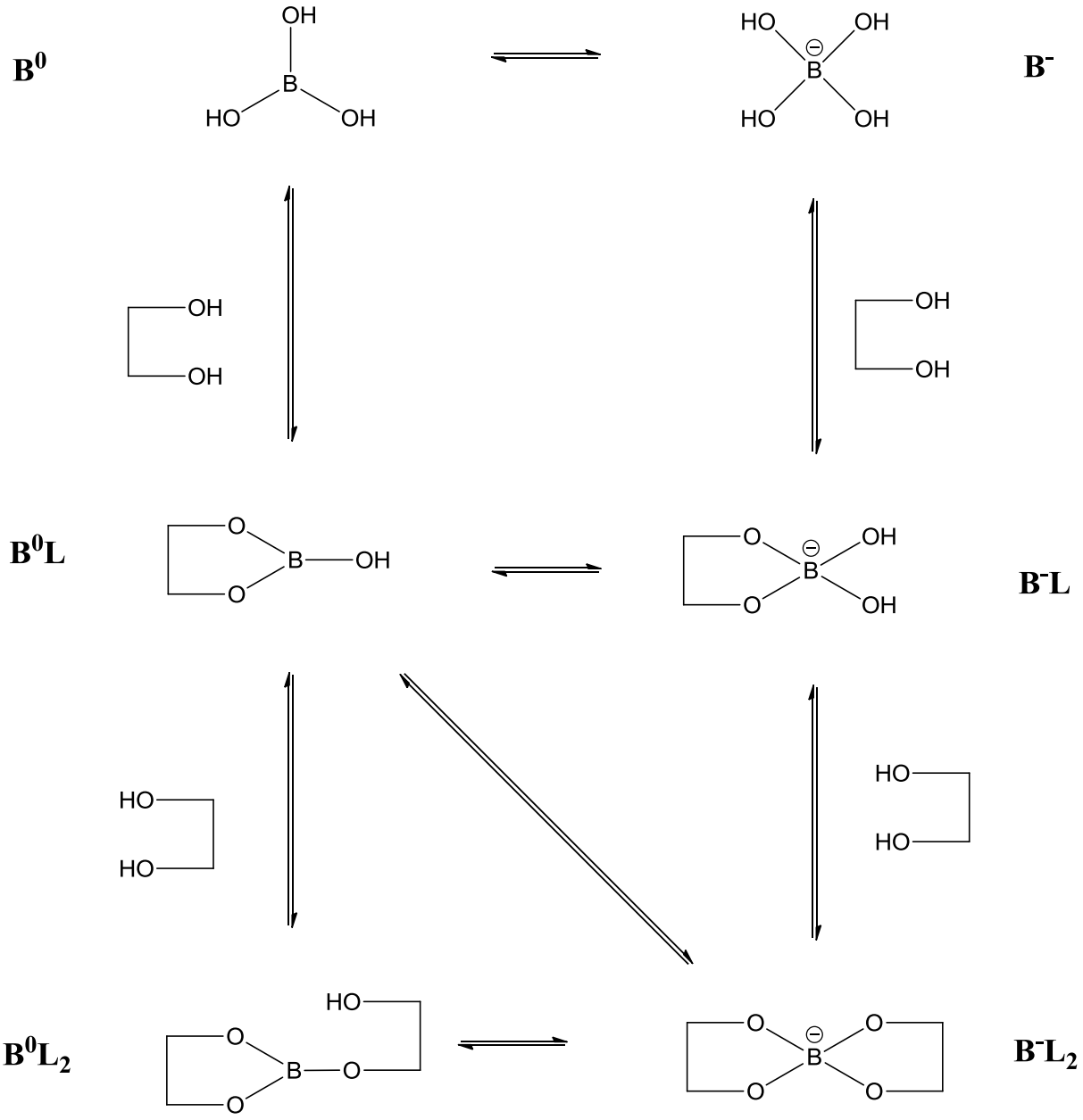


Figure 5.2 Boron diol binding scheme

Future work on the di-methyl apiose (DMA) system to help identify which model description of RG-II provides more helpful insight could also focus on the reported cation dependence[2,3]. RG-II dimerization appears to have a significant dependence on large cations

that are divalent. This dependence inspires several questions for a computational chemist to be able to attempt to answer. Does accounting for the cation significantly change the thermodynamic properties of the reaction? How does the size of the cation make a difference? Why are divalent cations particularly effective at stabilizing the crosslink? How does the role of the cation compete or corroborate the role of the solvent? What type of solvent model is required?

When one considers the larger motif of scaling small calculations to large, many more questions need to be considered. How can more of the RG-II be modeled? How can the pH dependence be better understood? Is it possible to have QM/MM models that make sense for this system? Will MM models be able to include atomistic parameters or will saccharide units need to be considered as a whole for significant fraction of the system[5,6]? How is solvent best modeled in plants? The investigation of the RG-II complex and boron's role in it clearly has significant questions left to answer.

The role of boron dative bonding in biochemical systems is a rapidly expanding field of study that promises a wealth of practical applications when the method of its interaction is well understood. Computationally much of the work need is in large scale systems using methods that currently do not model boron well. It is therefore important to continue to scale our understanding of small boron containing systems to large.

References

(1) HAKOMORI, S. *J. Biochem. (Tokyo)* **1964**, *55*, 205–&.

(2) Pellerin, P.; Doco, T.; Vidal, S.; Williams, P.; Brillouet, J.; ONeill, M. *Carbohydr. Res.* **1996**, *290*, 183–197.

(3) ONeill, M.; Warrenfeltz, D.; Kates, K.; Pellerin, P.; Doco, T.; Darvill, A.; Albersheim, P. *J. Biol. Chem.* **1996**, *271*, 22923–22930.

(4) VANDUIN, M.; PETERS, J.; KIEBOOM, A.; VANBEKKUM, H. *Tetrahedron* **1984**, *40*, 2901–2911.

(5) Noto, R.; Martorana, V.; Bulone, D.; San Biagio, P. L. *Biomacromolecules* **2005**, *6*, 2555–2562.

(6) da Silva, C. O. *Theoretical Chemistry Accounts* **2006**, *116*, 137–147.

CHAPTER 6. Acknowledgements

First and foremost I would like to thank my wife Pam. Her indomitable patience through my long graduate career has been a boon to me and I will eternally be grateful for her love and support.

I would like to express my thanks to my parents whose subtle support of my enthusiasm for science has lead me to the place that I am, and whose persistent praise gave me the audacity to believe that I could accomplish anything.

I would like to thank Dr. Holme for his gentle guidance over the years has taught me more things than I will ever know. To Jeff Raker I offer my eternal gratitude. His willingness to help in the completion of this document was astounding and more generous of a gift than anyone is worthy of.

Finally I would like to thank my friends throughout graduate schools whose persistent ridicule drove me to complete the things I have started and whose distractions have sustained me for these many years. In particular I would like to thank Nate Frank for his constant encouragement and challenging which more than anything helped me accomplish the goals I set out for myself.

And to all who have helped me through the years,

Thanks!