ABSTRACT

MOKRAUER, DAVID SAMUEL. Interpolatory Surrogate Models for Light-Induced Transition Dynamics. (Under the direction of Dr. C. T. Kelley.)

The potential energy of a molecule is a function of its geometry. Molecules observed naturally will be in conformations that minimize this potential energy and some molecules have multiple minima. In this dissertation, we develop algorithms that simulate transition paths between local energy minima through higher energy states.

Computing the potential energy is an expensive optimization process that may fail due to poor initial iterates. An N atom molecule has 3N-6 degrees of freedom so the problem must be simplified to a few choice coordinates. Prior simulations rarely used more than a single degree of freedom. While this may be successful for a small molecule, simulations for larger molecules are not feasible with such simplification.

We addressed some of the difficulties in computing the potential energy for specified geometries by developing continuation schemes. These continuation schemes incorporate pre-processing of some non-design variables for improved results. Our continuation methods run in parallel and exhibit good scalability.

Simulation of the transitions is done by following the gradient flow. Even though our continuation methods make it possible to compute the potential energy at nearly any point in the design space, increasing the degrees of freedom in the simulation is still a burden because the energy computations are so expensive. We decreased the number of function evaluations by incrementally constructing cubic spline surrogates of the energy functional in small patches surrounding the gradient flow path. The tensor product grids required for the spline surrogate exhibit exponential complexity in the number of degrees of freedom. We are able to obtain polynomial complexity by replacing splines with Smolyak sparse interpolation. Finally we incorporate multiple error approximation and control schemes to maintain accuracy and decrease the simulation time.

Interpolatory Surrogate Models for Light-Induced Transition Dynamics

by
David Samuel Mokrauer

A dissertation submitted to the Graduate Faculty of
North Carolina State University
in partial fulfillment of the
requirements for the Degree of
Doctor of Philosophy

Applied Mathematics

Raleigh, North Carolina

2012

APPROVED BY:

Dr. Jerry Whitten	Dr. Ralph Smith
Dr. Dwight Woolard	Dr. C. T. Kelley Chair of Advisory Committee

DEDICATION

This work is dedicated to my beautiful wife, Danielle. Every day she inspires me to be the best person that I can be. It is my deepest wish that each of my children grow up to find as amazing a partner as I have found for myself.

The most memorable moments of my years of graduate school were the birth of my 2 children, Karen and Gavin. Nary a moment goes by when they are not on my mind. All of the work of my life will always be in their honor.

It is tough for me to think about mathematics without thinking of my parents. My mother began teaching me about right angles before I was in kindergarten and my father was showing me calculus in the sixth grade. I am profoundly grateful for the work ethic and values they instilled in me.

BIOGRAPHY

David Mokrauer was born on February 13, 1979 in Summit, NJ. Both of his parents have bachelors degrees in mathematics so perhaps a career in mathematics was simply destiny. He graduated from Westfield High School in Westfield, NJ in 1997. The most important event in David's life occurred in May 2001 when he met Danielle Sciarrone who would become his wife in 2007. In 2002 he graduated from the College of New Jersey with honors in Mathematics and went on to teach high school mathematics in Cinnaminson, NJ for 4 years. In 2007 David entered North Carolina State University to pursue a doctorate in applied mathematics. In July 2010 Danielle and David were blessed with their daughter, Karen, and in October 2011 they were blessed once again by the birth of their son, Gavin.

ACKNOWLEDGEMENTS

I would like to thank my wife, Danielle, for the great sacrifices she has made so that I may accomplish this goal. She has been patient, understanding, and supportive throughout the entire process. I am forever in her debt.

Few phd students are lucky enough to work with an advisor as fantastic as Tim Kelley. His enthusiasm is infectious and I am thankful for his dedication to my success. Having an advisor who values your time is worth its weight in gold. Often graduate school is a story of strife and although I worked extremely hard, I had a lot of fun thanks to Tim's sense of humor and quick wit. Meetings, conferences, classes, and any other events always included some laughter.

I am grateful for the mentoring of Dwight Woolard. This thesis exists thanks to his vision. I would also like to thank him for the concrete chicken adorning my desk.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

The potential energy of a molecule depends on the interatomic forces between all of the electrons and nuclei within the molecule. Nature dictates that the molecule maintain a geometry that is a minimum of the potential energy. Some molecules have multiple potential energy minima. By exciting a molecule photonically it is possible that the molecule will transition from one of these minima to another. Simulating this reaction has historically been limited to a one [47] or two dimensional model[2, 44]. If the reaction can be simulated using just a single degree of freedom, then we call that coordinate the intrinsic reaction coordinate (IRC). As molecules get more complex one should expect that there is no longer a single IRC which will capture the effects of excitation. The algorithms that follow increase the potential degrees of freedom for this simulation to double digits and by doing so allow for more accurate simulations of much larger molecules.

The standard model used to compute the potential energy has nuclei in fixed locations surrounded by moving electrons which are occupying orbitals. The geometry of a molecule with N atoms is uniquely determined by $3N-6$ nuclear coordinates. Based on specified nuclear coordinates, software determines the electrons' orbitals and thus the potential energy of the molecule. Potential energy is a quantized value meaning that for a given value of the nuclear coordinates, there are only a subset of the real numbers which can be possible values of the molecule's potential energy. When an electron within the molecule is excited it will change states to occupy a higher orbital. This change in state also signifies an increase in potential energy of the molecule. As a consequence of the excitation the new orbital causes a change in the interatomic forces within the molecule. While the potential energy had been a minimum before the excitation there is no guarantee that the current geometry is a minimum in the new state thus the $3N-6$ nuclear coordinates will relax to a local minimum energy in that new state.

Modeling the energy as a function of $3N-6$ coordinates is too complex for current technology. Fortunately many of those coordinates will remain relatively stationary during the relaxation so we may simplify the model to a function of only the significant nuclear coordinates. This means we partition the coordinates into design variables, x, and dependent variables, ξ. We use software to determine optimal values of ξ for given values of x. This optimization requires a good initial guess for the values of those dependent variables, but we do not have one for most choices of x. The only value of ξ that we know is the value at the initial geometry which is a minimum energy. This led us to use continuation in order to successfully calculate the potential energy for any chosen values of x.

Having decreased the number of variables in the model and devised a method for evaluating the potential energy of the molecule at all values of the design variables, we could now simulate the relaxation. Our first simulation method computed the potential energy of the molecule at all points on a two-dimensional square grid. The surface is then interpolated using a cubic spline to obtain a surrogate for the energy function. The relaxation of the coordinates follows the gradient flow of the surrogate. A finite difference of the surrogate serves as the gradient and we use continuous steepest descent to determine the path. This method of simulation successfully finds a path between minima for our test molecule 2-butene. Unfortunately much of the surface that we compute goes unseen by the optimization and thus is wasted effort.

In order not to waste valuable computer time computing points which we won't use, we then built the surface incrementally. Starting with a local minimum we compute energy at points on a much smaller square grid surrounding the point. On this smaller surface we optimize to a local minimum or a boundary. If we come to a boundary, then we compute energy on another small grid surrounding the boundary point. This method gives us identical results with far fewer computations. This method becomes far too computationally intensive when we increase the degrees of freedom. The number of gridpoints we need to evaluate with square interpolation grids will grow exponentially in the number of degrees of freedom which will render it impossible to run high degree of freedom simulations.

Smolyak's multidimensional interpolation algorithm uses sparse grids instead of square ones. The grids are generated as a linear combination of tensor products of one dimensional interpolation nodes. The set of nodes that are commonly used are nested in the one dimension and the Smolyak algorithm inherits the nesting in higher dimension. Smolyak's algorithm allows for polynomial exactness to any degree. By replacing square grids with Smolyak grids our patches can be produced for much larger degrees of freedom since the number of Smolyak gridpoints grow polynomially in the degrees of freedom instead of exponentially for square grids. Once we had a technique for constructing patches we proceeded to develop error controls which would allow us to grow the size of the individual patches to be as large as possible.

Our first method of error estimation was borrowed from trust-region algorithms [23]. At the terminal point of the gradient flow on the surrogate we calculated the actual energy. We calculated both the predicted decrease in the energy from the initial point on the patch to the terminal point and the actual decrease in energy from the initial point to the terminal point. If the ratio of the actual decrease to predicted decrease was near 1, then we could grow the patch. Similarly if the ratio of the actual decrease to predicted decrease was not close to 1, then we would shrink the patch. Otherwise the patch remained the same size.

We were able to run successful simulations using a trust-region approach to error control, but we identified some drawbacks. This method requires an actual energy calculation at the terminal point of each patch. This serial calculation takes a long time and ends up doubling the simulation time. The trust-region approach also does not recommend how much to grow or shrink the patch. We would like them to be as large as possible while remaining accurate and never having an energy calculation fail.

Smolyak's algorithm using the Chebyshev extrema as the one-dimensional interpolation nodes provides nested sets of multi-dimensional interpolation nodes. This means that a $k + 1$ degree exact grid contains all the points for the k degree exact grid. If we subtract the lower order surrogate from the higher order surrogate, we get an estimate for the actual error as a function of the grid length h. Using this estimate we can calculate how large h can be while remaining accurate to our chosen tolerance. Since we would also like to prevent convergence failures of every energy calculation we only use the formula to calculate a new h if the number of iterations for every energy calculation stays low. If the energy calculations take too many iterations we shrink h by a factor of 2 regardless of the accuracy.

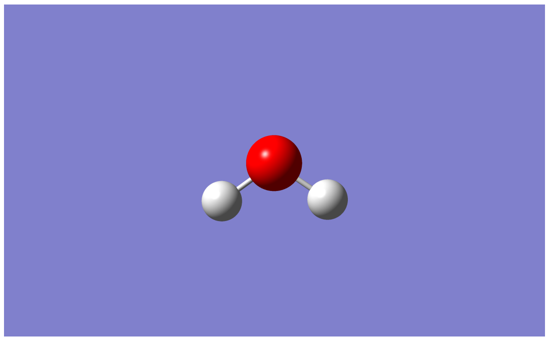
This final version of the software is titled LITES for Light-Induced Transition Effects Simulator and the code for the entire simulation appears in the Appendix C.

Chapter 2

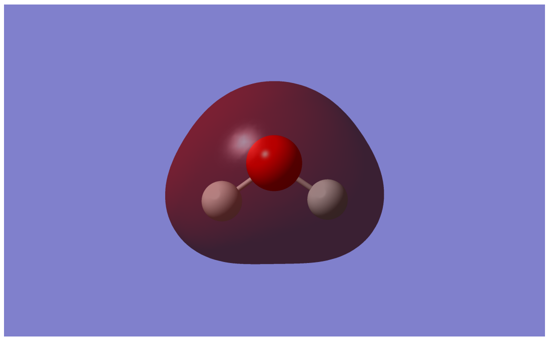
Background Chemistry

2.1 Describing a Molecule

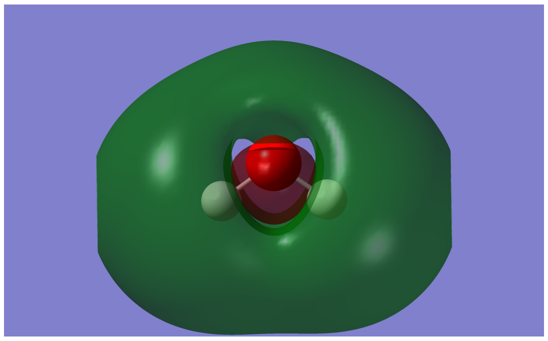
Molecules form when atoms bond and share their electrons. The molecule consists of the nuclei of the atoms which are in a fixed position surrounded by electrons moving throughout fixed orbitals. The standard picture of a molecule displays only the nuclei and the bonds between them. Figure 2.1 shows a water molecule both with and without the orbitals. Each of the orbitals contains 2 of the molecule's electrons.



(a) Water molecule showing only nuclei and bonds



(b) Water molecule including the lowest occupied energy orbital



(c) Water molecule including the highest occupied energy orbital

Figure 2.1: water molecule with and without orbitals shown

The fixed locations of the nuclei describe the geometry of the molecule. In this work, we define those locations by internal coordinates instead of Cartesian [3, 20, 14, 39]. These consist of 3 types of coordinates:

1. Bond length, the distance between bonded nuclei.
2. Bond angle, the angle formed by 3 nuclei connected by 2 bonds.
3. Torsion or dihedral angle, formed by 4 nuclei connected by 3 bonds. Its value is the measure of the angle formed by the projection of the 2 outer bonds onto the plane perpendicular to the center bond.

Examples of all of these coordinates are shown in Figure 2.2. A molecule with N atoms is uniquely determined by $3N-6$ internal coordinates [22]. There are $N-1$ bond lengths,

$N-2$ bond angles, and $N-3$ torsion angles. The coordinates are collected into an array called a Z-matrix.

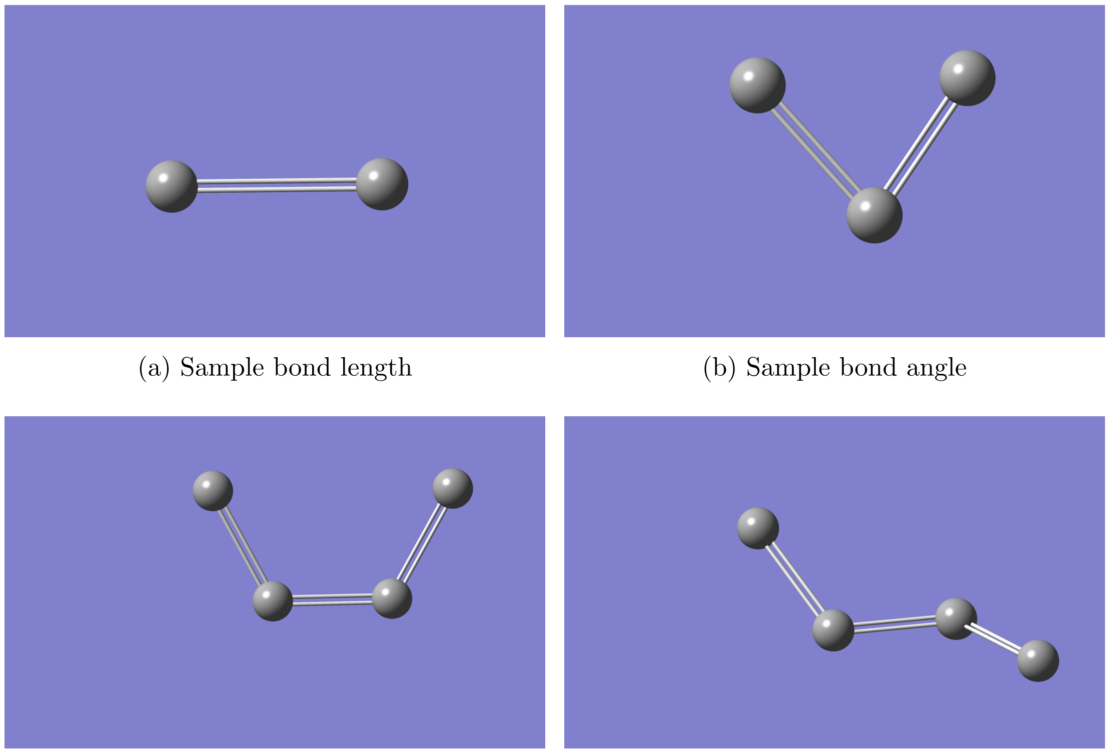
2.1.1 Example

We will use the molecule ethylene, C_2H_4. Figure 2.3 was generated by the Z-matrix in Table 2.1

Table 2.1: Sample Z-matrix for ethylene

Type of atom	Atom 2	Bond length	Atom 3	Bond Angle	Atom 4	Torsion Angle
C						
C	1	1.3				
H	1	1.1	2	120		
H	2	1.1	1	120	3	180.0
H	1	1.1	2	120	4	0.0
H	2	1.1	1	120	5	100.0

The first column gives the types of atoms. The second column indicates the number of the atom bonded to the atom in the first column. For instance, we see that the carbon in the second row is bonded to the carbon in the first row. The third column shows the length of that bond, in this case the carbon 1 - carbon 2 bond is 1.3 angstroms. The fourth column gives the third atom that forms the angular bond and the fifth column gives the measure of that bond. In the fourth row we see that hydrogen 4 is bonded to carbon 2, which is bonded to carbon 1, and the angle formed by those 2 bonds is 120°. Finally the last 2 columns give the fourth nucleus which forms the torsion angle and the corresonding measure. The last row shows hydrogen 6 bonded to carbon 2 bonded to carbon 1 bonded to hydrogen 5, and the dihedral angle they form measures 100°.



(a) Sample bond length

(b) Sample bond angle

(c) 0° torsion angle between 4 atoms

(d) Torsion angle rotated to 140°

Figure 2.2: Sample internal coordinates

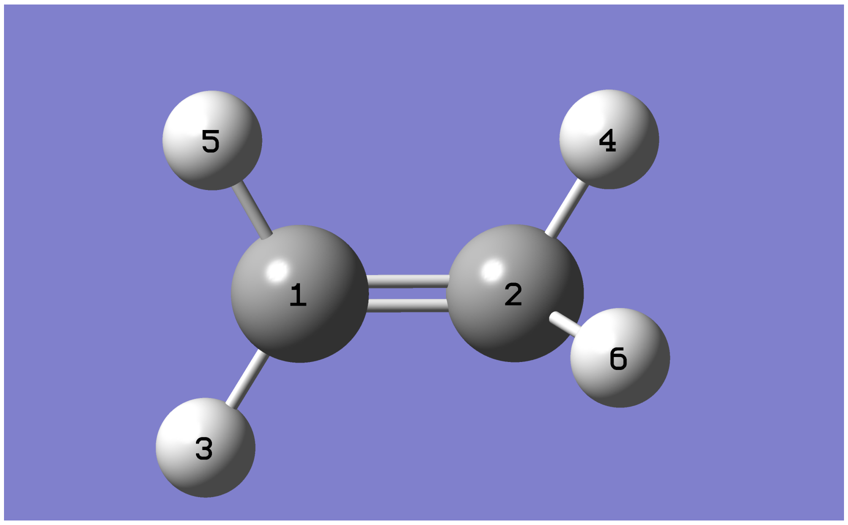


Figure 2.3: ethylene molecule C_2H_4

2.2 Quantized Operators

The Z-matrix contains information about the nuclei of the atoms; it does not give any information about the orbitals occupied by the electrons. The $3N-6$ coordinates are sufficient for uniquely determining the orbitals [22]. These orbitals are solutions to the time independent Schrödinger equation:

$$\hat{H}\Psi = E\Psi. \tag{2.1}$$

E is the energy of the wave function Ψ describing these orbitals and $\hat{H}$ is the polyatomic Hamiltonian operator for a molecule of k atoms with n electrons. The Hamiltonian consists of the sum of the kinetic and potential energy [17, 21]. The kinetic energy of a particle with wavefunction, ψ, is

$$T\psi = -\frac{h^2}{8\pi^2 m}\nabla^2\psi = -\frac{\hbar^2}{2m}\nabla^2\psi \qquad \hbar = \frac{h}{2\pi} \tag{2.2}$$

where h is Planck's constant, $6.6 \times 10^{-34} J \cdot s$, and m is the mass of the particle. The potential energy for our systems is the attraction/repulsion between charged particles.

This term is given by

$$\frac{Q_1 Q_2}{4\pi\epsilon_0 r_{12}} = \frac{Q_1' Q_2'}{r_1 2} \qquad\qquad Q_i' = \frac{Q_i}{\sqrt{4\pi\epsilon_0}} \tag{2.3}$$

where Q_i is the charge of the particle, $\epsilon_0 = 8.854 \times 10^{-12} C^2 N^{-1} m^{-2}$ is called the electric constant, and r_{ij} is the distance between particles i and j. For an electron $Q' = 1$ and for a nucleus of p protons $Q' = p$. For our system of k nuclei and n electrons there will be 2 potential energy terms for

$$V = -\sum_{i=1}^{n}\sum_{j=1}^{k}\frac{Z_j}{r_{ij}} + \sum_{i=1}^{n-1}\sum_{j=i+1}^{n}\frac{1}{r_{ij}}. \tag{2.4}$$

where Z_j is the number of protons in the j^{th} nucleus. Combining the kinetic and potential energy we have the entire Hamiltonian to be

$$\hat{H} = T + V = -\frac{\hbar}{2m}\sum_{i=1}^{n}\nabla_i^2 - \sum_{i=1}^{n}\sum_{j=1}^{k}\frac{Z_j}{r_{ij}} + \sum_{i=1}^{n-1}\sum_{j=i+1}^{n}\frac{1}{r_{ij}} \tag{2.5}$$

When an operator applied to a function results only in values that belong to some discrete subset of the real numbers, we say the operator is quantized. Wavefunctions that are solutions to 2.1 are eigenfunctions of 2.5 and we will find that the associated eigenvalues will only take certain discrete values.

2.2.1 Example

Consider a particle in a one-dimensional box where the potential energy, $V(x)$, is zero inside the box and infinite outside of the box as in Figure 2.4.

$$V(x) = \begin{cases} 0 & \text{if } 0 \le x \le a \\ \infty & \text{else} \end{cases} \tag{2.6}$$

The Hamiltonian for this system is

$$\hat{H} = -\frac{\hbar^2}{2m}\frac{d^2}{dx^2} + V(x). \tag{2.7}$$

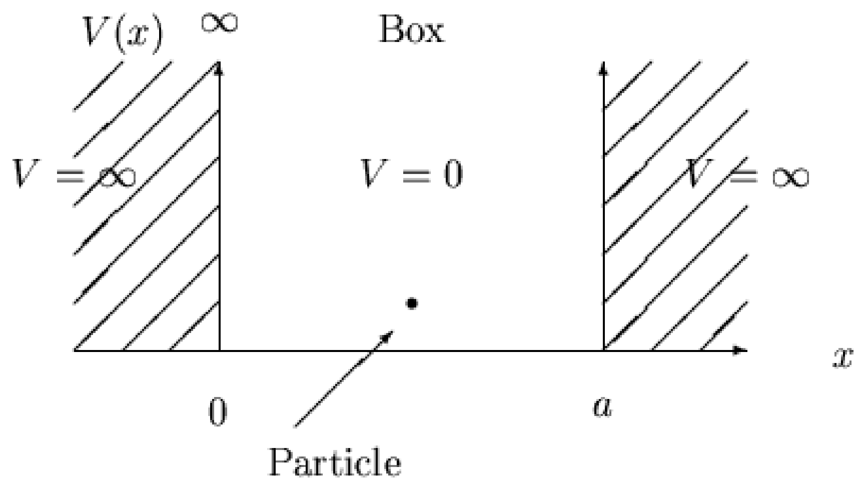


Figure 2.4: Potential energy function for a particle in a one-dimensional box.

Our Schrödinger equation will be

$$-\frac{\hbar^2}{2m}\frac{d^2}{dx^2}\psi(x) + V(x)\psi(x) = E\psi(x). \tag{2.8}$$

In the locations outside the box where the potential is infinite the wavefunction will vanish. Thus, we may conclude $\psi(x) = 0$ outside the box so we only need solve 2.8 inside the box where $V(x) = 0$.

$$-\frac{\hbar^2}{2m}\frac{d^2}{dx^2}\psi(x) = E\psi(x) \tag{2.9}$$

Solutions to this type of differential equation may be found in [40]. Those solutions will be

$$\psi(x) = c_1 \cos\left(\frac{\sqrt{2mE}}{\hbar}x\right) + c_2 \sin\left(\frac{\sqrt{2mE}}{\hbar}x\right) \tag{2.10}$$

Now we apply the boundary conditions. Since we know that $\psi(0) = 0$, we have $c_1 = 0$ so

$$\psi(x) = c_2 \sin\left(\frac{\sqrt{2mE}}{\hbar}x\right). \tag{2.11}$$

We also know that $\psi(a) = 0$ so

$$0 = c_2 \sin\left(\frac{\sqrt{2mE}}{\hbar}a\right). \tag{2.12}$$

This can only be the case when

$$\frac{\sqrt{2mE}}{\hbar}a = \pm n\pi, \quad n = 0, 1, 2, ... \tag{2.13}$$

Thus we may solve for the energy of the sytem to be

$$E = \frac{1}{2m}\left(\frac{n\pi\hbar}{a}\right)^2, \quad n = 0, 1, 2, ... \tag{2.14}$$

Equation 2.14 shows that allowable values for the energy of the particle in the box will take only discrete values depending on the quantum number n. Thus the energy of the system is quantized.

2.3 Hydrogen Atom

The only molecule where a wavefunction may be calculated explicitly is a single hydrogen atom. We are able to solve 2.54 for the hydrogen atom where $n = k = 1$. The Hamiltonian is

$$-\frac{\hbar^2}{2\mu}\nabla^2 - \frac{Ze'^2}{r}. \tag{2.15}$$

μ is the center of mass between the nucleus and the electron and $e' = \frac{1.6\times10^{-19}}{\sqrt{4\pi\epsilon_0}}C$ is the charge of a single electron. We use spherical coordinates to define the wavefunction and we will use separation of variables to solve 2.54.

$$\psi(r, \theta, \phi) = R(r)\Theta(\theta)\Phi(\phi) \tag{2.16}$$

The Laplacian in spherical coordinates is

$$\nabla^2 = \frac{\partial^2}{\partial r^2} + \frac{2}{r}\frac{\partial}{\partial r} + \frac{1}{r^2}\frac{\partial^2}{\partial \theta^2} + \frac{1}{r^2}\cot\theta\frac{\partial}{\partial\theta} + \frac{1}{r^2\sin^2\theta}\frac{\partial^2}{\partial\phi^2}. \tag{2.17}$$

We begin by solving for $\Phi(\phi)$ by isolating all terms in 2.15 containing $\Phi(\phi)$ and setting them to be a constant

$$\frac{1}{\Phi(\phi)}\frac{\partial^{\Phi}(\phi)}{\partial\phi^2} = \alpha. \tag{2.18}$$

This will yield solutions

$$\Phi(\phi) = e^{\pm im\phi}, \quad m \in \mathbb{Z}, \quad \alpha = -m^2. \tag{2.19}$$

This solution contains the quantum number m. Now we use the solution for $\Phi(\phi)$ and isolate the terms containing $\Theta(\theta)$ setting them equal to the constant λ

$$\frac{1}{\Theta(\theta)}\left(\frac{\partial^2}{\partial\theta^2} + \cot\theta\frac{\partial}{\partial\theta} + \frac{\alpha}{\sin^2\theta}\right)\Theta(\theta) = \lambda. \tag{2.20}$$

Solutions to this type of equation can be found in [31, 41] and use the Legendre functions $P_l^{|m|}(x)$

$$\begin{aligned}\Theta(\theta) = P_l^{|m|}(\cos\theta) = (1-\cos^2\theta)\frac{d^{l+|m|}}{dx^{l+|m|}}(\cos^2\theta - 1)^l, \quad l = 0, 1, 2, ...\\ \text{where } \lambda = -l(l+1), \quad l \geq |m|.\end{aligned} \tag{2.21}$$

This solution gave us another quantum number, l. Finally we may use both 2.21 and 2.19 to solve for $R(r)$

$$\left[\frac{\partial^2}{\partial r^2} + \frac{2}{r}\frac{\partial}{\partial r} - \frac{l(l+1)}{r^2} - \frac{2\mu}{\hbar^2}\frac{Ze'}{r}\right]R(r) = \frac{-2\mu E}{\hbar^2}R(r). \tag{2.22}$$

Methods for solving this equation may also be found in [41] and will contain the Laguerre polynomials

$$L_{\alpha,\beta}(x) = \frac{\partial^\beta}{\partial x^\beta}\left(e^x\frac{\partial^\alpha}{\partial x^\alpha}x^\alpha e^{-x}\right). \tag{2.23}$$

Solutions to the final equation 2.22 contain the quantum number

$$B \geq l + 1, \tag{2.24}$$

$$R_{B,l}(r) = \left(\frac{r}{a}\right)^l e^{-\frac{r}{2a}} L_{B+l,2l+1}\left(\frac{r}{a}\right) \tag{2.25}$$
$$\text{where } a = \sqrt{\frac{\hbar^2}{8\mu E}}.$$

We may now state the full wavefunction solution to the hydrogen atom in terms of the quantum numbers n, l, and m using 2.25, 2.19, and 2.21

$$\psi_{B,l,m}(r,\theta,\phi) = \left(\frac{r}{a}\right)^l e^{-\frac{r}{2a}} L_{B+l,2l+1}\left(\frac{r}{a}\right)(1-\cos^2\theta)\frac{d^{l+|m|}}{dx^{l+|m|}}(\cos^2\theta - 1)^l e^{\pm im\phi}. \tag{2.26}$$

The associated energy of the wavefunction will be

$$E_n(\psi_{B,l,m}) = -\frac{\mu e'^4 Z^2}{2B^2\hbar^2}. \tag{2.27}$$

These exact solutions to the hydrogen atom can be used to approximate the orbitals electrons occupy in larger systems, atoms, and molecules.

2.4 Spin

The Hamiltonian which we used to find exact wavefunctions for the hydrogen atom was sufficient for a system with a single electron, but when there are multiple electrons present we must take into account an additional property called spin. Electrons have angular momentum due to their motion just like all other particles, but they also have an intrinsic spin angular momentum. A wavefunction for an electron must also incorporate its spin angular momentum. The spin of an electron has 2 possible values, $m_s = \frac{1}{2}$ and $m_s = -\frac{1}{2}$. The elements of the Hamiltonian that involve spin do not interact with the spatial variables so the wavefunction may be separated

$$\psi(r,\theta,\phi,m_s) = \psi(r,\theta,\phi)g(m_s). \tag{2.28}$$

This product of a spatial wavefunction with a spin function is called a spin-orbital. The common way to denote the spin functions is

$$g\left(\frac{1}{2}\right) = \alpha, \quad g\left(-\frac{1}{2}\right) = \beta \tag{2.29}$$

Spin will have no effect on the overall energy of the system since

$$\hat{H}\psi(r,\theta,\phi)\alpha = \alpha\hat{H}\psi(r,\theta,\phi) = \alpha E\psi(r,\theta,\phi) = E\psi(r,\theta,\phi)\alpha. \tag{2.30}$$

Even though the energy of the wavefunction remains unchanged by spin, the number of solutions doubles, due to the 2 values of m_s.

2.5 Calculating Energy

The only wavefunction that can be solved exactly is the wavefunction for a single hydrogen atom. In all other instances we must approximate the solution. There are a few methods for doing so. The variation method assumes a structure for the wavefunction that is dependent on certain parameters, then optimizes over those parameters. Perturbation theory approaches the problem by using a Hamiltonian with a known wavefunction solution and then slightly increasing the complexity of the Hamiltonian.

2.5.1 Variation Method

The first method for approximating the energy of a system is called the variation method. The idea is to assume a class of functions that will be the wavefunction for the system, then we find the optimal function among that class. For example, below we will assume that the solution to a system is of the form $x^2 - c$ and then we determine the optimal c. We find the optimal values for the parameters using the Rayleigh Quotient. For any operator, the Rayleigh quotient is bounded below by the least eigenvalue [24].

$$W = \frac{\int \psi^* \hat{H}\psi d\tau}{\int \psi^* \psi d\tau} \geq E_0. \tag{2.31}$$

Since we are able to evaluate the integrals in 2.31, we can find the ψ which minimizes the Rayleigh Quotient and use that as our approximate solution.

2.5.2 Example

We will solve the problem of the particle in a box with a slight modification, $V(x) = 0$ for $-a \leq x \leq a$. We will attempt to approximate the wavefunction with

$$\psi(x) = x^2 - c. \tag{2.32}$$

This will give us a Rayleigh quotient of

$$W = \frac{\int_{-a}^{a} (x^2 - c) \left[\frac{-\hbar^2}{2m} \frac{d^2}{dx^2} (x^2 - c) \right] dx}{\int_{-a}^{a} (x^2 - c)(x^2 - c) dx} \tag{2.33}$$

$$W = \frac{\frac{-\hbar^2}{m} \left[\frac{2}{3} a^3 - 2ca \right]}{\frac{2}{5} a^5 - \frac{4}{3} ca^3 + 2c^2 a} \tag{2.34}$$

Now we differentiate with respect to c and set it to 0 to get

$$4c^2a^2 - \frac{8}{3} ca^4 + \left(\frac{8}{9} - \frac{4}{5} \right) a^6 = 0. \tag{2.35}$$

Solving for c we have

$$c = \frac{1}{3} a^2 \pm a^2 \sqrt{\frac{4}{45}} \tag{2.36}$$

Now we have the optimal approximation for the energy of the system among function of the form 2.34 by substituting $c = \frac{1}{3} a^2 - a^2 \sqrt{\frac{4}{45}}$ back into 2.34

$$W \approx -\frac{1.6771 \hbar^2}{a^2 m} \tag{2.37}$$

A common choice for the form of the wavefunction is linear combinations of linearly independent basis functions

$$\psi = c_1 f_1 + c_2 f_2 + ... + c_n f_n = \sum_{i=1}^{n} c_i f_i. \tag{2.38}$$

Now we substitute this function into the parts of the Rayleigh quotient 2.31

$$\int \psi^* \hat{H} \psi d\tau = \int \sum_{i=1}^{n} c_i f_i \hat{H} \sum_{j=1}^{n} c_j f_j d\tau = \sum_{i=1}^{n} \sum_{j=1}^{n} c_i c_j \int f_i \hat{H} f_j d\tau = \sum_{i=1}^{n} \sum_{j=1}^{n} c_i c_j H_{ij} \tag{2.39}$$

$$\int f_i \hat{H} f_j d\tau = H_{ij}. \tag{2.40}$$

Similarly we have

$$\int \psi^* \psi d\tau = \sum_{i=1}^{n} \sum_{j=1}^{n} c_i c_j S_{ij} \tag{2.41}$$

$$\int f_i f_j d\tau = S_{ij}. \tag{2.42}$$

Thus

$$W = \frac{\sum_{i=1}^{n} \sum_{j=1}^{n} c_i c_j H_{ij}}{\sum_{i=1}^{n} \sum_{j=1}^{n} c_i c_j S_{ij}}. \tag{2.43}$$

Now we optimize over the parameters

$$\frac{\partial}{\partial c_k} \sum_{i=1}^{n} \sum_{j=1}^{n} c_i c_j H_{ij} = 2 \sum_{i=1}^{n} c_i H_{ik}. \tag{2.44}$$

Similarly

$$\frac{\partial}{\partial c_k} \sum_{i=1}^{n} \sum_{j=1}^{n} c_i c_j S_{ij} = 2 \sum_{i=1}^{n} c_i S_{ik}. \tag{2.45}$$

Now we differentiate the Rayleigh quotient

$$\frac{\partial W}{\partial c_k} = \frac{\left(\sum_{i=1}^{n} \sum_{j=1}^{n} c_i c_j S_{ij}\right)\left(2\sum_{i=1}^{n} c_i H_{ik}\right) - \left(\sum_{i=1}^{n} \sum_{j=1}^{n} c_i c_j H_{ij}\right)\left(2\sum_{i=1}^{n} c_i S_{ik}\right)}{\left(\sum_{i=1}^{n} \sum_{j=1}^{n} c_i c_j S_{ij}\right)^2}. \tag{2.46}$$

Next we set ∇W to 0 so we may find optimal values for the parameters.

$$\left(\sum_{i=1}^{n} \sum_{j=1}^{n} c_i c_j S_{ij}\right)\left(2\sum_{i=1}^{n} c_i H_{ik}\right) - \left(\sum_{i=1}^{n} \sum_{j=1}^{n} c_i c_j H_{ij}\right)\left(2\sum_{i=1}^{n} c_i S_{ik}\right) = 0 \tag{2.47}$$

$$\sum_{i=1}^{n} c_i H_{ik} = \frac{\sum_{i=1}^{n} \sum_{j=1}^{n} c_i c_j H_{ij}}{\sum_{i=1}^{n} \sum_{j=1}^{n} c_i c_j S_{ij}} \sum_{i=1}^{n} c_i S_{ik} \tag{2.48}$$

$$\sum_{i=1}^{n} c_i H_{ik} = W \sum_{i=1}^{n} c_i S_{ik} \tag{2.49}$$

$$\sum_{i=1}^{n} c_i \left[H_{ik} - W S_{ik}\right] = 0 \tag{2.50}$$

This gives us the following system of equations

$$\begin{bmatrix} H_{11} & H_{12} & \cdots & H_{1n} \\ H_{21} & H_{22} & \cdots & H_{2n} \\ \vdots & \ddots & \ddots & \vdots \\ H_{n1} & H_{n2} & \cdots & H_{nn} \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{bmatrix} - W \begin{bmatrix} S_{11} & S_{12} & \cdots & S_{1n} \\ S_{21} & S_{22} & \cdots & S_{2n} \\ \vdots & \ddots & \ddots & \vdots \\ S_{n1} & S_{n2} & \cdots & S_{nn} \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{bmatrix} = 0. \tag{2.51}$$

We may simplify this equation further by requiring that our basis functions f_i be orthonormal so that

$$\int f_i^* f_j d\tau = S_{ij} = \delta_{ij} \tag{2.52}$$

and we will have an eigenvalue problem

$$\begin{bmatrix} H_{11} & H_{12} & \cdots & H_{1n} \\ H_{21} & H_{22} & \cdots & H_{2n} \\ \vdots & \ddots & \ddots & \vdots \\ H_{n1} & H_{n2} & \cdots & H_{nn} \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{bmatrix} = W \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{bmatrix}. \tag{2.53}$$

2.5.3 Perturbation Theory

The second method for approximating the energy of a system is called perturbation theory. This method uses a solution to a system similar to the current one to construct an approximation. We assume that there is an incremental difference between the system with a known solution and the current one. Call that increment λ. We are looking for solutions to the time-independent Shrödinger equation

$$\hat{H}\psi = E\psi \tag{2.54}$$

We expand our wavefunction in a power series about the parameter λ

$$\psi_l = \psi_l|_{\lambda=0} + \frac{\partial \psi_l}{\partial \lambda}|_{\lambda=0}\lambda + \frac{\partial^2 \psi_l}{2\partial \lambda^2}|_{\lambda=0}\lambda^2 ... \tag{2.55}$$

Similarly we may expand the energy in a Taylor series

$$E_l = E_l|_{\lambda=0} + \frac{\partial E_l}{\partial \lambda}|_{\lambda=0}\lambda + \frac{\partial^2 E_l}{2\partial \lambda^2}|_{\lambda=0}\lambda^2 ... \tag{2.56}$$

Each of the partial derivates recieves a superscript to denote the order

$$\psi_l^{(k)} = \frac{\partial^k \psi_l}{k!\partial\lambda^k} \quad E_l^{(k)} = \frac{\partial^k E_l}{k!\partial\lambda^k} \tag{2.57}$$

$$\psi_l = \psi_l^{(0)} + \lambda\psi_l^{(1)} + \lambda^2\psi_l^{(2)}... \tag{2.58}$$

$$E_l = E_l^{(0)} + \lambda E_l^{(1)} + \lambda^2 E_l^{(2)}... \tag{2.59}$$

We do not know the value of ψ_l so we will use a nearby problem whose solution is known. Assume that if the Hamiltonian is H_0 in 2.54 then we have a known solution $\{\psi_l^0\}, \{E_l^0\}$ and that the wavefunctions of the solution are orthogonal

$$H_0\psi_l^0 = E_l^0\psi_l^0 \tag{2.60}$$

$$\int \psi_i^0\psi_j^0 = \delta_{ij}. \tag{2.61}$$

We separate our Hamiltonian

$$\hat{H} = H_0 + \lambda h' \tag{2.62}$$

where λ is small. We assume that the first order correction is the known solution to the nearby problem

$$\psi_l^{(0)} = \psi_l^0 \quad E_l^{(0)} = E_l^0 \tag{2.63}$$

We expand 2.54 in terms of our known solutions and the parameter λ

$$\begin{gathered}(H_0 + \lambda h')\left(\psi_l^{(0)} + \lambda\psi_l^{(1)} + \lambda^2\psi_l^{(2)} + ...\right) = \\ \left(E_l^{(0)} + \lambda E_l^{(1)} + \lambda^2 E_l^{(2)} + ...\right)\left(\psi_l^{(0)} + \lambda\psi_l^{(1)} + \lambda^2\psi_l^{(2)} + ...\right)\end{gathered} \tag{2.64}$$

Next we may operate and combine terms with like powers of λ

$$H_0\psi_l^{(0)} = E_l^{(0)}\psi_l^0 \tag{2.65}$$

$$(H_0 - E_l^{(0)})\psi_l^{(1)} = (E_l^{(1)} - h')\psi_l^{(0)} \tag{2.66}$$

$$(H_0 - E_l^{(0)})\psi_l^{(2)} = E_l^{(2)}\psi_l^{(0)} + E_l^{(1)}\psi_l^{(1)} + h'\psi_l^{(1)} \tag{2.67}$$

$$\vdots$$

Now we may use our set of known solutions, $\{\psi_l^0\}$ as a basis for $\psi_l^{(1)}$

$$\psi_l^{(1)} = \sum_k a_k \psi_k^0, \qquad \text{where } a_k = \langle \psi_k^{(0)} | \psi_l^{(1)} \rangle. \tag{2.68}$$

This expansion is substituted into 2.66

$$(H_0 - E_l^0) \sum_k a_k \psi_k^0 = \sum_k (E_k^0 - E_l^0) \psi_k^0 = (E_l^{(1)} - h') \psi_l^0. \tag{2.69}$$

If we multiply both sides of the equation by ψ_l^0 and then integrate we will have

$$\begin{aligned}
\int \psi_l^0 \sum_k (E_k^0 - E_l^0) \psi_k^0 &= \int \psi_l^0 (E_l^{(1)} - h') \psi_l \\
\sum_k \int \psi_l^0 (E_k^0 - E_l^0) \psi_k^0 &= \int \psi_l^0 (E_l^{(1)} - h') \psi_l^0 \\
\sum_k (E_k^0 - E_l^0) \langle \psi_l^0 | \psi_k^0 \rangle &= E_l^{(1)} \langle \psi_l^0 | \psi_l^0 \rangle - \langle \psi_l^0 | h' | \psi_l^0 \rangle && (2.70) \\
0 &= E_l^{(1)} - \langle \psi_l^0 | h' | \psi_l^0 \rangle \\
E_l^{(1)} &= \langle \psi_l^0 | h' | \psi_l^0 \rangle && (2.71)
\end{aligned}$$

In order find an expression for the wavefunction we need to determine the coefficients, a_k, so we expand it in terms of the orthonormal set of known solutions, $\{\psi_l^0\}$, as in equation 2.68. Next we substitute $a_k = \langle \psi_k^{(0)} | \psi_l^{(1)} \rangle$ into equation 2.70 and solve for a_k to obtain our expression

$$a_k = \frac{-\langle \psi_k^0 | h' | \psi_l^0 \rangle}{E_k^0 - E_l^0}. \tag{2.72}$$

Summing up over all k solutions to the original Hamiltonian, H_0, we may specifiy the first order correction energy and wavefunction

$$E_l = E_l^0 + \lambda E_l^{(1)} = E_l^0 + \langle \psi_l^0 | h' | \psi_l^0 \rangle \tag{2.73}$$

$$\psi_l = \psi_l^0 - \lambda \sum_{k \neq l} \frac{\langle \psi_k^0 | h' | \psi_l^0 \rangle}{E_k^0 - E_l^0} \psi_k^0. \tag{2.74}$$

We expect the power series to converge to the exact solution for the Hamiltonian specific to our problem, so we may truncate when we are satisfied with the accuracy of the current wavefunction. Both the variation method and perturbation theory require some initial information to the methods. For the variation method it is an initial wavefunction and for perturbation theory it is an initial Hamiltonian with solution. These methods are made much more efficient by addressing the initial values and Gaussian may compute energy using either method.

2.5.4 Determinantal Wavefunctions

In section 2.5.1 we computed a wavefunction using the variation method by first assuming the solution will have a specific structure, namely $\psi(x) = x^2 - c$. In this section we discuss the specific structure that the computational methods within Gaussian assume which are based on the behavior of electrons. In a system with multiple electrons each electron will occupy an orbital which is some product of spatial and spin functions. We denote the spatial functions by χ_i and number the electrons occupying the orbitals. If the first electron is occupying the first spin orbital with alpha spin we would have

$$\chi_1(1)\alpha(1). \tag{2.75}$$

These single electron spin orbitals will have the form 2.38. A wavefunction for a 2 electron system may be

$$\psi = \chi_1(1)\alpha(1) + \chi_1(2)\beta(2) \tag{2.76}$$

or

$$\psi = \frac{1}{2}\left(\chi_1(1)\alpha(1)\chi_1(2)\beta(2) - \chi_1(1)\beta(1)\chi_1(2)\alpha(2)\right) \tag{2.77}$$

where the 2 electrons occupy the same orbital with different spins. Wavefunctions of systems with multiple electrons have been shown to be antisymmetric [26]. This means that interchanging the locations or spin of 2 electrons negates the wavefunction. By defining the wavefunction for an N particle system as a determinant of an $N \times N$ matrix of a set of spin orbitals as in 2.77 the antisymmetry requirement will be upheld

$$\psi = \frac{1}{\sqrt{N!}}\det(\chi_1(1)\alpha(1), \chi_1(2)\beta(2), ...) =$$
$$\frac{1}{\sqrt{N!}}\begin{vmatrix} \chi_1(1)\alpha(1) & \chi_1(1)\beta(1) & \chi_2(1)\alpha(1) & ... \\ \chi_1(2)\alpha(2) & \chi_1(2)\beta(2) & \chi_2(2)\alpha(2) & \ddots \\ \vdots & \vdots & \vdots & \ddots \end{vmatrix}. \tag{2.78}$$

2.5.5 Hartree-Fock Self-Consistent Field Method

The Hartree-Fock method [46] is a specific application of the variation method which astutely separates the inner product $\langle\psi|\hat{H}|\psi\rangle$ by electronic interactions. We assume that there is a distinct spin orbital for each electron in an atom. The potential energy between two electrons is

$$\frac{Q_iQ_j}{4\pi\epsilon_0 r_{ij}} \tag{2.79}$$

We rearrange the Hamiltonian in terms of the electrons' interactions. Define

$$h_i = \frac{-\hbar^2}{2m}\nabla_i^2 - \sum_N \frac{Z_N}{r_{iN}} \tag{2.80}$$

to be the Hamiltonian of the kinetic energy and nuclear repulsion of the i^{th} electron. We next approach the interactions between the spin orbitals. Coulomb integrals are the repulsion between electrons occupying different spin orbitals

$$J_{ij} = \int\int \frac{|\chi_i(1)|^2|\chi_j(2)|^2}{r_{12}} dv_k dv_l \tag{2.81}$$

$$= \langle|\chi_i(1)|^2|\frac{1}{r_{12}}||\chi_j(2)|^2\rangle \tag{2.82}$$

Electrons may also exchange orbitals and this energy is

$$K_{ij} = -\int\int \frac{\chi_i^*(1)\chi_j^*(2)\chi_i(2)\chi_j(1)}{r_{12}} dv_1 dv_2 \tag{2.83}$$

$$= -\langle\chi_i(1)\chi_j(2)|\frac{1}{r_{12}}|\chi_i(2)\chi_j(1)\rangle. \tag{2.84}$$

Thus our full energy expression will be

$$E = \sum_i \langle\chi_i(i)|h_i|\chi_i(i)\rangle + \frac{1}{2}\sum_i\sum_j (J_{ij} - K_{ij})\,. \tag{2.85}$$

We do not know the spin orbitals that our electrons occupy, so in order to compute the energy of the molecule we must optimize the spin orbitals.

In order to determine the spin orbitals we use some complete set of orthogonal functions, $\{f_j\}$, which we call a basis set and let g be the spin of the orbital

$$\chi_i = \phi_i g = g \sum_{m=1}^{\infty} c_{im} f_m \tag{2.86}$$

Since the basis of functions is complete, this sum will be exact [24]. We cannot use the entire set of functions so we truncate the sum at some term

$$\phi_i \approx \sum_{m=1}^{k} c_{im} f_m. \tag{2.87}$$

We have now restated the energy to be a function of these coefficients and so the energy of the molecule is

$$\min_{c_{im}} E(c_{im}). \tag{2.88}$$

The orbitals comprising the wavefunction are orthogonal so we also introduce a Lagrange multiplier, ϵ, to address this constraint to obtain a new function for minimization

$$E' = E + \sum_k \epsilon_k (1 - \langle \phi_k | \phi_k \rangle). \tag{2.89}$$

In order to optimize we set the derivative to 0

$$\frac{\partial E'}{\partial c_{im}} = 0. \tag{2.90}$$

We differentiate the expression in parts

$$\frac{\partial \langle \phi_i(1) g(1) | h_i | \phi_i(1) g(1) \rangle}{\partial c_{im}} = 2 \langle f_m(1) g(1) | h_i | \chi_i(1) \rangle \rangle. \tag{2.91}$$

$$\frac{\partial \sum_j J_{ij}}{\partial c_{im}} = 2 \sum_j \langle f_m(1) g(1) \chi_i(1) | \frac{1}{r_{12}} || \chi_j(2)|^2 \rangle = 4 \langle f_m(1) g(1) \chi_i(1) | \frac{1}{r_{12}} | \rho(2) \rangle. \tag{2.92}$$

$$\rho(k) = \sum_{i=1}^{n} |\chi_i(k)|^2. \tag{2.93}$$

ρ is called the density matrix.

$$\begin{aligned}
\frac{\partial \sum_j K_{ij}}{\partial c_{im}} &= -\sum_j \langle f_m(1)g(1)\chi_j(2)|\frac{1}{r_{12}}|\chi_i(2)\chi_j(1)\rangle \\
&\quad +\langle \chi_i(1)\chi_j(2)|\frac{1}{r_{12}}|f_m(2)g(2)\chi_j(1)\rangle \\
&= -2\sum_j \langle f_m(1)\chi_i(2)|\frac{1}{r_{12}}|\chi_j(2)\chi_j(1)\rangle \\
&= -4\langle f_m(1)\chi_i(2)|\frac{1}{r_{12}}|\gamma(1,2)\rangle \qquad (2.94) \\
\text{where} \quad \gamma(1,2) &= \sum_i \chi_i(1)\chi_i(2) \qquad (2.95)
\end{aligned}$$

γ is called the exchange matrix. Finally we differentiate the multiplier term

$$\frac{\partial \epsilon_i(1 - \langle \phi_i(1)g(1)|\phi_i(1)g(1)\rangle)}{\partial c_{im}} = -2\epsilon_i\langle f_m(1)|\chi_i(1)\rangle. \qquad (2.96)$$

Combining our terms we have

$$\begin{aligned}
&\langle f_m(1)g(1)|h_i|\chi_i(1)\rangle\rangle + \langle f_m(1)g(1)\chi_i(1)|\frac{1}{r_{12}}|\rho(2)\rangle \\
&-\langle f_m(1)\chi_i(2)|\frac{1}{r_{12}}|\gamma(1,2)\rangle - \epsilon_i\langle f_m(1)|\chi_i(1)\rangle = 0. \qquad (2.97)
\end{aligned}$$

Now we may expand (we drop the spin term for simplicity since it does not affect the calculation)

$$\chi_i = \sum_{j=1}^{k} c_{ij} f_j \qquad (2.98)$$

$$\begin{aligned}
&\sum_{j=1}^{k} c_{ij}[\langle f_m(1)|h_i|f_j(1)\rangle + \langle f_m(1)f_j(1)|\frac{1}{r_{12}}|\rho(2)\rangle \\
&-\langle f_m(1)f_j(2)|\frac{1}{r_{12}}|\gamma(1,2)\rangle - \epsilon_m\langle f_m(1)|f_j(1)\rangle] = 0. \qquad (2.99)
\end{aligned}$$

Thus we have the eigenvalue equation

$$\sum_j (F_{mj} - \epsilon_m \delta_{mj}) c_{mj} = 0 \quad (2.100)$$

$$F_{mj} = \langle f_m(1)|h_i|f_j(1)\rangle + \langle f_m(1)f_j(1)|\frac{1}{r_{12}}|\rho(2)\rangle - \langle f_m(1)f_j(2)|\frac{1}{r_{12}}|\gamma(1,2)\rangle. \quad (2.101)$$

2.6 Geometry Optimization

Using the methods shown in 2.5.5 we may compute the energy of a molecule for any values of the molecule's atom's coordinates, p. Hamilton's principle tells us that the molecule will minimize its energy in any quantum state

$$E_n = \min_p E_n(p). \quad (2.102)$$

This optimization is performed using a method called direct inversion in the iterative subspace (DIIS) [42, 13, 12, 27, 30, 51]. DIIS is a quasi-Newton method [23] which constructs the consecutive iterates using a linear combination of potential solutions.

All quasi-Newton methods use an approximate Hessian which is updated as the optimization proceeds. The steps are as follows

- Begin at x_c
- Compute $d = -H_c^{-1}\nabla f(x_c)$
- $x_+ = x_c + \lambda d$
- update H_c to H_+

For $E(p)$ we are able to compute an analytic gradient, $\nabla E(p)$.

2.6.1 Energy Gradient

The gradient of the energy can be found using the Hellman-Feynman theorem [26]. We are differentiating the energy expression with respect to some coordinate, p

$$\frac{\partial}{\partial p}E = \frac{\partial}{\partial p}\int \psi^* \hat{H}\psi d\tau \tag{2.103}$$

$$\frac{\partial E}{\partial p} = \int \frac{\partial}{\partial p}\left(\psi^* \hat{H}\psi d\right)\tau \tag{2.104}$$

$$= \int \left(\frac{\partial \psi^*}{\partial p}\right)\hat{H}\psi d\tau + \int \psi^* \left(\frac{\partial \hat{H}\psi}{\partial p}\right) d\tau. \tag{2.105}$$

First we differentiate the term with the Hamiltonian

$$\frac{\partial}{\partial p}\left(\hat{H}\psi\right) = \frac{\partial \hat{H}}{\partial p}\psi + \hat{H}\frac{\partial \psi}{\partial p}. \tag{2.106}$$

So we have

$$\frac{\partial}{\partial p}E = \int \frac{\partial \psi *}{\partial p}\hat{H}\psi d\tau + \int \psi^* \frac{\partial \hat{H}}{\partial p}\psi d\tau + \int \psi^* \hat{H}\frac{\partial \psi}{\partial p} d\tau. \tag{2.107}$$

The first term can be evaluated using the Schrödinger equation

$$\int \frac{\partial \psi^*}{\partial p}\hat{H}\psi d\tau = E\int \frac{\partial \psi *}{\partial p}\psi d\tau \tag{2.108}$$

The last term may be evaluated using the fact that the Hamiltonian is Hermitian

$$\int \psi^* \hat{H}\frac{\partial \psi}{\partial p} d\tau = \int \frac{\partial \psi}{\partial p}\left(\hat{H}\psi\right)^* d\tau = E\int \psi^* \frac{\partial \psi}{\partial p} d\tau. \tag{2.109}$$

Since the wavefunction is normalized we know that

$$\int \psi^* \psi d\tau = 1 \Rightarrow \frac{\partial}{\partial p}\int \psi^* \psi d\tau = 0. \tag{2.110}$$

Thus

$$\frac{\partial E}{\partial p} = \int \psi^* \frac{\partial \hat{H}}{\partial p}\psi d\tau. \tag{2.111}$$

Now we apply the Hellman-Feynman theorem to the Hamiltonian for our system of

k atoms and n electrons

$$\hat{H} = -\frac{\hbar}{2m}\sum_{i=1}^{n}\nabla_i^2 - \sum_{i=1}^{n}\sum_{j=1}^{k}\frac{Z_j}{r_{ij}} + \sum_{i=1}^{n-1}\sum_{j=i+1}^{n}\frac{1}{r_{ij}} + \sum_{i=1}^{k-1}\sum_{i+1}^{k}\frac{Z_iZ_j}{r_{ij}}. \tag{2.112}$$

We take $p = x_p$ to be a nuclear cartesian coordinate in the x direction and now differentiate

$$\frac{\partial \hat{H}}{\partial x_p} = \frac{\partial}{\partial x_p}\frac{-\hbar}{2m}\sum_{i=1}^{n}\nabla_i^2 - \frac{\partial}{\partial x_p}\sum_{i=1}^{n}\sum_{j=1}^{k}\frac{Z_j}{r_{ij}} + \frac{\partial}{\partial x_p}\sum_{i=1}^{n-1}\sum_{j=i+1}^{n}\frac{1}{r_{ij}} + \frac{\partial}{\partial x_p}\sum_{i=1}^{k-1}\sum_{i+1}^{k}\frac{Z_iZ_j}{r_{ij}} \tag{2.113}$$

$$= -\frac{\partial}{\partial x_p}\sum_{i=1}^{n}\sum_{j=1}^{k}\frac{Z_j}{r_{ij}} + \frac{\partial}{\partial x_p}\sum_{i=1}^{k-1}\sum_{i+1}^{k}\frac{Z_iZ_j}{r_{ij}} \tag{2.114}$$

$$= -Z_p\sum_{i=1}^{n}\frac{(x_i - x_p)}{r_{ip}^3} + Z_p\sum_{j\neq p}^{k}\frac{Z_j(x_j - x_p)}{r_{jp}^3}. \tag{2.115}$$

Thus we have an expression for the gradient of the wavefunction

$$\frac{\partial E}{\partial x_p} = \int \psi^*\left(-Z_p\sum_{i=1}^{n}\frac{(x_i - x_p)}{r_{ip}^3} + Z_p\sum_{j\neq p}^{k}\frac{Z_j(x_j - x_p)}{r_{jp}^3}\right)\psi d\tau. \tag{2.116}$$

2.6.2 Pulay Mixing

We are solving the problem

$$\min_p E(p). \tag{2.117}$$

Pulay mixing[42, 13] begins with a set of coordinates, p_k, and an approximate Hessian, H_k [50]. Next we obtain a set of vectors, p_0^i, through the following set of iterations

$$p_0^0 = p_0 \tag{2.118}$$

$$p_0^1 = p_0^0 - H_0^{-1}\nabla E(p_0^0) \tag{2.119}$$

$$p_0^2 = p_0^1 - H_0^{-1}\nabla E(p_0^1) = p_0 - H_0^{-1}\nabla E(p_0^0) - H_0^{-1}\nabla E(p_0^1) \tag{2.120}$$

$$p_0^i = p_0 - \sum_{j=0}^{i} H_0^{-1}\nabla E(p_0^j) \tag{2.121}$$

We continue to generate these vectors, p_0^i, until the $m+1^{\text{st}}$ vector is nearly a linear combination of the other m vectors (in a least-squares sense). These m vectors are then used to construct the iterate

$$p_{k+1} = \sum_{i=1}^{m} c_i p_k^i \quad \text{by solving} \tag{2.122}$$

$$\min_{c_i} \sum_{i=1}^{m} c_i (p_k^i - p_k^{i-1}) = \sum_{i=1}^{m} c_i \Delta p_k^i \tag{2.123}$$

$$\text{subject to} \quad \sum_{i=1}^{m} c_i = 1. \tag{2.124}$$

This can be solved using a Lagrange multiplier, λ,

$$\begin{bmatrix} B_{11} & \dots & B_{1m} & -1 \\ \vdots & \ddots & \vdots & \vdots \\ B_{m1} & \dots & B_{mm} & -1 \\ -1 & \dots & -1 & 0 \end{bmatrix} \begin{bmatrix} c_1 \\ \vdots \\ c_m \\ \lambda \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ -1 \end{bmatrix} \tag{2.125}$$

$$B_{ij} = \Delta p_k^i \Delta p_k^j \tag{2.126}$$

Once we have determined a new iterate, p_{k+1}, DIIS uses the BFGS update scheme [23] for computing a new approximate Hessian

$$H_+ = H_c + \frac{yy^T}{y^T s} - \frac{(H_c s)(H_c s)^T}{s^T H_c s} \tag{2.127}$$

$$s = x_+ - x_c \tag{2.128}$$

$$y = \nabla f(x_+) - \nabla f(x_c). \tag{2.129}$$

Geometry optimizations use more convergence criteria than a standard quasi-Newton method [27], which would normally terminate at a small gradient. The four criteria are

- RMS of the force is less than 10^{-5}
- Maximum single force is less than 1.5×10^{-5}
- RMS of the geometry displacement is less than 4×10^{-5}
- Maximum single displacement is less than 6×10^{-5}

Chapter 3

Interpolation

3.1 Background

The simulations we have developed have used two different methods of interpolation. Initially we constructed the simulation to use cubic splines. The advantage of using splines is that there are many robust codes available for implementation. Similarly the implementation is also eased by the regular pattern of the nodes. This regular pattern of nodes causes us problems in high dimensional simulations since the number of nodes grows exponentially in the degrees of freedom as we discuss in Section 5.3.

The interpolation method that we found superior to splines for our purposes was developed by Smolyak. The Smolyak algorithm [56, 60] has been utilized by the numerical integration community to perform integration in high dimension for some time [61, 36, 19, 37, 4] due to the fact that it has a few pleasing characteristics:

- The method is exact for polynomials of selected degree k in d dimensions.
- Although the algorithm does not use the minimum number of nodes for a degree k interpolation in d dimensions, the number of nodes grows polynomially with the degrees of freedom.
- The nodes are well dispersed throughout the domain.
- There is a simple formula for determining the nodes given degree k and dimension d.

The polynomial growth in the number of nodes gives us the opportunity to feasibly run simulations in up to 10 degrees of freedom. By choosing nested nodes for the Smolyak algorithm we also develop error estimation techniques in Chapter 6.

3.2 Cubic Spline

A cubic spline approximates a function by connecting a set of cubic polynomials on a domain[18, 43]. The advantage to this construction is that there is global smoothness without using a high degree polynomial. Given a set of $n+1$ well-ordered nodes, $\{x_0, ..., x_n\}$, and the corresponding function values at each node, $\{y_0, ..., y_n\}$ any cubic spline, $p(x)$, will satisfy three criteria:

1. On each subinterval $[x_i, x_{i+1}]$ p(x) is a cubic polynomial.
2. $p(x_i) = y_i$ for $i = 0...n$.
3. $p(x)$ is twice continuously differentiable within $[x_0, x_n]$.

This leaves the spline underdetermined because there are a total of n intervals. Each interval requires 4 coefficients so there are a total of $4n$ coefficients to be determined. Condition 2 accounts for $2n-2$ equations and condition 3 accounts for $2n$ equations so we are left with 2 degrees of freedom at the boundary of the domain. There are three common types of conditions to fully determine the spline:

1. A *clamped spline* has $p'(x_0)$ and $p'(x_n)$ predetermined.
2. A *natural spline* has $p''(x_0) = p''(x_n) = 0$.
3. A *periodic spline* had $p^j(x_0) = p^j(x_n)$ for $j = 1, 2$.

3.2.1 Derivation

Let $M_i = p''(x_i)$ and $p_i(x)$ be the cubic polynomial on the interval $[x_i, x_i + 1]$. Since these are cubic polynomials $p''(x)$ is piecewise linear, thus

$$p_i''(x) = M_i \frac{x_{i+1} - x}{h_i} + M_{i+1} \frac{x - x_i}{h_i} \tag{3.1}$$

where $h_i = x_{i+1} - x_i$. Next integrate twice to get

$$p_i(x) = \frac{M_i(x_{i+1} - x)^3}{6h_i} + \frac{M_{i+1}(x - x_i)^3}{6h_i} + c_i(x - x_i) + d_i \tag{3.2}$$

Now we may substitute our constraints $p_i(x_i) = y_i$ and $p_i(x_{i+1}) = y_{i+1}$ into $p_i(x)$ to determine the constants c_i and d_i.

$$c_i = \frac{y_{i+1} - y_i}{h_i} - \frac{h_i}{6}(M_{i+1} - M_i) \tag{3.3}$$

$$d_i = y_i - M_i \frac{h_i^2}{6}. \tag{3.4}$$

The values of M_i can be determined by imposing the continuity constraint on the first derivative

$$p_i'(x_i) = -\frac{M_i(x_{i+1} - x_i)^2}{2h_i} + c_i = \frac{M_i(x_i - x_{i-1})^2}{2h_{i-1}} + c_{i-1} = p_{i-1}'(x_i) \tag{3.5}$$

This gives us a three term recurrence relation for the values M_i

$$\frac{h_{i-1}}{6} M_{i-1} + \left(\frac{h_i}{2} - \frac{h_i}{6} + \frac{h_{i-1}}{2} - \frac{h_{i-1}}{6} \right) M_i + \frac{h_i}{6} M_{i+1} = \frac{y_{i+1} - y_i}{h_i} - \frac{y_i - y_{i-1}}{h_{i-1}} \tag{3.6}$$

Which may be simplified further to

$$\frac{h_{i-1}}{h_i + h_{i-1}} M_{i-1} + 2M_i + \frac{h_i}{h_i + h_{i-1}} M_{i+1} = \frac{6}{h_i + h_{i-1}} \left(\frac{y_{i+1} - y_i}{h_i} - \frac{y_i - y_{i-1}}{h_{i-1}} \right). \tag{3.7}$$

The final constraints determined by the type of spline complete the system.

3.3 Smolyak Interpolation

3.3.1 Formulation in a Single Dimension

Assume we would like to approximate the value of a function

$$f : [a, b] \to \mathbb{R} \tag{3.8}$$

at some point in the domain $[a, b]$. Given a set of $g + 1$ nodes

$$\{x_i\} \in [a, b] \quad i = 1...k + 1 \tag{3.9}$$

and the corresponding set of function values

$$\{f(x_i) \in \mathbb{R}\} \quad i = 1...k + 1 \tag{3.10}$$

there exists a unique polynomial $\Pi_k(x)$ of degree k such that

$$\Pi_k(x_i) = f(x_i) \quad \text{for } i = 1...k + 1 \tag{3.11}$$

The equation for that polynomial is

$$\Pi_k(x) = \sum_{i=1}^{k+1} f(x_i) l_i(x) \tag{3.12}$$

Where $l_i(x)$ is the ith Lagrange interpolating polynomial:

$$l_i(x) = \prod_{j=1, j\neq i}^{k+1} \frac{x - x_j}{x_i - x_j} \tag{3.13}$$

The interpolation will be exact at each node since

$$l_i(x_j) = \begin{cases} 1 & \text{if } j = i \\ 0 & \text{if } j \neq i \end{cases} \tag{3.14}$$

In order to determine the nodes we assume

$$f : [-1, 1] \to \mathbb{R}. \tag{3.15}$$

Once the nodes are determined we scale $[-1, 1]$ to any desired rectangular domain $[a, b]$. In this work our nodes are the extrema of the Chebyshev polynomials[5]. For a degree k interpolating polynomial the $k + 1$ extrema which will be the nodes are

$$x_i = -\cos \frac{\pi(i - 1)}{k - 1}, \quad i = 1...k + 1. \tag{3.16}$$

By selecting only values of k so that

$$k(z) = 2^{z-1} \quad z \in \mathbb{N} \tag{3.17}$$

we obtain a set of nodes that are nested. This means that the set of nodes determined by $z+1$, denoted by $\{x_i\}_{z+1}$, will contain the every node in the set determined by z.

$$\{x_i\}_z \subset \{x_i\}_{z+1} \tag{3.18}$$

3.3.2 Example

If $z = 3$, then $k(z) = 4$. We obtain the following set of nodes

$$x_i = -\cos\frac{\pi(i-1)}{4}, \quad i = 1, 2, 3, 4, 5 \tag{3.19}$$

$$\{x_i\}_3 = \{-1, -\frac{\sqrt{2}}{2}, 0, \frac{\sqrt{2}}{2}, 1\} \tag{3.20}$$

Increasing z by 1 we have $z = 4$ and $k(z) = 8$, giving us a new set of nodes, the nodes $\{x_i\}_3$ are in bold.

$$x_i = -\cos\frac{\pi(i-1)}{8}, \quad i = 1, 2, 3, 4, 5, 6, 7, 8, 9 \tag{3.21}$$

$$\{x_i\}_4 = \{-\mathbf{1}, -0.9239, -\frac{\sqrt{\mathbf{2}}}{\mathbf{2}}, -0.3827, \mathbf{0}, 0.3827, \frac{\sqrt{\mathbf{2}}}{\mathbf{2}}, 0.9239, \mathbf{1}\} \tag{3.22}$$

3.3.3 Formulation in High Dimension

Multi-dimensional Lagrange interpolation requires a multivariate analog to the Lagrange polynomial. We expect that polynomial to exhibit the property that

$$l_i(Y_j) = \begin{cases} 1 & \text{if } j = i \\ 0 & \text{if } j \neq i \end{cases} \tag{3.23}$$

if Y_j are the set of nodes in $\mathbb{R}^d$. We accomplish this by taking the product of the single dimension polynomials assuming that the i^{th} node corresponds to the single coordinate nodes $\{m_1, ..., m_d\}$ for each of the variables $\{y_1, ..., y_d\}$ [9, 49, 16]

$$Li(y_1, .., y_d) = l_{m_1}(y_1)\dot{\dots} l_{m_d}(y_d) = \prod_{i=1}^{d} l_{m_i}(y_i). \tag{3.24}$$

In constructing an interpolation in dimension $d > 1$ we want to preserve the properties of the one dimension interpolation [48]. We will be approximating a function

$$f : [-1, 1]^d \to \mathbb{R} \tag{3.25}$$

where once again we may scale the unit cube to any desired domain. For each of the d directions $r = 1, ..., d$ we have a one-dimensional interpolation of degree $k(z)$ from 3.12

$$U^z(x) = \Pi_{k(z)}(x). \tag{3.26}$$

Let $\mathbf{Z}$ be the multi-index of length d containing the individual values of z that determine the degree of the interpolation in each of the d coordinate directions. Thus $U^{Z_r}(x)$ is the interpolant in the rth coordinate.

3.3.4 Example

If $d = 3$ and $\mathbf{Z} = (2, 3, 2)$ then we would have a $k(Z_1) = 2^{Z_1 - 1} = 2$ degree interpolation in the first direction, $U^2(x) = \Pi_2(x)$. Similarly we have $2^{Z_2 - 1} = 4$ and $2^{Z_3 - 1} = 2$ degree interpolations in the other 2 respective directions.

Define points Y in $\mathbb{R}^d$ by the variables y_r in each direction

$$Y = (y_1, y_2, ..., y_d) \tag{3.27}$$

Now we define a tensor product between d one-dimensional interpolations in the d different directions with degree determined by $\mathbf{Z}$

$$\mathbf{U}^{\mathbf{Z}}(Y) = \bigotimes_{r=1}^{d} U^{Z_r}(x) = \sum_{s_1=1}^{k(Z_1)+1} \sum_{s_2=1}^{k(Z_2)+1} \dots \sum_{s_N=1}^{k(Z_N)+1} f(x_{s_1}, x_{s_2}, ..., x_{s_N}) \prod_{r=1}^{d} l_{s_r}(y_r) \tag{3.28}$$

3.3.5 Example

Let $d = 2$ and $\mathbf{Z} = (2, 2)$. Our tensor product would look like

$$\mathbf{U}^{(2,2)}(y_1, y_2) = U^2(x) \otimes U^2(x) = \sum_{s_1=1}^{3} \sum_{s_2=1}^{3} f(x_{s_1}, x_{s_2}) l_{s1}(y_1) l_{s_2}(y_2) \tag{3.29}$$

The Chebyshev extrema associated with $z = 2$ can be found with 3.16. They are $\{x_i\}_2 = \{-1, 0, 1\}$. So we obtain

$$\begin{aligned}
\mathbf{U}^{(2,2)}(y_1, y_2) = f(-1,-1)&\frac{(y_1-0)(y_1-1)}{(-1-0)(-1-1)}\frac{(y_2-0)(y_2-1)}{(-1-0)(-1-1)} + \\
f(-1,0)&\frac{(y_1-0)(y_1-1)}{(-1-0)(-1-1)}\frac{(y_2+1)(y_2-1)}{(0+1)(0-1)} + \\
f(-1,1)&\frac{(y_1-0)(y_1-1)}{(-1-0)(-1-1)}\frac{(y_2+1)(y_2-0)}{(1+1)(1-0)} + \\
f(0,-1)&\frac{(y_1+1)(y_1-1)}{(0+1)(0-1)}\frac{(y_2-0)(y_2-1)}{(-1-0)(-1-1)} + \\
f(0,0)&\frac{(y_1+1)(y_1-1)}{(0+1)(0-1)}\frac{(y_2+1)(y_2-1)}{(0+1)(0-1)} + \\
f(0,1)&\frac{(y_1+1)(y_1-1)}{(0+1)(0-1)}\frac{(y_2+1)(y_2-0)}{(1+1)(1-0)} + \\
f(1,-1)&\frac{(y_1+1)(y_1-0)}{(1+1)(1-0)}\frac{(y_2-0)(y_2-1)}{(-1-0)(-1-1)} + \\
f(1,0)&\frac{(y_1+1)(y_1-0)}{(1+1)(1-0)}\frac{(y_2+1)(y_2-1)}{(0+1)(0-1)} + \\
f(1,1)&\frac{(y_1+1)(y_1-0)}{(1+1)(1-0)}\frac{(y_2+1)(y_2-0)}{(1+1)(1-0)}
\end{aligned} \tag{3.30}$$

Smolyak's algorithm uses linear combinations of the tensor products 3.28. We derive the formula in the same identically to [60]. The user determines a degree of one-dimensional polynomial exactness which is maintained in the multi-dimensional interpolation. For a d dimensional interpolation with exactness k, define

$$q = d + k \tag{3.31}$$

We define a difference of single dimension interpolations

$$\Delta^0 = 0, \quad \Delta^z(x) = U^z(x) - U^{z-1}(x) \tag{3.32}$$

We also define

$$\|\mathbf{Z}\| = \sum_{r=1}^{N} Z_r \tag{3.33}$$

and recall that $\mathbf{Z}$ contains the degrees of the interpolation in each direction so every element is non-negative. Smolyak's interpolation is a set of tensor products of the single dimension difference functions $\Delta^z(x)$

$$A(q,d)(Y) = \sum_{\|\mathbf{Z}\| \leq q} \bigotimes_{r=1}^{d} \Delta^{Z_r}(x) \tag{3.34}$$

Since we have $\Delta^0 = 0$ any $\mathbf{Z}$ containing an element that is 0 will not be included in the sum. Thus we can be sure that $\mathbf{Z} \geq \mathbf{1}$, where $\mathbf{1}$ is a vector of length N with every element equal to 1. Define the set of allowable multi-indices $\mathbf{Z}$

$$Q(q,d) = \{\mathbf{Z} \in \mathbb{R}^d | \mathbf{Z} \geq \mathbf{1}, \|\mathbf{Z}\| \leq q\} \tag{3.35}$$

The cardinality of the set $Q(q,d)$ is $\binom{q}{d}$. We have

$$\begin{aligned} A(q,d) = \sum_{\mathbf{Z} \in Q(q,d)} \bigotimes_{r=1}^{d} \Delta^{Z_r} &= \sum_{\mathbf{Z} \in Q(q-1,d-1)} \left(\bigotimes_{r=1}^{d-1} \Delta^{Z_r} \right) \otimes \sum_{Z_n=1}^{q-\|\mathbf{Z}\|} \Delta^{Z_d} \\ &= \sum_{\mathbf{Z} \in Q(q-1,d-1)} \left(\bigotimes_{r=1}^{d-1} \Delta^{Z_r} \right) \otimes U_{q-\|\mathbf{Z}\|}, \end{aligned} \tag{3.36}$$

since we obtain a telescoping series using 3.32

$$\sum_{i=1}^{m} \Delta^i = U^m. \tag{3.37}$$

We now derive an explicit form of $A(q,d)$ for all indices $\mathbf{Z}$ for which $Z_l = j_l + \alpha_l$ with $\alpha \in \{0,1\}^d$ and $\|\alpha\| \leq q - \|j\|$. Furthermore, the sign of $\otimes_{l=1}^{d} U^{j_l}$ in this case is $(-1)^{|\alpha|}$.

Let

$$b(z,N) = \sum_{\alpha\in\{0,1\}^d, |\alpha|\le z} (-1)^{|\alpha|} \tag{3.38}$$

This and 3.36 yield

$$A(q,d) = \sum_{j\in Q(q,d)} b(q-|j|,d) \bigotimes_{l=1}^{d} U^{j_l}. \tag{3.39}$$

We now compute $b(z,d)$. Clearly, we can sum with respect to $|\alpha| = 0,1,...,d$. Since $|\alpha| = j$ corresponds to $\binom{d}{j}$ terms, we have

$$b(z,d) = \sum_{j=0}^{\min z,d} \binom{d}{j} (-1)^j = (-1)^z \binom{d-1}{z}. \tag{3.40}$$

In particular, $b(z,d) = 0$ for $z \ge d$. Thus

$$A(q,d)(Y) = \sum_{\mathbf{Z}\in Q(q,d)} (-1)^{q-\|\mathbf{Z}\|} \binom{d-1}{q-\|\mathbf{Z}\|} \bigotimes_{r=1}^{d} U^{Z_r}(x). \tag{3.41}$$

3.3.6 Example

We take $d = 2$, $k = 2$, so $q = d + k = 4$. Generate

$$\begin{aligned} &Q(4,2) = \{\mathbf{Z} \in \mathbb{R}^2 | \mathbf{Z} \ge \mathbf{1}, \|\mathbf{Z}\| \le 4\} = \\ &\{(1,3),(3,1),(2,2),(2,1),(1,2),(1,1)\}. \end{aligned} \tag{3.42}$$

Knowing that 3 is the largest element of $\mathbf{Z}$ we generate the sets of one dimensional interpolation points from formulas 3.16 and 3.17 for z=3,2,1.

$$z = 3 \Rightarrow k(z) = 4 \qquad \{x_i\}_3 = \{-1, -\tfrac{\sqrt{2}}{2}, 0, \tfrac{\sqrt{2}}{2}, 1\} \tag{3.43}$$

$$z = 2 \Rightarrow k(z) = 2 \qquad \{x_i\}_2 = \{-1, 0, 1\} \tag{3.44}$$

$$z = 1 \Rightarrow k(z) = 1 \qquad \{x_i\}_1 = \{0\} \tag{3.45}$$

Each pair (a,b) in Q(4,2) defines a set of points defined by the cartesian product of $\{x_a\}$ with $\{x_b\}$. For instance, if we begin with the first element of Q, (1,3), we have the following points at which the function needs to be evaluated

$$\{x_3\} \times \{x_1\} = \{(0,-1)(0,-\frac{\sqrt{2}}{2}),(0,0),(0,\frac{\sqrt{2}}{2}),(0,1)\}. \tag{3.46}$$

Since (1,3) will also be an element of Q, the reflection of these points along the line y=x will also be points for evaluation. For the element (2,2) we have these interpolation points

$$\{x_2\} \times \{x_2\} = \{(-1,-1),(-1,0),(-1,1),(0,-1),(0,0),(0,1),(1,-1),(1,0),(1,1)\} \tag{3.47}$$

The entire set of points can be observed in Figure 3.1 (a) along with the grids for successively larger values of k. It is important to note that each grid contains all of the points from the previous grid. These grids contain fewer points than a spline using grids derived from the same one-dimensional set of points as in figure 3.1 (d).

3.4 Complexity

We derive a bound for the number of gridpoints in the same manner as [37]. Let X_j^i be the set of nodes used by the single dimension interpolation U_j^i. Denote the number of nodes as num(X_j^i) and $X_j^{i+1} \setminus X_j^i$ be the nodes contained in X_j^{i+1} that are not in X_j^i. Denote the grid for the interpolation scheme $A(d+k,d)$ to be $H(d+k,d)$.

Theorem 3.4.1. *For some constant value of k as $d \to \infty$ the number of nodes for Smolyak's algorithm grows polynomially in d. That is*

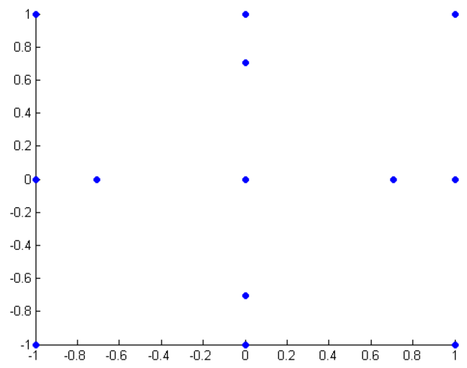
$$num(H(d+k,d)) \approx \frac{2^k d^k}{k!} \tag{3.48}$$

Proof. The sparse grid defined by $A(d+k,d)$ has the grid

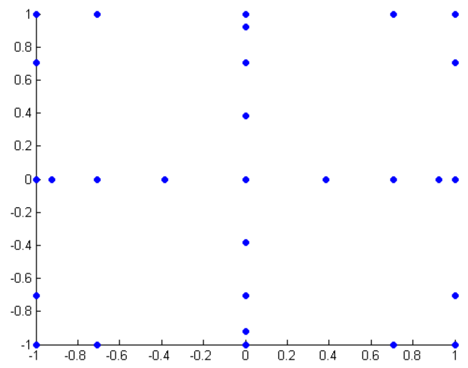
$$H(d+k,d) = \bigcup_{k+1 \le \|\mathbf{i}\| \le d+k} X_1^{i_1} \times ... \times X_d^{i_d}. \tag{3.49}$$

Due to the fact that $X_j^0 = \emptyset$ and num$(X_j^2 \setminus X_j^1) = 2$ we have

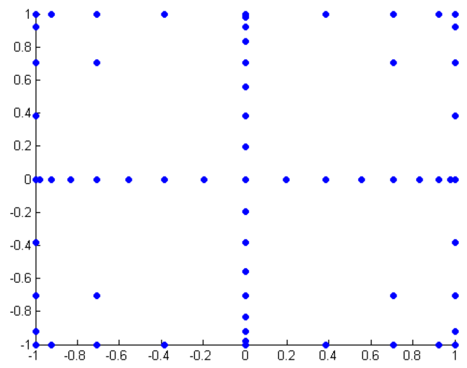
$$H(d+k,d) \supset \bigcup_{\|\mathbf{i}\|=d+k, i_m \le 2\,\forall m} \left(X_1^{i_1} \setminus X_1^{i_1-1}\right) \times ... \times \left(X_d^{i_d} \setminus X_d^{i_d-1}\right). \tag{3.50}$$



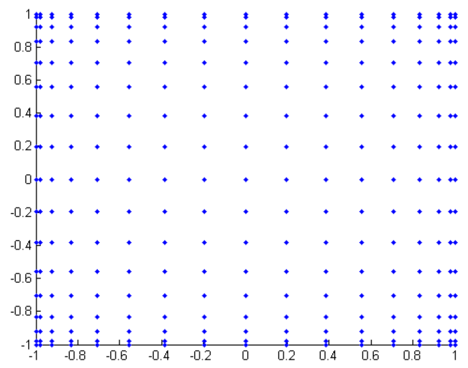
(a) Smolyak points for N=2, d=2



(b) Smolyak points for N=2, d=3



(c) Smolyak points for N=2, d=4



(d) Spline grid using the same points as Smolyak for N=2, d=4

Figure 3.1: Nested sets of interpolation points as d increases.

Since we are going to let d be large, we may assume that $k \leq d$, thus that gives us a lower bound on the number of nodes

$$\text{num}\left(H(k+d,d)\right) \geq \binom{d}{k} \cdot 2^k. \tag{3.51}$$

If x is a node of $H(d+k,d)$, then there may only be k coordinates of x which are not members of X_j^1. Let $J = \{j_1, ..., j_v\}$ be a set of directions so that $\{x_j \notin X_j^1 | \forall j \in J\}$. The cardinality of J is $v \leq k$. If $v = k$ we have

$$x_j \in X_j^2 \setminus X_j^1 \quad \forall j \in J. \tag{3.52}$$

The number of nodes $x \in H(d+k,d)$ so that $x_j \notin X_j^1$ if and only if $j \in J$ is bounded above by 2^k. If $v < k$, then

$$x_j \in \bigcup_{i=1}^{k} X_j^i \;\; \forall j \in J. \tag{3.53}$$

The number of nodes $x \in H(d+k,d)$ so that $x_j \notin X_j^1$ if and only if $j \in J$ in this case is found by simple counting to be

$$\left(\sum_{i=1}^{k} \mathrm{num}(X^i)\right)^k = c_k. \tag{3.54}$$

So we obtain a bound on $num(H(d+k,d))$ by counting the number of nodes for all possible J

$$\mathrm{num}(H(d+k,d)) \le \sum_{v=0}^{k-1} \binom{d}{v} \cdot c_k + \binom{d}{k} \cdot 2^k \le k \cdot d^{k-1} \cdot c_k + \binom{d}{k} \cdot 2^k. \tag{3.55}$$

Now we let $d \to \infty$ and remind the reader that $d >> k$ so we have

$$\lim_{d\to\infty} \mathrm{num}(H(d+k,d)) \le \lim_{d\to\infty} k \cdot d^{k-1} \cdot c_k + \tfrac{d!}{k!(d-k)!} \cdot 2^k \tag{3.56}$$

$$= \lim_{d\to\infty} k \cdot d^{k-1} \cdot c_k + d(d-1)...(d-k+1) \cdot \tfrac{2^k}{k!} \tag{3.57}$$

$$\approx \tfrac{2^k d^k}{k!} \tag{3.58}$$

□

3.5 Interpolation Error

3.5.1 1 Dimension

Let the operator Π_k be the interpolating polynomial at $n+1$ distinct nodes.

Theorem 3.5.1. *Let ω_{k+1} be the nodal polynomial at the $n+1$ interpolation nodes*

$$\omega_{k+1} = \prod_{i=0}^{k} (x - x_i). \tag{3.59}$$

Then the single dimension interpolation error is

$$E_k(x) = \frac{f^{(k+1)}(\xi)}{(k+1)!}\omega_{k+1}(x) \leq \frac{f^{(k+1)}(\xi)}{(k+1)!}H^{k+1} \tag{3.60}$$

where H is the maximal distance between gridpoints.

Proof. If $x = x_i$ then $E_k(x) = 0$. Assume $x \in [x_0, x_k]$, but $x \neq x_i$. Define the polynomial

$$P(t) = E_k(t) - \frac{\omega_{k+1}(t)E_k(x)}{\omega_{k+1}(x)} \tag{3.61}$$

then $P(t)$ has at least $k+2$ zeros in $[x_0, x_k]$. There are $k+1$ zeros at the interpolation nodes as well as a zero when $t = x$. By the mean value theorem $P'(t)$ has at least $k+1$ zeros. We may continue to differentiate and we have $P^{k+1}(t)$ has at least 1 zero. Since $\Pi_k f$ is a degree k polynomial we have

$$E_k^{(k+1)}(t) = f^{(k+1)}(t). \tag{3.62}$$

Similarly

$$\omega_{k+1}^{(k+1)}(t) = k+1!. \tag{3.63}$$

Thus we have

$$P^{(k+1)}(t) = f^{(k+1)}(t) - \frac{(k+1)!E_k(x)}{\omega_{k+1}(x)}. \tag{3.64}$$

We know that $P^{(k+1)}(t)$ has at least one root. Let $\xi \in [x_0, x_k]$ be such that $P(\xi) = 0$, then we have

$$P^{(k+1)}(\xi) = f^{(k+1)}(\xi) - \frac{(k+1)!E_k(x)}{\omega_{k+1}(x)} \tag{3.65}$$

$$f^{(k+1)}(\xi) = \frac{(k+1)!E_k(x)}{\omega_{k+1}(x)} \tag{3.66}$$

$$E_k(x) = \frac{f^{(k+1)}(\xi)}{(k+1)!}\omega_{k+1}(x). \tag{3.67}$$

Since for all x within the interpolation interval, $x - x_i \leq H$ we have

$$\omega_{k+1}(x) \leq H^{k+1}$$

$$E_k(x) = \frac{f^{(k+1)}(\xi)}{(k+1)!}\omega_{k+1}(x) \leq \frac{f^{(k+1)}(\xi)}{(k+1)!}H^{k+1}. \tag{3.68}$$

□

3.5.2 2 Dimensions

Define

$$D_x^m f = \frac{\partial^m}{\partial x^m} f. \tag{3.69}$$

Let k_x be the number of interpolation nodes in the x coordinate and H the maximal distance between nodes.

Theorem 3.5.2. *The 2 dimensional interpolation error is given by*

$$\begin{aligned}\|f - \Pi f\| \leq \; & \tfrac{\|D_y^{(k_y+1)} f\|}{(k_y+1)!} H^{(k_y+1)} + \tfrac{\|D_x^{(k_x+1)} f\|}{(k_x+1)!} H^{(k_x+1)} \\ & + \tfrac{\|D_y^{(k_y+1)} D_x^{(k_x+1)} f\|}{(k_y+1)! \cdot (k_x+1)!} H^{k_y+1} H^{k_x+1}\end{aligned} \tag{3.70}$$

Proof. Define the interpolation operators in coordinate directions Π_x and Π_y. So we have

$$\Pi f = \Pi_x \Pi_y f = \Pi_y \Pi_x f \tag{3.71}$$

Now we use the triangle inequality

$$\|f - \Pi f\| \leq \|f - \Pi_y f\| + \|\Pi y f - \Pi f\| \tag{3.72}$$

The first term can be handled by 3.60

$$\|f - \Pi_y f\| \leq \frac{\|D_y^{(k_y+1)} f\|}{(k_y+1)!} H^{k_y+1}. \tag{3.73}$$

The second term is the interpolation error of the x-interpolant operating on the y-interpolant

$$\|\Pi_y f - \Pi_x(\Pi_y f)\| \leq \frac{\|D_x^{(k_x+1)}\Pi_y f\|}{(k_x+1)!}H^{k_x+1} = \frac{\|\Pi_y D_x^{(k_x+1)} f\|_\infty}{(k_x+1)!}H^{k_x+1}. \tag{3.74}$$

$\Pi_y D_x^{(k_x+1)} f$ is the interpolation in y of $D_x^{(k_x+1)} f$ so we may apply 3.60

$$\|D_x^{(k_x+1)} f - \Pi_y D_x^{(k_x+1)} f\| \leq \frac{\|D_y^{(k_y+1)} D_x^{(k_x+1)} f\|}{(k_y+1)!}H^{k_y+1}. \tag{3.75}$$

Now we use the triangle inequality again

$$\begin{aligned}\|\Pi_y D_x^{(k_x+1)} f\| &= \|\Pi_y D_x^{(k_x+1)} f - D_x^{(k_x+1)} f + D_x^{(k_x+1)} f\| \\ &\leq \|D_x^{(k_x+1)} f\| + \frac{\|D_y^{(k_y+1)} D_x^{(k_x+1)} f\|}{(k_y+1)!}H^{k_y+1}.\end{aligned} \tag{3.76}$$

We substitute 3.76 into 3.74

$$\|\Pi_y f - \Pi_x(\Pi_y f)\| \leq \frac{\|D_x^{(k_x+1)} f\|}{(k_x+1)!}H^{(k_x+1)} + \frac{\|D_y^{(k_y+1)} D_x^{(k_x+1)} f\|}{(k_y+1)! \cdot (k_x+1)!}H^{k_y+1}H^{k_x+1}. \tag{3.77}$$

Combining the two terms from 3.72 we have a bound

$$\begin{aligned}\|f - \Pi f\| \leq \quad & \tfrac{\|D_y^{(k_y+1)} f\|}{(k_y+1)!}H^{(k_y+1)} + \tfrac{\|D_x^{(k_x+1)} f\|}{(k_x+1)!}H^{(k_x+1)} \\ & + \tfrac{\|D_y^{(k_y+1)} D_x^{(k_x+1)} f\|}{(k_y+1)! \cdot (k_x+1)!}H^{k_y+1}H^{k_x+1} = \mathcal{O}(H^{\min(k_x,k_y)+1})\end{aligned} \tag{3.78}$$

□

3.5.3 d Dimensions

Let the d degrees of freedom be the variables x_1 through x_d and we will assume that the same set of one dimensional interpolation nodes is used for each of the degrees of freedom and that number of nodes is n.

Theorem 3.5.3. *Let $\sigma(m, j)$ be the set of all combinations of j integers between 1 and m, so that elements of $\sigma(m, j)$ are multi-indices. For instance $(2, 3, 7) \in \sigma(8, 3)$. The* *interpolation error in d dimensions is given by*

$$\|f - \Pi f\| \leq \sum_{j=1}^{d} \left[\sum_{\boldsymbol{I} \in \sigma(d,j)} \frac{\|D_{x_{I_1}}^{n+1} ... D_{x_{I_j}}^{n+1} f\|}{((n+1)!)^j} H^{j(n+1)} \right] = \mathcal{O}(H^{n+1}) \tag{3.79}$$

Proof. We will prove by induction. Assume that for $d-1$ variables the interpolation $\Pi^{(d-1)}$

$$\|f - \Pi^{(d-1)} f\| \leq \sum_{j=1}^{d-1} \left[\sum_{\mathbf{I} \in \sigma(d-1,j)} \frac{\|D_{x_{I_1}}^{n+1} ... D_{x_{I_j}}^{n+1} f\|}{((n+1)!)^j} H^{j(n+1)} \right] \tag{3.80}$$

Now we use the triangle inequality

$$\|f - \Pi f\| \leq \|f - \Pi^{(d-1)} f\| + \|\Pi^{(d-1)} f - \Pi f\| \tag{3.81}$$

The first term is exactly the error given by 3.80 so we only need handle the the second term. Once again we have the single dimension interpolation error of the $d-1$ dimension interpolating polynomial

$$\|\Pi^{(d-1)} f - \Pi_{x_d}(\Pi^{(d-1)} f)\| \leq \frac{\|D_{x_d}^{(k_{x_d}+1)} \Pi^{(d-1)} f\|}{(k_{x_d}+1)!} H^{k_{x_d}+1} \leq \frac{\|\Pi^{(d-1)} D_{x_d}^{(k_{x_d}+1)} f\|}{(k_{x_d}+1)!} H^{k_{x_d}+1}. \tag{3.82}$$

The triangle inequality again gives us

$$\begin{aligned} \|\Pi^{(d-1)} D_{x_d}^{(k_{x_d}+1)} f\| &\leq \|\Pi^{(d-1)} D_{x_d}^{(k_{x_d}+1)} f - D_{x_d}^{(k_{x_d}+1)} f + D_{x_d}^{(k_{x_d}+1)} f\| \\ &\leq \|\Pi^{(d-1)} D_{x_d}^{(k_{x_d}+1)} f - D_{x_d}^{(k_{x_d}+1)} f\| + \|D_{x_d}^{(k_{x_d}+1)} f\|. \end{aligned} \tag{3.83}$$

The second term is given by 3.80 with $D_{x_d}^{(k_{x_d}+1)} f$ replacing f and so we may combine all our terms to get our formula

$$\|f - \Pi f\| \leq \|f - \Pi^{(d-1)} f\| + \|\Pi^{(d-1)} f - \Pi f\| \tag{3.84}$$

$$\leq \|f - \Pi^{(d-1)} f\| + \frac{\|\Pi^{(d-1)} D_{x_d}^{(k_{x_d}+1)} f - D_{x_d}^{(k_{x_d}+1)} f\| + \|D_{x_d}^{(k_{x_d}+1)} f\|}{(k_{x_d}+1)!} H^{k_{x_d}+1} \tag{3.85}$$

$$\leq \sum_{j=1}^{d} \left[\sum_{\mathbf{I} \in \sigma(d,j)} \frac{\|D_{x_{I_1}}^{n+1} ... D_{x_{I_j}}^{n+1} f\|}{((n+1)!)^j} H^{j(n+1)} \right] \tag{3.86}$$

□

3.5.4 Taylor Series

We can develop useful error estimates for our interpolation using multivariable Taylor's theorem. Taylor polynomials in multiple variables do not give rise to as simple a formula as the univariate case, so we will need to develop some notation.

3.5.5 Example

For a quadratic approximation using a Taylor polynomial we have

$$f(x+hu) = f(x) + h\nabla f(x)^T u + \frac{h^2}{2!}u^T\nabla^2 f(x)u + \mathcal{O}(h^3). \tag{3.87}$$

Were we to desire a cubic term this notation would fail us.

For the multivariate case we proceed as in [7], let

$$\Delta x = \begin{bmatrix} x_1 - u_1 & x_2 - u_2 & ... & x_d - u_d \end{bmatrix} \tag{3.88}$$

and

$$\nabla = \begin{bmatrix} \frac{\partial}{\partial x_1} & \frac{\partial}{\partial x_1} & ... & \frac{\partial}{\partial x_d} \end{bmatrix}^T \tag{3.89}$$

then we have the differential operator

$$\Delta x \cdot \nabla = \Delta x_1 \frac{\partial}{\partial x_1} + ...\Delta x_d \frac{\partial}{\partial x_d}. \tag{3.90}$$

We also use the analog to the binomial expansion

$$\binom{r}{p_1 p_2 ... p_d} = \frac{r!}{p_1! p_2! ... p_d!}, \qquad p_1 + p_2 + ... + p_d = r. \tag{3.91}$$

The expansion for higher degree differentials will be

$$(\Delta x \cdot \nabla)^r = \sum_{p_1+...+p_d=r} \binom{r}{p_1 ... p_d} (\Delta x_1)^{p_1} ... (\Delta x_d)^{p_d} \frac{\partial^r}{\partial x_1^{p_1} ... \partial x_d^{p_d}}. \tag{3.92}$$

Finally we may define the m^{th} degree Taylor operator and polynomial

$$T_m = \sum_{r=0}^{m} \frac{1}{r!} (\Delta x \cdot \nabla)^r \tag{3.93}$$

$$T_m f = \sum_{r=0}^{m} \frac{1}{r!} (\Delta x \cdot \nabla)^r f(x) \approx f(x + \Delta x) \tag{3.94}$$

along with the remainder

$$R_{m,x}(\Delta x) = \frac{1}{m!} \int_0^1 (\Delta x \cdot \nabla)^{m+1} f(x + t\Delta x)(1-t)^m dt \tag{3.95}$$

which we may approximate by

$$R_{m,x}(\Delta x) \approx \frac{1}{(m+1)!} (\Delta x \cdot \nabla)^{m+1} f(x + \theta \Delta x) \qquad 0 \leq \theta \leq 1 \tag{3.96}$$

$$= \mathcal{O}(H^{m+1}) \qquad \Delta x_i \leq H \quad \forall i. \tag{3.97}$$

3.5.6 Exactness

In one dimension Lagrange interpolation will exactly interpolate polynomials of degree k if there are k+1 distinct nodes used for the interpolation. Smolyak interpolation will exactly interpolate all polynomials of a certain degree as well, but that degree is not as easily discernible. Let $\mathbb{P}_m$ be the space of polynomials in one variable of degree m or less.

Theorem 3.5.4. *The interpolation formula*

$$A(q,d)(Y) = \sum_{\mathbf{Z} \in Q(q,d)} (-1)^{q - \|\mathbf{Z}\|} \binom{d-1}{q - \|\mathbf{Z}\|} \bigotimes_{r=1}^{d} U^{Z_r}(x) \tag{3.98}$$

will exactly reproduce all polynomials of the form

$$\sum_{|i|=q} \left(\mathbb{P}_{m_{i_1}} \otimes ... \otimes \mathbb{P}_{m_{i_d}} \right) \tag{3.99}$$

Proof. We prove this by induction on d [36]. For the case where $d = 1$ we have $A(q,1) = U^q$ which is exact. In the case that $d > 1$ the function is a product of univariate polynomials

$$f = f_{i_1} \otimes ... \otimes f_{i_N}. \tag{3.100}$$

We use the first Smolyak formula 3.34 and assume that $A(q, d)$ is exact

$$A(q, d+1) = \sum_{\|\mathbf{Z}\| \leq q} \bigotimes_{r=1}^{d+1} \Delta^{Z_r} = \sum_{l=d}^{q-1} A(l, d) \otimes \Delta^{q-l} \tag{3.101}$$

$$A(q, d+1)f = \sum_{l=d}^{q-1} A(l, d) \left(f_{i_1} \otimes ... \otimes f_{i_d}\right) \cdot \left(\Delta^{q-l} f_{i_{d+1}}\right). \tag{3.102}$$

Let

$$m = \sum_{j=1}^{d} i_j \tag{3.103}$$

so that

$$q = m + i_{d+1}. \tag{3.104}$$

We know that

$$A(l, d) \left(f_{i_1} \otimes ... \otimes f_{i_d}\right) = f_{i_1} \otimes ... \otimes f_{i_d} = f_d, \qquad \sum i_n = l < q \tag{3.105}$$

from the induction assumption. Also by the single dimension interpolation properties

$$U^{q-l} f_{i_{d+1}} = U^{q-l-1} f_{i_{d+1}} = f_{i_{d+1}}. \tag{3.106}$$

So we may write 3.102 as

$$A(q, d+1)f = \sum_{l=m}^{q-1} A(l, d) \left(f_{i_1} \otimes ... \otimes f_{i_d}\right) \cdot \left(U^{q-l} - U^{q-l-1}\right) \left(f_{i_{d+1}}\right). \tag{3.107}$$

Since $U^0 = 0$ this series telescopes to leave a single remaining term

$$A(q, d+1)f = A(l, d) \left(f_{i_1} \otimes ... \otimes f_{i_d}\right) \cdot U^{q-m} f_{i_{d+1}} = f_{i_1} \otimes ... \otimes f_{i_{d+1}} = f_{d+1}. \tag{3.108}$$

$\square$

This means that $A(d + k, d)$ is exact for all polynomials of degree less than or equal to k.

3.5.7 Example

Let $d = 3$ and $k = 2$ so $q = 5$, then each polynomial of degree 2 in 3 variables is a linear combination from a basis of monomials

$$\{x^2, y^2, z^2, xy, xz, yz, x, y, z, 1\}. \tag{3.109}$$

We form the multiindices which combine to form the interpolating polynomial

$$\{(3,1,1),(1,3,1),(1,1,3),(2,2,1),(2,1,2),(1,2,2),(2,1,1),(1,2,1),(1,1,2),(1,1,1)\}. \tag{3.110}$$

Recall that the indices are used to form nested single dimension interpolating polynomial of degree $k(z) = 2^{z-1}$ from equation 3.17. Now we observe the types of terms that each of the indexes give us

$$\begin{aligned}
(3,1,1) &\rightarrow x^4, x^3, ... \\
(1,3,1) &\rightarrow y^4, y^3, ... \\
(1,1,3) &\rightarrow z^4, z^3, ... \\
(2,2,1) &\rightarrow x^2y^2, x^2y, ... \\
(2,1,2) &\rightarrow x^2z^2, x^2z, ... \\
(1,2,2) &\rightarrow y^2z^2, y^2z, ... \\
(2,1,1) &\rightarrow x^2, x, ... \\
(1,2,1) &\rightarrow y^2, y, ... \\
(1,1,2) &\rightarrow z^2, z, ... \\
(1,1,1) &\rightarrow 1.
\end{aligned} \tag{3.111}$$

While we do obtain every term in the basis for quadratic polynomials in 3 dimensions, we cannot recover every cubic polynomial due to the absence of the cross-term xyz.

3.5.8 Error Based on Exactness

We would like to combine the exactness above with Taylor's theorem to get an approximation of the error based on the exactness of the approximation instead of the number of gridpoints as in Section 3.5.3. A bound on the operator was developed in [60] which we will restate here.

Theorem 3.5.5. *Assume the following all hold for the case* $d = 1$

$$\|I_1 - U^i\| \leq CD^i, \quad \forall i \geq 0 \tag{3.112}$$

$$\|\Delta_i\| = \|U^i - U^{i-1}\| \leq ED^i, \quad \forall i \geq 1 \tag{3.113}$$

then

$$e(A) = \|I_d - A(d+k, d)\| \leq CH^{d-1}\binom{d+k}{d-1}D^{d+k} \qquad \textit{where } H = \max(1/D, E) \tag{3.114}$$

For any function f we have

$$\|I_d(f) - A(f)\| \leq e(A)\|f\|. \tag{3.115}$$

Theorem 3.5.6. *For a function, f, Smolyak's formula with predetermined exactness k will yield an approximation with an error on the order of $k + 1$.*

Proof. Since these operators are linear we split the function with its Taylor series of degree k

$$f = T_k f + (f - T_k f) \tag{3.116}$$

$$\|I_d(f) - A(f)\| = \|I_d(T_k f) - A(T_k f) + I_d(f - T_k f) - A(f - T_k f)\| \tag{3.117}$$

Using the triangle inequality along with the polynomial exactness we have

$$\|I_d(f) - A(f)\| \leq \|I_d(T_k f) - A(T_k f)\| + \|I_d(f - T_k f) - A(f - T_k f)\| \tag{3.118}$$

$$\leq 0 + e(A)\|f - T_k f\| = \mathcal{O}(H^{k+1}) \tag{3.119}$$

□

Chapter 4

Surface Construction

4.1 Surrogate Models

The geometry of a molecule with N atoms is uniquely determined by $3N-6$ coordinates. The potential energy of a molecule is a function of all of the coordinates, p, as well as its quantum state, n.

$$E_n(p) \quad n = 0, 1, ... \tag{4.1}$$

In order to perform these simulations in a reasonable amount of time we must simplify the problem to use fewer coordinates. We achieve this by partitioning coordinate space into design variables, x, and dependent variables, ξ

$$p = (x, \xi). \tag{4.2}$$

The energy at any value of the design variables, x, on the ground state is determined by minimizing the energy as a function of the dependent variables

$$E_0(x) = \min_{\xi} E_0(x, \xi). \tag{4.3}$$

This optimization is performed internally by Gaussian and we describe the method in Section 2.6, but it is a slow and expensive calculation. We built our simulation by decreasing the number of times that we need to perform the optimization in Equation

4.3. This is done by constructing a surrogate model for the energy function

$$E_0(x) \approx E_0^s(x). \tag{4.4}$$

The surrogate model is an interpolant of the energy function calculated at pre-specified gridpoints. By using this method of pre-specified gridpoints we can take advantage of parallel computations to build the surrogate quickly, potentially in the time it takes for a single optimization if there are enough processors. The surrogate models allow for fast computation of the energy at any point within the interpolation domain and we can also easily construct an analytic gradient for the surrogate to approximate the gradient of the actual energy function. Each of the methods described in the rest of this chapter focus on the calculation of the actual energy at all of the interpolation nodes for the surrogate.

4.2 Excited States

Even more difficult than the ground state energy optimization in Equation 4.3 is an optimization in the excited state

$$E_n(x) = \min_{\xi} E_n(x, \xi), \quad n \neq 0. \tag{4.5}$$

There are some methods for performing this calculation [59, 15], but they are inadequate for dealing with larger molecules.

The common way to approximate the excited state energy is through vertical excitation [58]. This means we optimize the geometry on the ground state and then compute the excited state energy for the ground state geometry

$$E_n(x) \approx E_n(x, \xi_j) \;\text{ where }\; E_0(x, \xi_j) \leq E_0(x, \xi_j + \delta). \tag{4.6}$$

The excited state surrogate is constructed identically to the previous section, but it is important to note that the potential energy that we are using is no longer a local minimum geometry.

4.3 Gaussian Scan

We first used the tools within Gaussian to draw the potential energy surfaces (PES). This is done by specifying the angles to be rotated, the number of degrees per rotation, and the number of rotations. Gaussian also allows the user to choose the number of processors for parallel computation. The energy optimizations are performed in parallel, but each of the gridpoints are evaluated in serial.

We have identified 2 problems with Gaussian's internal scan. Since the gridpoints must be evaluated in serial, Gaussian's internal scan is an inefficient method for high-performance clusters. Since $E(x) = E(x, \xi)$ is found by optimizing ξ, Gaussian needs a good initial iterate, ξ_0 for each gridpoint. By leaving the choice of initial iterate to gaussian, the optimization is often unable to converge. When there is a convergence failure of a gridpoint, the entire scan terminates instead of computing the energy of the next point in the succession. Secondly, we have found that the surface generated when this scan is successful may be inconsistent with the physics. These surfaces are expected to be C^2, and at the very least the energy should depend continuously on the angles [32]. Figures 4.1 and 4.2 were both generated by a Gaussian scan and the large jump discontinuities are clearly visible.

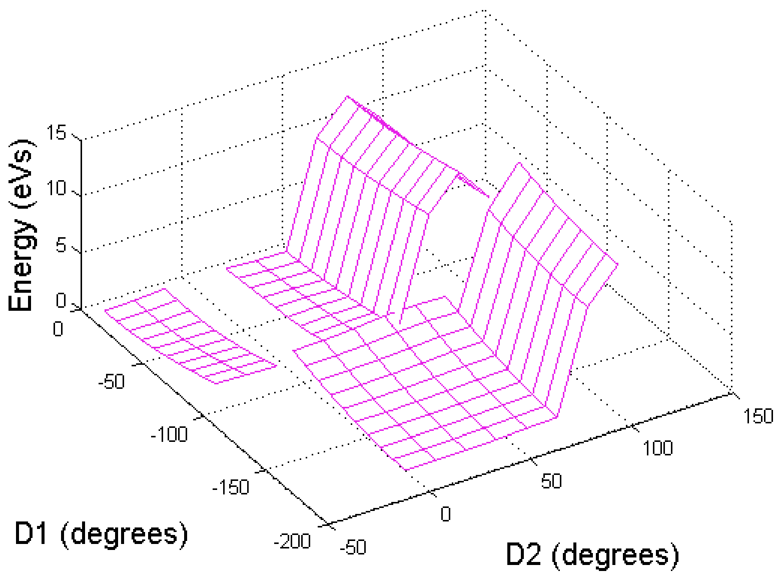


Figure 4.1: PES generated by 2 Gaussian scans for Stilbene

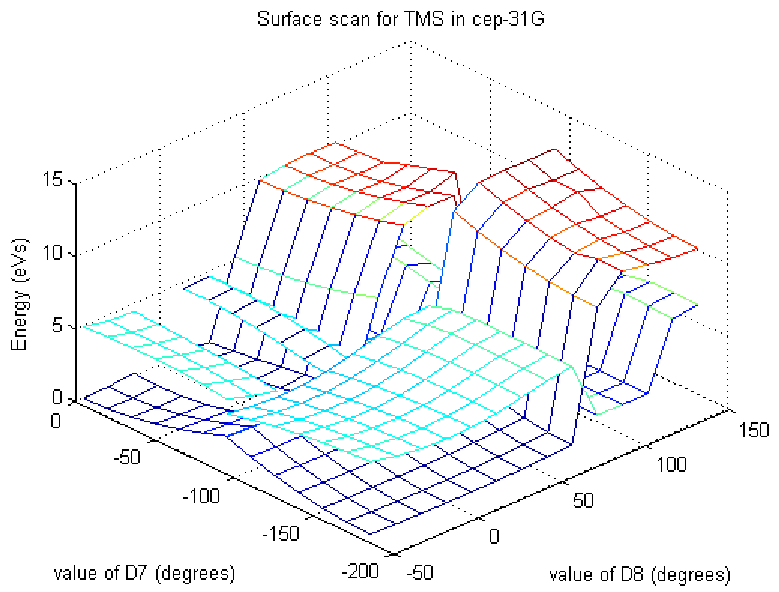


Figure 4.2: PES generated with 2 Gaussian scans for TMS

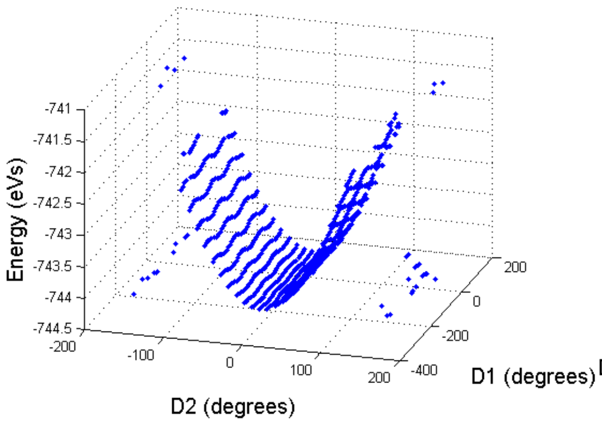
4.4 Single Initial Iterate

Since Gaussian computes the energy at each gridpoint in serial, it is highly inefficient to run its scan on a large cluster. We begin all of our surface constructions with a single known Z-matrix, $(x_0, \xi(x_0))$. The fastest way to compute an entire PES would be to compute the energy for each point on a specified grid simultaneously on separate processors. Each of those separate energy computations needs an initial Z-matrix for the optimizations. At each gridpoint the value of the design variables, x_i, is fixed, but the remaining coordinates in the Z-matrix, the dependent variables ξ, must be specified with an initial value in order for the optimization to begin. The simplest guess for the initial iterate, ξ, is the value of those variables in the original Z-matrix, then each gridpoint will be the result of the following optimization

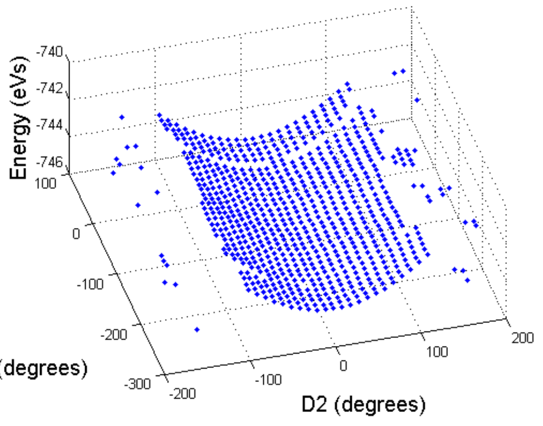
$$E^i = \min_{\xi} E(x_i, \xi(x_0)). \tag{4.7}$$

Where $\xi(x_0)$ is the optimal value of the dependent variables at the initial point of the scan. If we have d independent variables, then the initial iterate at each gridpoint will have $3N - 6 - d$ coordinates that are identical.

While this method is extremely efficient in parallel performance it also does not always produce feasible results. Although the surface looks to be differentiable almost everywhere, there are still large discontinuities which remains inconsistent with the physics, as seen in figure 4.3.



(a)



(b)

Figure 4.3: Two views of PES for 2-butene computed in parallel with the same initial iterate at each point

4.5 Expanding Perimeter

For Quasi-Newton method optimizations to successfully converge they must have a good initial iterate. The previous surfaces used the same initial iterate, $\xi(x_0)$, at each gridpoint. Clearly the dependent variables from the initial Z-matrix, $\xi(x_0)$, are good initial iterates for some of the gridpoints since there is a smooth lower section in Figure 4.1. In order to manage good iterates and take advantage of parallelism we used a succession of gridpoints [34]. Each point on the grid, assuming it converges successfully, should be a good initial iterate for any points surrounding it. This is due to the fact that the surfaces are at least twice differentiable [32]. The scheme is illustrated in figure 4.4. Since the user begins with one Z-matrix (in our case it is a local minimizer), the dependent variables in that Z-matrix, $\xi(x_0)$, serve as the initial iterate for the 8 points in a rectangle surrounding it on a 2-dimensional grid (labeled 0 through 8). Once the optimizations at each of those 8 gridpoints have returned (converged or not) a new rectangle is produced. Each optimized geometry's dependent variables, $\xi(x_i)$ for $i = 1...8$, in the finished round will serve as the initial iterate for a new point on the next rectangle. The corners of the rectangle are the initial iterates for 3 points on the next round (e.g. point 1 is the iterate for points 9, 10, and 24). In the event that a file fails to converge, the nearest point of a lesser number replaces it as the initial iterate for the point (e.g. if point 12 did not converge then point 11 will be the initial iterate for both 28 and 29). Each round has 8 gridpoints more than the previous round. Each gridpoint in a round is submitted independently as a Gaussian job and this parallelism immensely speeds up the computations.

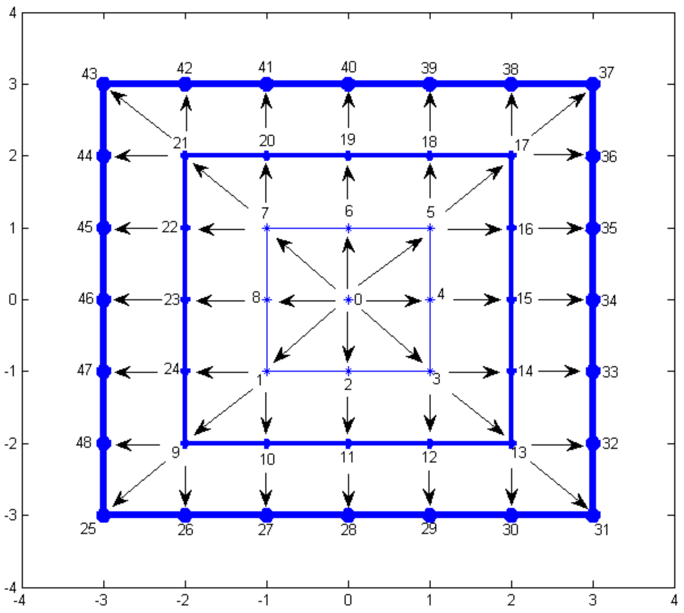


Figure 4.4: 2-D expanding perimeter scheme

The expanding perimeter algorithm in 2 dimensions is:

While this method of continuation accomplishes its goal of using nearby converged geometries as the initial iterates, it does not work perfectly. The PES still had some large discontinuities as in Figure 4.6. In order to properly use the expanding perimeter algorithm to draw a smooth PES for stilbene from a single point, we also had to incorporate some of our expectations about the physics of the molecule. Figure 4.5 shows the location of D1 which is the rotation of the middle bond. Both atoms making up the central bond are also bonded to hydrogen atoms. As the angle D1 rotates we can expect the hydrogens to rotate symmetrically along with it so that the ring and hydrogen will

Algorithm 4.1 Expanding Perimeter 2-Dimensional Algorithm

```
input $x_0$, $\xi(x_0)$, step, num_steps
Calculate E$^0$=F$(x_0, \xi(x_0))$, $x_{cur} = 1$
$H = \begin{bmatrix} -1 & -1 & -1 & 0 & 1 & 1 & 1 & 0 \\ -1 & 0 & 1 & 1 & 1 & 0 & -1 & -1 \end{bmatrix}$
for m=1:8 do
    E$^m$=F$(H(:,m) * step + x_0, \xi(x_0))$
end for
for n=1:num_steps do
    while E$^{(2n-1)^2}$ : E$^{(2n-1)^2+8n-1}$ do not exist do
        continue
    end while
    j=1
    for i=0:8$n$ do
        if E$^{(2n-1)^2+i}$ converged then
            last_converged=$(2n-1)^2 + i$
        end if
        if $i = 0$ mod$(2n)$ then
            if j=1 then
                for k=1:2 do
                    Calculate E$^{x\_cur} = F(n * step * H(:,j) + x_0, \xi(\text{last\_converged}))$
                    j=j+1, x_cur=x_cur+1
                end for
            else
                for k=1:3 do
                    Calculate E$^{x\_cur} = F(n * step * H(:,j) + x_0, \xi(\text{last\_converged}))$
                    j=j+1, x_cur=x_cur+1
                end for
            end if
        else
            Calculate E$^{x\_cur} = F(n * step * H(:,j) + x_0, \xi(\text{last\_converged}))$
            j=j+1, x_cur=x_cur+1
        end if
    end for
end for
```

remain approximately 180° from one another. The angle specifying the rotation of the hydrogen need not be a dependent variable, but by pre-processing the initial iterate to maintain the symmetry within the molecule we may improve our initial iterates. Specifically in Figure 4.7 we pre-process the torsion angles 5-3-1-19 and 7-4-2-28 to be D1-180°. This continuation combined with pre-processing produced a PES that converges at every gridpoint and is continuous.

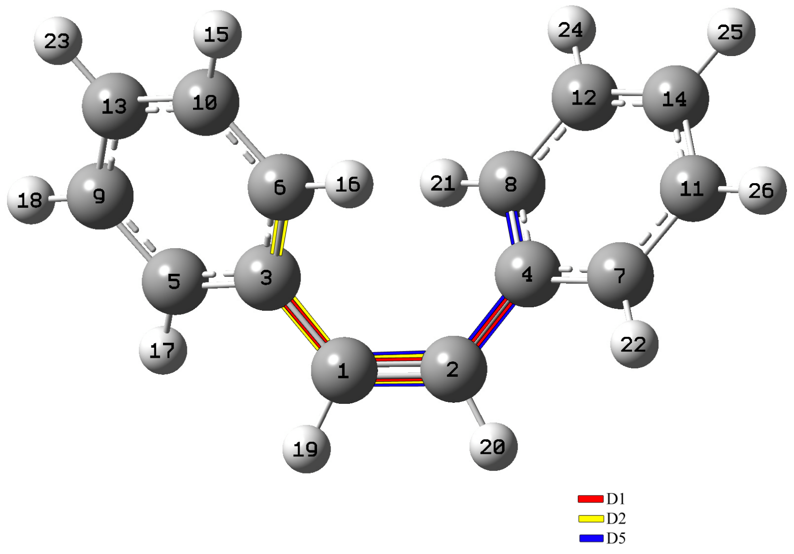


Figure 4.5: Stilbene molecule

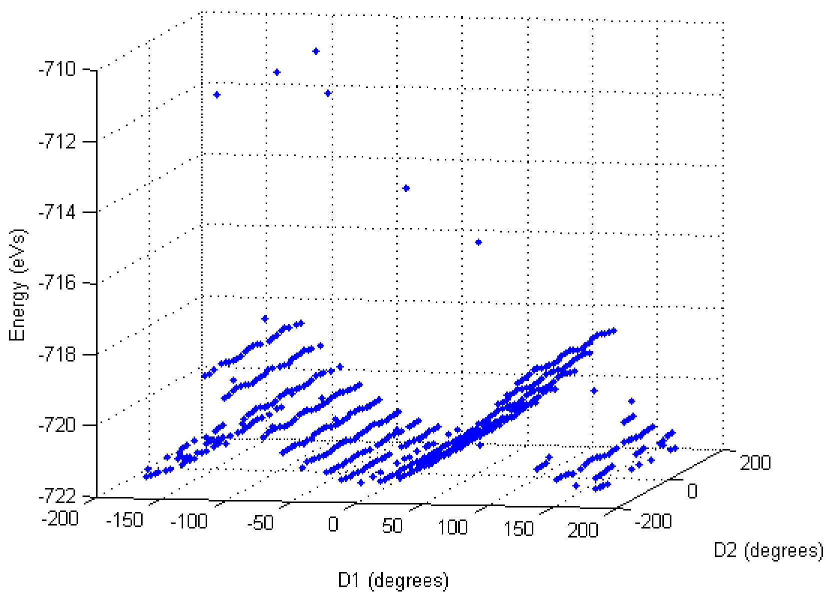


Figure 4.6: Butene surface generated by expanding perimeter without pre-processing

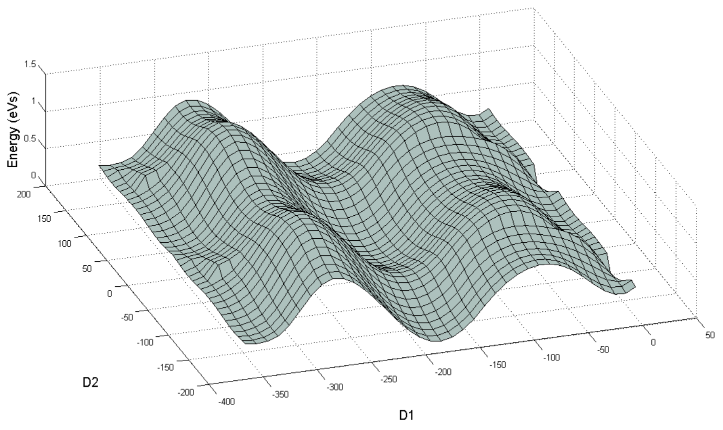


Figure 4.7: Stilbene surface generated by expanding perimeter algorithm and pre-processing

4.6 Ray Generation

Expanding perimeter successfully draws the PES, but the method of concentric squares once again leaves processors idle unnecessarily since the gridpoints in the next round are not submitted until every point in the previous round has finished. By changing the file dependence, this problem can be alleviated. Instead of filling the grid by drawing squares around the initial iterate, we draw rays with endpoint at the original gridpoint, x_0. Each gridpoint along the ray serves as the initial iterate for the gridpoint following it. This means that eight rays originate from the initial gridpoint. Each new point on the diagonal rays begin both a new horizontal and a new vertical ray. This entire set of rays can be generated ahead of time, and each time the energy is successfully optimized at a gridpoint the converged geometry can be used for the next gridpoint on the ray without awaiting other computations to return. In the event that a computation does not converge, the last converged geometry in the ray is used for the gridpoint succeeding it. Not only will this method expand the grid more quickly to maintain use of all available processors, but the generalization into larger dimensions is much simpler.

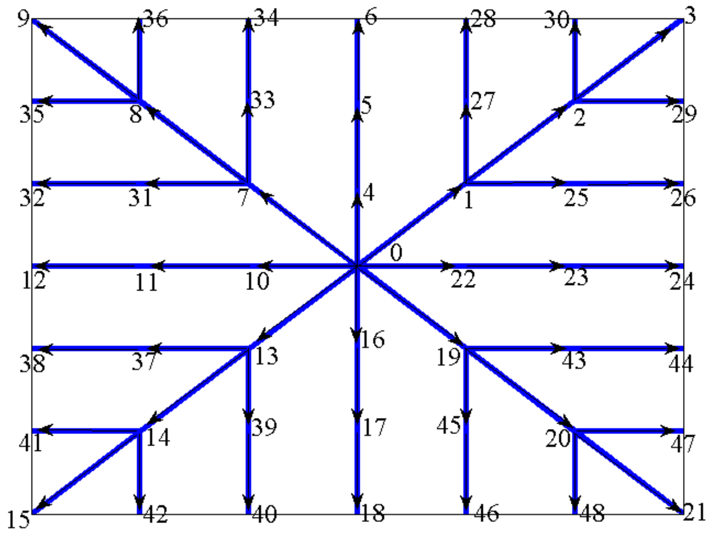


Figure 4.8: 2-Dimensional ray generation file dependence

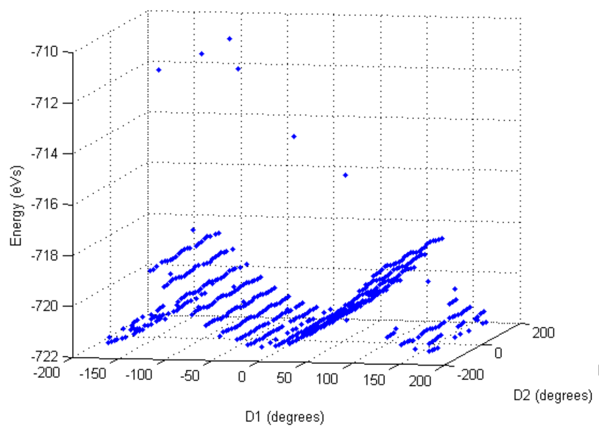
The ray generation algorithm in 2 dimensions is:

Algorithm 4.6.1 Ray Generation

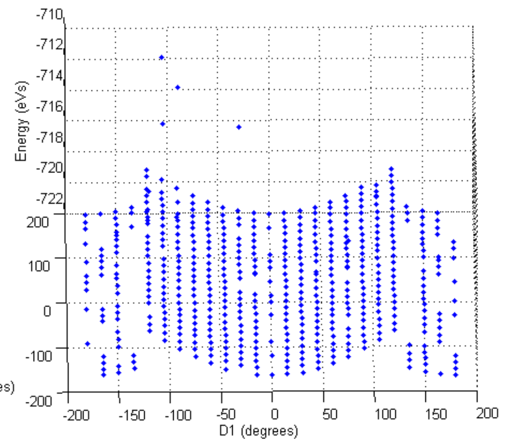
```
input $x_0$, $\xi(x_0)$, step, bounds
$cur\_file = 1$
$H = \begin{bmatrix} 1 & 0 & -1 & -1 & -1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & -1 & -1 & -1 & 0 \end{bmatrix}$
for m=1:8 do
    n=1
    $Ray_m = [x0', H(:,m)', 0]$
    while $H(:,m) * n * step + x_0$ =bounds do
        $Ray_m = [Ray_m, cur\_file]$
        $cur_file = cur\_file + 1$
        $n = n + 1$
    end while
end for
$m = m + 1$
for k=1:2:7 do
    for $j = length(Ray_k) - 6$ do
        $Ray_m = [x0' + j * H(:,k)', H(1,k), 0, Ray_k(j+5)]$
        $n = 1$
        while $Ray_m(1) + n * step * H(1,k)$ =bounds do
            $Ray_m = [Ray_m, cur\_file]$
            $cur\_file = cur\_file + 1$
            $n = n + 1$
        end while
        $Ray_{m+1} = [x0' + j * H(:,k)', 0, H(2,k), Ray_k(j+5)]$
        $n = 1$
        while $Ray_{m+1}(1) + n * step * H(2,k)$ =bounds do
            $Ray_{m+1} = [Ray_{m+1}, cur\_file]$
            $cur\_file = cur\_file + 1$
            $n = n + 1$
        end while
        $m = m + 2$
    end for
  end for
  $completed\_files = []$
  Calculate $E^0$=F$(x_0, \xi(x_0))$
  for $i = 0 : cur\_file - 1$ do
    if ($E^i$ exists) and ($i \in completed\_files = False$) then
      $completed\_files = [completed\_files, i]$
      for j=1:m-1 do
        if $i \in Ray_j == True$ then
          $k = Ray_j(index(i))$
          $E^{Ray_j(k+1)} = F(Ray_j(1:2) + (k-4) * Ray_j(3:4), \xi(x_i))$
        end if
      end for
    end if
  end for
```

We applied the ray generation algorithm to 2-butene without pre-processing the angles [35]. The results in figure 4.9 imply that pre-processing the angles in the same manner as we did with expanding perimeter is necessary to prevent jump discontinuities. Figure 4.10 was generated by pre-processing 2-butene in the same fashion that we pre-processed Stilbene in Figure 4.7, and the results are a continuous surface that matches our expectation. The ray generation algorithm also exhibits good weak scalability meaning that doubling the size of the problem (number of computations) along with doubling the number of processors does not have an effect on the overall compute time.

We performed a scalability study of the ray generation algorithm. All computations were performed on the high performance computing cluster at North Carolina State University. Our chassis has 60 quad core Xeon processors with 2GB distributed memory per core and dual gigabit ethernet interconnects. The operating system is Red Hat Linux 2.6.9 Potential energy computations are performed using Gaussian 03. Script editing is done with Python 2.5.4. The results of this scalability study are shown in table 4.1.

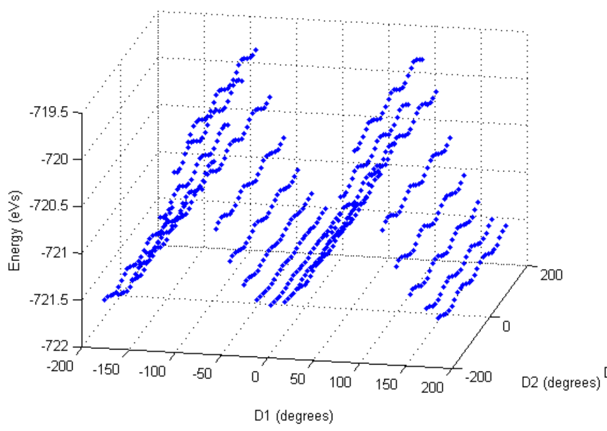


(a)

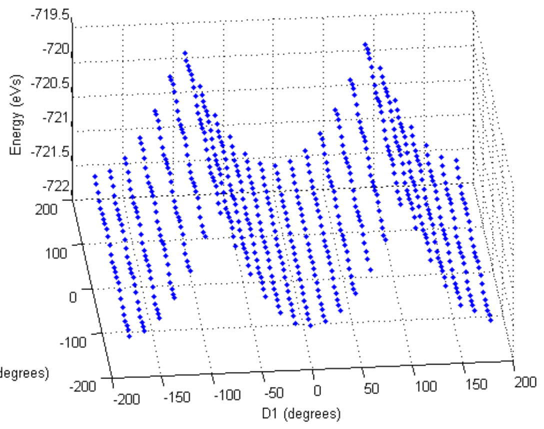


(b)

Figure 4.9: Two views of PES for 2-butene computed using ray generation without pre-processing



(a)



(b)

Figure 4.10: Two views of PES for 2-butene computed with ray generation with pre-processing

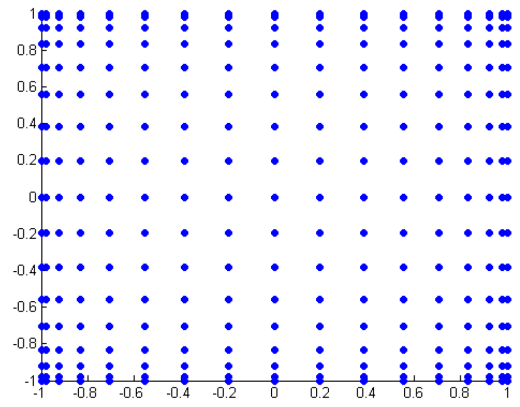
Table 4.1: Scalability

Grid Size	Processors	Time (secs.)
17x17	12	525
25x25	24	451
37x37	48	570

4.7 Sparse Interpolation

The gridpoints generated by the Smolyak algorithm (see Section 3.3.3) are not distributed in a way that makes it possible to use either ray generation or expanding perimeter to construct the PES. Since we use this method to compute only small patches of the full PES (see Section 5.2.3) we do not need to use continuation to prevent discontinuities. Instead we manage large discontinuities, slow convergence, and failure to converge by shrinking the size of the patch.

Both the expanding perimeter and the ray generation algorithms are continuation algorithms. The Smolyak algorithm uses the same initial iterate for every gridpoint. Since we are evaluating only portions of the surface in an incremental fashion, we will be using a new initial iterate for each new patch, so each patch is a continuation of the previous. Figure 4.12 is a full PES for 2-Butene computed using only the sparse gridpoints and some pre-processing. The advantage of sparse interpolation is that our PES becomes feasibly computable with degrees of freedom far larger than 2. Smolyak grids grow polynomially in size as the dimension increases, whereas the grids for the splines we had used earlier were square and thus grew exponentially in size. Figure 4.11 compares the gridpoints from a tensor grid to Smolyak's grid produced from the same one-dimensional nodes.


(a) Gridpoints resulting from Tensor of Chebyshev Nodes

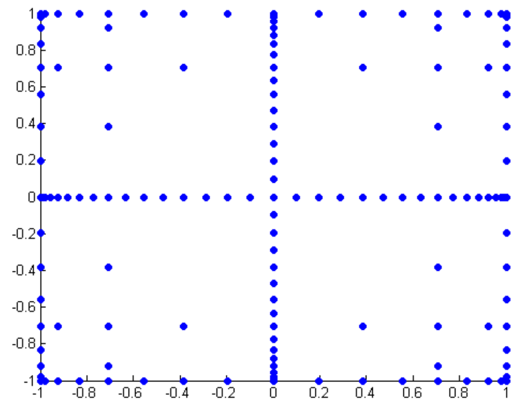
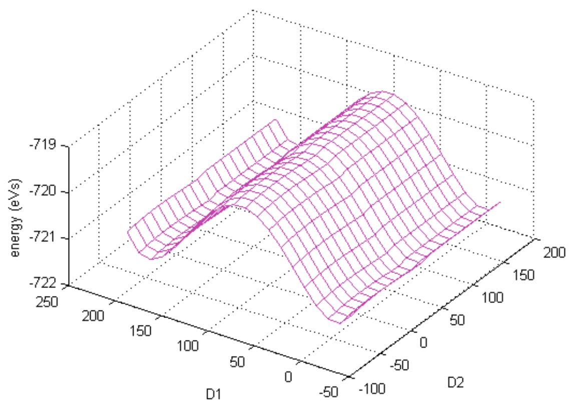

(b) Sparse Gridpoints resulting from the same Chebyshev Nodes

Figure 4.11: Tensor grid vs. Sparse grid


(a)

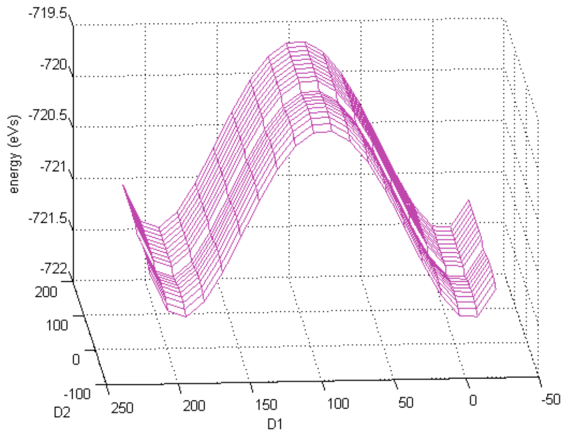

(b)

Figure 4.12: Two views of PES for 2-butene computed on a sparse grid from a single iterate

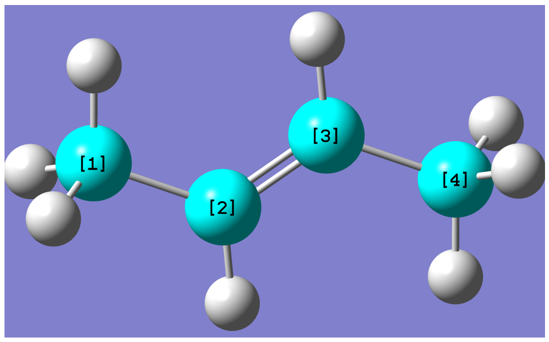
Chapter 5

Simulation

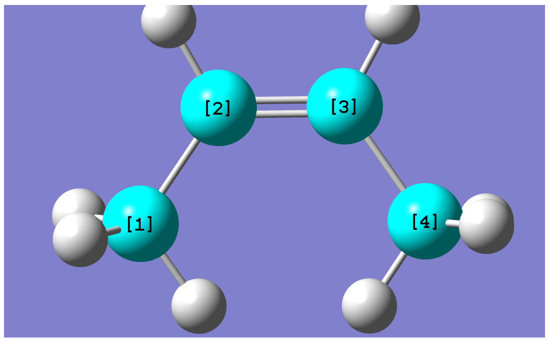
5.1 Single Degree of Freedom

We use 2-butene as a demonstration molecule because it has a known transition path [38, 55, 29]. We reproduce the results from [29] using Gaussian. Since we are performing the simulation with a single degree of freedom, it is easiest to compute the entire potential energy curve and then interpolate it using a standard cubic spline (Section 3.2). Here we use the default spline in Matlab which is a cubic spline with continuous 3rd derivative. Starting from a local minimum geometry in the ground state we excite the molecule from the ground state, $E_0(x_0)$ to a preselected excited state, $E_n(x_0)$. Excitation creates an entirely new energy function, and the current point is no longer a local minimum. To find a local minimum on the current energy level, we follow the gradient descent direction. Once we have reached the local minimum on the current state, the molecule then is excited or emits energy and changes energy levels again. Once we return to the ground state and relax for the last time, we check if the new value of the coordinate differs from the initial value. When the final geometry differs from the initial geometry, the simulation has successfully discovered a path between stable geometries.

Figure 5.1 shows the two stable geometries for 2-butene. It is clear that the largest difference between the two conformations is the rotation of the double bond in the center of the molecule. Figure 5.2 depicts the entire simulation for 2-Butene with this single coordinate as the only independent variable.



(a) Trans 2-Butene



(b) Cis 2-Butene

Figure 5.1: Butene molecule, C_4H_8

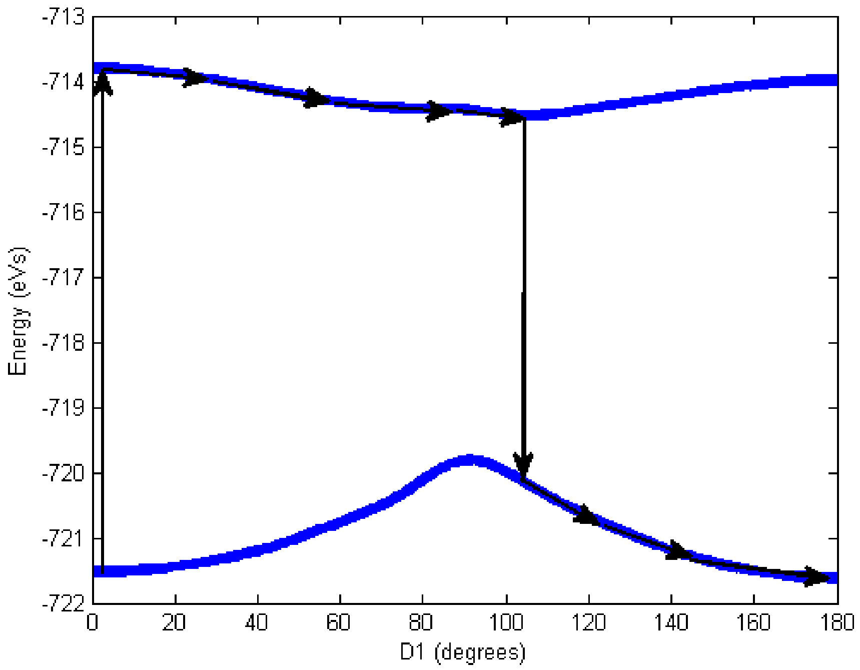


Figure 5.2: 2-Butene transition path in a single degree of freedom

5.2 Two Degrees of Freedom

5.2.1 Integration

Our simulations are meant to successfully predict the natural relaxation of a molecule. This means we do not want to take large steps when integrating on the surface since they may pass over a local minimum. We use continuous steepest descent [8, 28] to simulate this relaxation. This method uses an ODE solver (in our case a Runge-Kutta 45 method [1]) to integrate

$$\dot{x} = -\nabla f(x). \tag{5.1}$$

In our case, the independent variables are the design variables, x, and $f(x)$ is the surrogate model for the energy, $E_n^s(x)$. We have used both a finite difference and an analytic gradient for this integration. Gaussian is able to compute an analytic gradient, but calls to Gaussian are quite expensive and you have to compute the gradient of $\xi(x)$. A robust integrator, like ode45, has stepsize management that ensures local minima are not missed by the integration path.

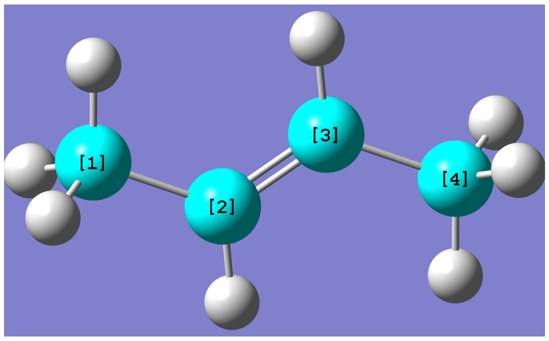
5.2.2 Full Surface Simulation

Since we had a test molecule with a known excitation path we could expand our simulations to higher dimension and have confidence in the validity of our simulation's results if they verify the known result of excitation. Using the expanding perimeter algorithm, we computed the entire PES for 2-butene by adding a second coordinate to the computation. This coordinate can be seen in figure 5.3 (c). Once all of the gridpoints had successfully converged, we could interpolate the PES with Matlab's cubic spline. In two dimensions Matlab's cubic spline requires a square grid. With a complete surrogate for the surfaces, we may excite the molecule from any point on the ground surface, $E_0^s(x) \to E_i^s(x)$. Next,the molecule relaxes to a local minimum in that state and transfers to a new surface. This process continues until it reaches a local minimum on the ground state. The sequence of the excited states is determined at the beginning of the simulation. The full surface simulation algorithm is

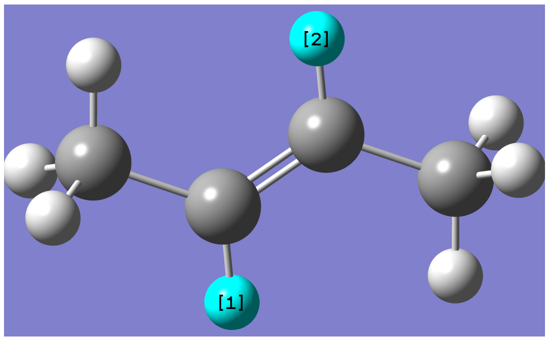
A successful optimization on the full surface can be seen in figure 5.4. In this simulation we computed the surface using the expanding perimeter algorithm.

Algorithm 5.1 Full Surface Simulation Algorithm

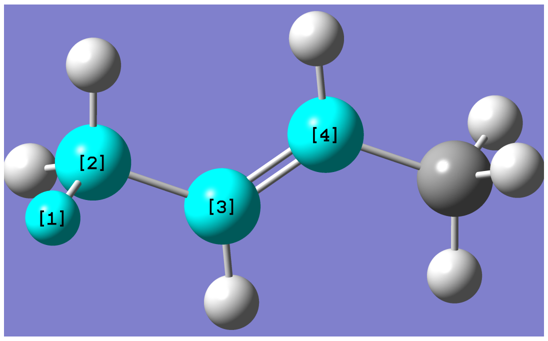
```
input states, $x_{cur}$
for i in states do
    $x_{cur} = ode\_solver(\dot{x}_{cur} = -\nabla E_i^s(x_{cur})$
end for
```



(a) Coordinate to be rotated consists of the bond angle between atoms labeled 2 and 3



(b) Atoms 1 and 2 whose locations are preprocessed for better convergence of gaussian optimizations



(c) 2nd coordinate to be rotated consists of the bond angle between atoms labeled 2 and 3

Figure 5.3: 2-butene torsion angles selected for a 2 dimensional simulation

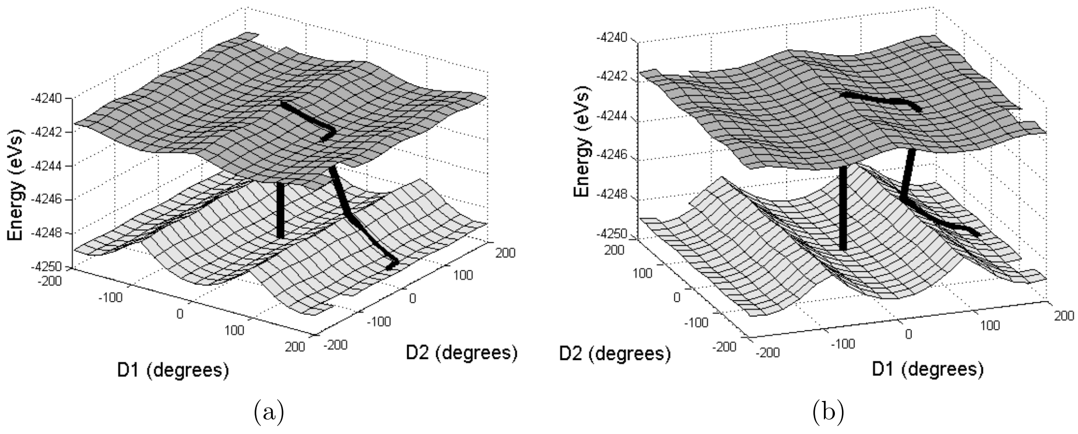


Figure 5.4: Successful transition path for 2-butene simulated on a full PES computed with expanding perimeter

5.2.3 Incremental Surfaces

Simulations run on a fully computed surface are successful, but they have 3 major drawbacks

- The process is very slow since it must compute energy at so many gridpoints.
- Resources are wasted computing areas of the surface never seen by the optimization.
- The cost of each energy computation grows exponentially with the number of atoms in the molecule.

For these reasons, we aimed to keep the number of calls to Gaussian to a minimum. We decreased the number of calls to Gaussian by computing small patches of the PES and integrating either to a local minimum or to a boundary on those patches. The integration is performed using ODE45 in Matlab [53]. Once we have exhausted the search on the current patch we draw a new patch. The surrogate for the surface on the patch is still interpolated with matlab's default cubic spline.

5.2.4 The Next Patch

Each of our patches is square of length h. Obviously an integration step will never collide directly with a boundary of the domain. In the case where the minimum does not lie within the domain, we terminate the integration when a step, y_i, falls within some distance of the interior boundary Ω

$$\|\Omega - y_i\|_1 \leq \delta, \quad y_i \in \Omega. \tag{5.2}$$

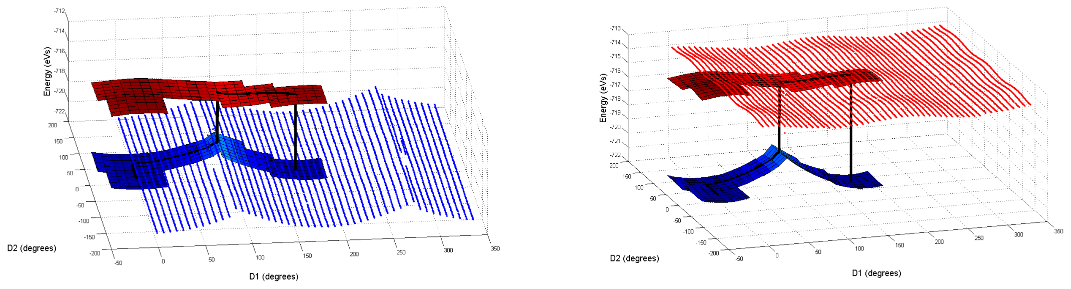
Normally the value of δ is chosen to be $.05h$. If the integration step exceeds the boundary, then we reject the step and shrink the stepsize similar to the case when the error is too large (see Appendix A).

Upon termination at the boundary of a patch we have to draw a new patch. We expect the integration to continue in the direction it was going when it terminated inside the previous patch, so it would be redundant to center the new patch at that terminal point. Instead we shift the center of the patch so that any variable that was within δ of the previous boundary lies at $\hat{\delta}$ from the previous boundary with which it collided. The normal choice of $\hat{\delta}$ is $.2h$. For example, if the last integration terminated at the point $y_i = (-.99, .5)$ and the domain had been $[-1, 1]^2$, then we would center the next patch at $(-.99, .5) + (-1, 0) * .6 = (-1.59, .5)$. The vector (-1,0) signifies that the terminal point was near the lower boundary of the first variable.

We also have to deal with the termination of a patch at a minimum. In this case the algorithm will change states, but we already own a surface in the next state in the same domain as the current patch due to the method for computing excited states. Whenever possible we would prefer to continue integrating without having to perform new energy evaluations and we address how we determine whether it is prudent to do so in Section 6.2.

5.2.5 2-butene Example

Figure 5.5 displays the outcome of the simulation performed on incrementally constructed surfaces. Each surface was drawn using the ray generation algorithm from Section 4.6. The results were identical to the path found on the full surface, but the compute time decreased. The entire simulation including computing a full PES in figure 5.4 lasted 3275.66 seconds while the incremental surfaces required only 1404.93 seconds. The speedup is limited by the ray generation algorithm because the algorithm initially calculates the energy at only a few points.



(a) Entire ground state PES overlaid with the patches computed in the simulation

(b) Entire excited state PES overlaid with the patches computed in the simulation

Figure 5.5: Incremental surface simulation of 2-butene transition path

5.3 More than Two Degrees of Freedom

The number of gridpoints necessary if we use surrogates that are splines on a square grid grows exponentially with the number of degrees of freedom. This would render our problem intractable for high dimensional simulations. The Smolyak algorithm gives us similar accuracy to a square or a full tensor grid. We showed in Section 3.4 that the growth in the number of gridpoints is polynomial in the degrees of freedom[61]

$$\dim(A(d+k,d)) \approx \frac{2^k}{k!} d^k. \tag{5.3}$$

We have no need to alter the methods of transitioning between patches to adapt to our new interpolation scheme so the transitions are the same as in Section 5.2.4. Combining the incremental surface construction with a sparse interpolation allowed us to perform rapid and efficient simulations in three or more degrees of freedom. We decided to name the multi-dimensional sparse version of the software LITES for Light-Induced Transition Effects Simulator.

5.3.1 Termination

Once all of the states for the simulation have been exhausted and the integration has terminated at a local minimizer of the surrgoate at the ground state, we would like to find the actual local minimizer of the energy function. This is easily handled by Gaussian since we are already nearby as we discussed in Section 4.5. Thus final point in the entire simulation will be calculated by an unconstrained geometry optimization with the initial iterate being the terminal point, $(x_{f_{sim}}, \xi_{f_{sim}})$, of the integration on the final patch

$$(x_f, \xi_f) = min_p E_0(x_{f_{sim}}, \xi_{f_{sim}}). \tag{5.4}$$

The full LITES algorithm will be

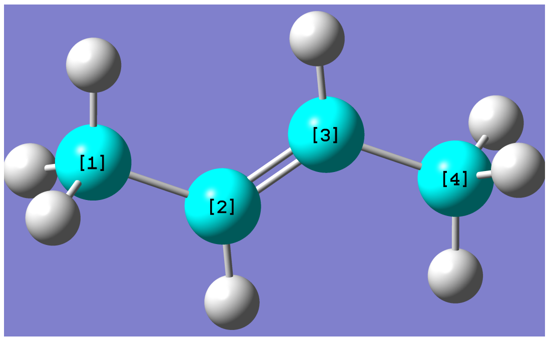
Algorithm 5.2 LITES Algorithm

```
**input** $h_0$, state_order, $x_0$, $k$
$h = h_0$
**for** state in state_order **do**
    edge = zeros(length($x0$))
    at_min = False
    **while** at_min = False **do**
        from $x0$, $h$, edge generate Smolyak grid
        Calculate $E_{state}(x)$ at each gridpoint in parallel and count maximum iterations
        **while** $E_{state}(x_i)$ fails **do**
            $h \to \frac{h}{2}$
            from $x0$, $h$, edge generate Smolyak grid
            Calculate $E_{state}(x)$ at each gridpoint in parallel and count maximum iterations
        **end while**
        integrate $\dot{x} = -\nabla E^S_{state}(x)$ using RK45, return at_min, $x_f$, edge
        $h$ =Patch_Control($x_f$, $h$,edge)
        $x_0 = x_f$
    **end while**
**end for**
Optimize $E_0$ with initial iterate $x_f$.
```

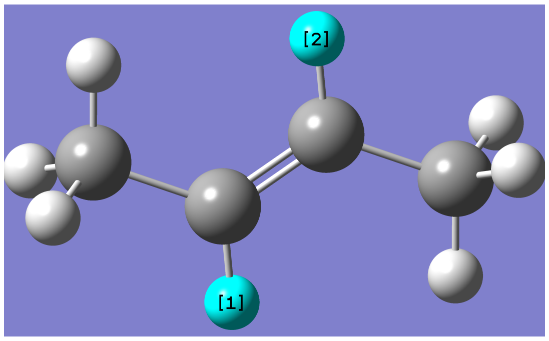
5.3.2 Example

We added a third angle to the simulation of 2-butene as shown in figure 5.6. Since we are no longer drawing full PES's, we no longer need the continuation methods from Chapter 4. Instead we will use a single initial iterate on each patch. The patches themselves serve as a continuation of the previous patch with a new initial iterate on each one. In the next chapter we develop methods of error approximation and control for the progression of patches.

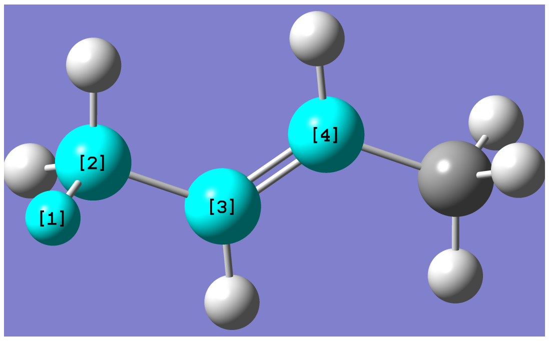
Applying the incremental sparse surface construction algorithm to 2-butene again yielded a successful path. The patches in Figure 5.7 grow in size throughout the simulation. The error control method was borrowed from trust region methods [23] and we discuss it in detail in Section 6.1. We expect these simulation methods to exhibit good weak scalability regardless of the error control technique and we demonstrate that they do in table 5.1.



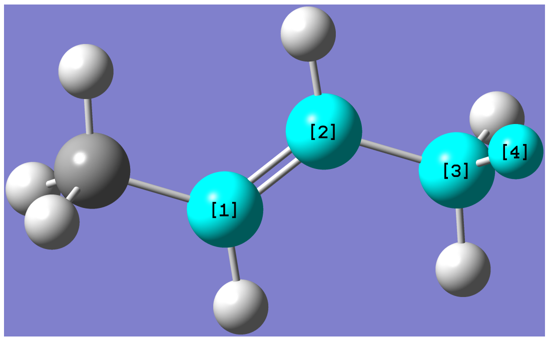
(a) Coordinate to be rotated consists of the bond angle between atoms labeled 2 and 3



(b) Atoms 1 and 2 whose locations are pre-processed for better convergence of gaussian optimizations

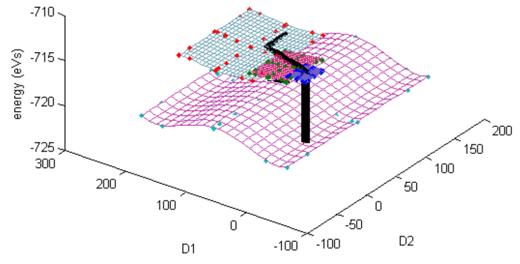


(c) 2nd coordinate to be rotated consists of the bond angle between atoms labeled 2 and 3

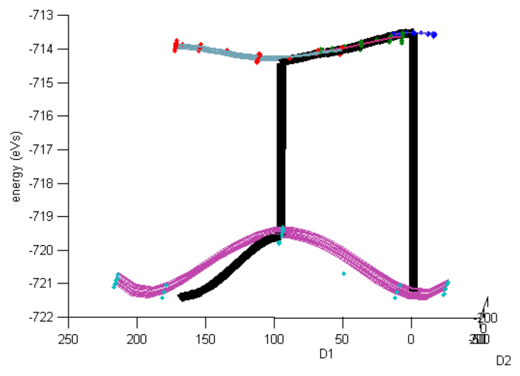


(d) 3nd coordinate to be rotated consists of the bond angle between atoms labeled 2 and 3

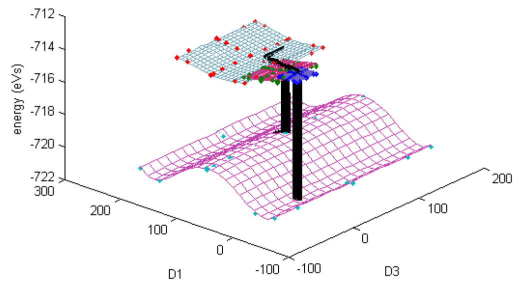
Figure 5.6: 2-butene torsion angles selected for a 3 dimensional simulation



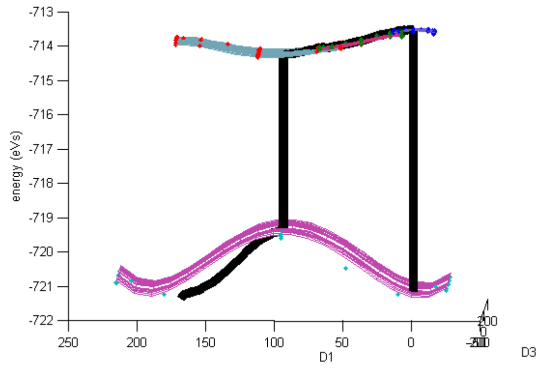
(a) First view of path in D1 D2 space



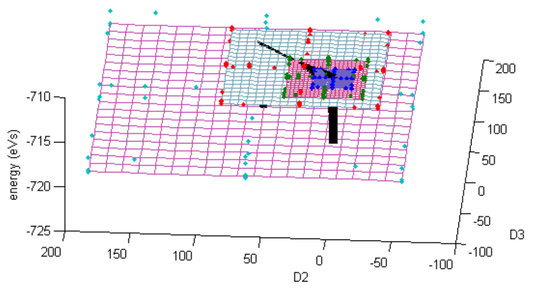
(b) Second view of path in D1 D2 space



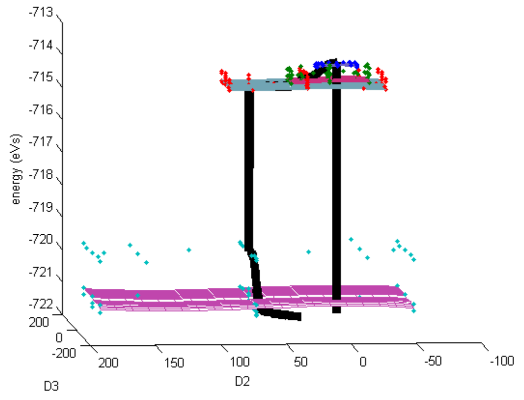
(c) First view of path in D1 D3 space



(d) Second view of path in D1 D3 space



(e) First view of path in D2 D3 space



(f) Second view of path in D2 D3 space

Figure 5.7: 3-D simulation of the transition path for 2-butene

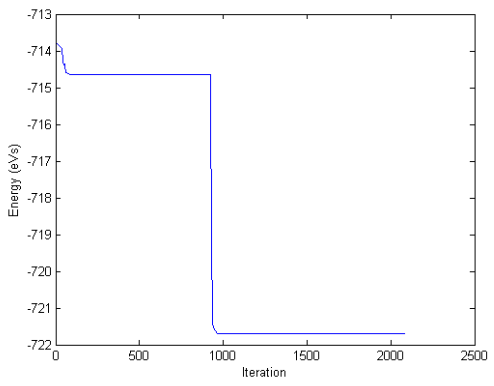


Figure 5.8: Plot of the energy path taken by the 3-D simulation

Table 5.1: Scalability of Sparse Incremental Surface Construction

degree of interpolation	Gridpoints	processors	Time (secs.)
2	25	13	190.96
3	69	39	194.05

Chapter 6

Patch Control

6.1 Trust Region Approach

We borrow ideas from trust-region algorithms [23] to adapt the size of the incremental surfaces. Each patch is the same length in all variables and we use the parameter h to be half of that length, similar to the radius of a trust region. At the conclusion of each patch's computation and subsequent integration we have 3 metrics which determine whether the size of the grid is sufficient:

- Have all of the gridpoints converged?
- The maximum number of internal optimizations for convergence among all of the points
- The ratio of the actual reduction in the energy to the predicted reduction in the energy: $\frac{ared}{pred}$

In the first case, if all the gridpoints do not converge then we cannot construct the surrogate model and thus we must shrink h and start again. Similarly if the number of internal optimizations at each gridpoint gets to be too large it likely signifies that the initial iterate, $\xi(x_0)$, is not a good one for all of the gridpoints and we will shrink h on the next patch. The variable *ared* is the actual reduction in the potential energy of the molecule at the endpoint of the integration on the patch

$$ared = E_n(x_0) - E_n(x_f). \tag{6.1}$$

The variable *pred* is the amount our surrogate model predicted the potential energy would decrease at the endpoint of the integration on the patch

$$pred = E_n(x_0) - \hat{E}_n(x_f). \tag{6.2}$$

If the value of the ratio $\frac{ared}{pred}$ is within the parameter ρ_1 of 1, then our model is very accurate and we may grow size of the patch, h. If that ratio is outside the parameter ρ_2 of 1, then the surrogate is not accurate and we decrease the parameter, h. In all other cases we do not change the size of the next patch. The Trust Region style error control algorithm is

Algorithm 6.1 Trust Region Approach Algorithm

```
input $\sigma$, $h_0$, max_iters, $\rho_1, \rho_2, x_{cur}, x_f$
calculate E($x_f$)
ared=E($x_{cur}$)-E($x_f$)
pred=E($x_{cur}$)-$\hat{E}$($x_f$)
if ($\|1 - |\frac{ared}{pred}|\| \leq \rho_1$) and (max_it≤max_iters) and (NaNs=0) then
   h=$\sigma$h
else if ($\|1 - |\frac{ared}{pred}|\| \geq \rho_2$) or (max_it>max_iters) or (NaNs≠ 0) then
   h=$\frac{h}{\sigma}$
end if
$x_{cur} = x_f$
return $x_{cur}, h$
```

6.1.1 Trust Region Approach Flow Chart

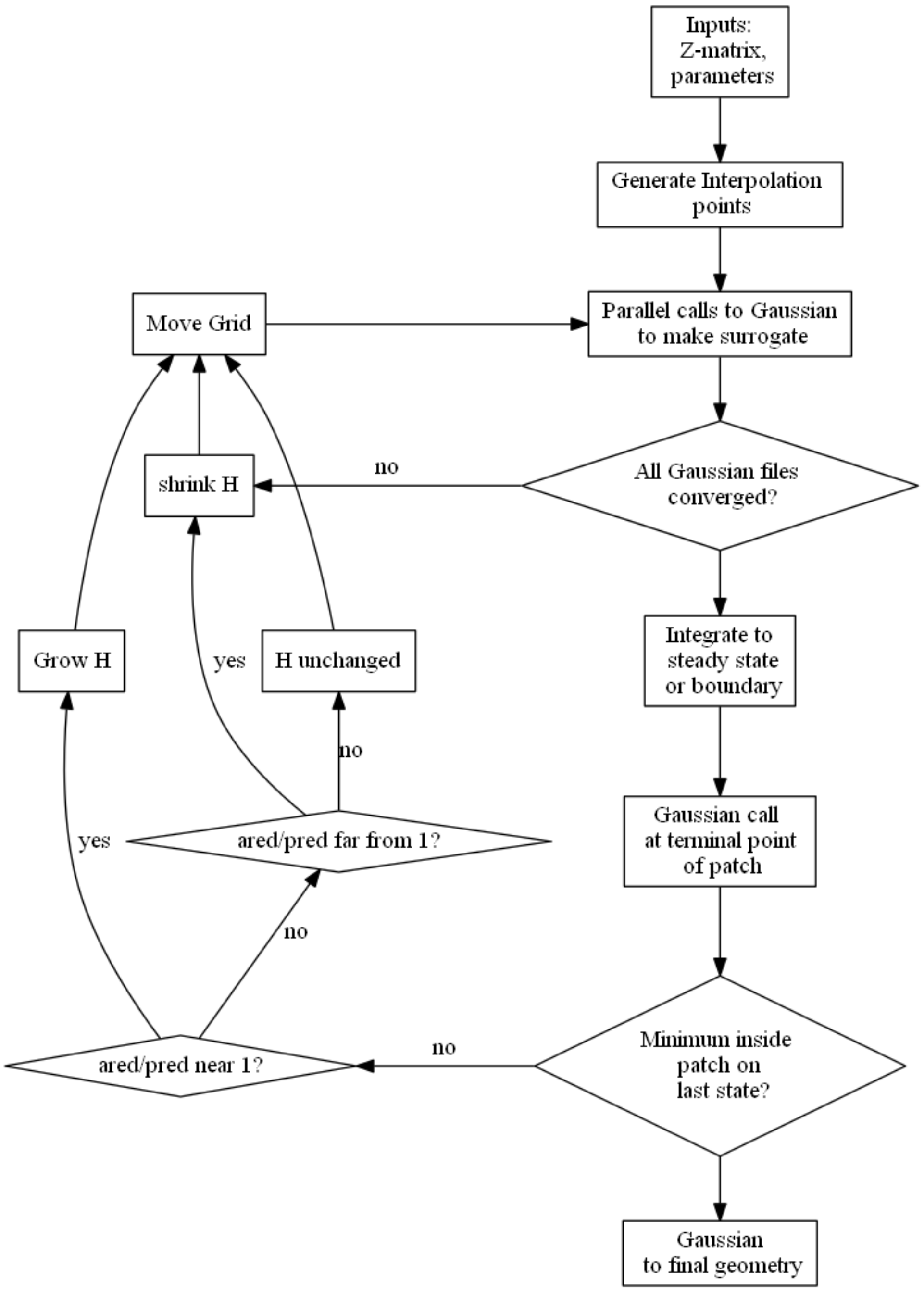

Inputs: Z-matrix, parameters
Generate Interpolation points
Move Grid
Parallel calls to Gaussian to make surrogate
no
shrink H
All Gaussian files converged?
Grow H
yes
H unchanged
Integrate to steady state or boundary
no
yes
ared/pred far from 1?
Gaussian call at terminal point of patch
no
no
ared/pred near 1?
Minimum inside patch on last state?
Gaussian to final geometry

Figure 6.1: Full flow chart for LITES with Trust Region approach

6.2 Runge-Kutta Approach

Runge-Kutta methods numerically solve ordinary differential equations (odes) by estimating the error of a time-step using a higher order approximation of the same time-step to give a good estimate of the error. The reason these methods are useful is that both approximations require the exact same function evaluations and so the error estimate is attainable with minimal work (see Appendix A). We can estimate the error on our patches using the same type of idea.

Smolyak's algorithm 3.3.3 using nested sets of Chebyshev extrema gives us properties that are similar to Runge-Kutta. Every approximation of order k has already evaluated the function at every gridpoint necessary for approximations of order less than k, thus lower order approximations are extremely inexpensive considering the time it requires for a function evaluation.

6.2.1 Example

Let us assume that we have obtained a cubic approximation for the energy function on a patch with the error given in Section 3.5.3

$$E_n^{S_3}(x) = E_n(x) + \mathcal{O}(h^4). \tag{6.3}$$

There is no expense (in function evaluations) of evaluating a quadratic approximation on the same patch

$$E_n^{S_2}(x) + \mathcal{O}(h^3). \tag{6.4}$$

Using these 2 approximations we may estimate the error at any point within the patch using the difference of the two approximations

$$\frac{\|E_n^{S_3}(x) - E_n^{S_2}(x)\|}{\|E_n^{S_2}(x)\|} = ch^3 = \epsilon. \tag{6.5}$$

Since the energy function is smooth this is a good approximation of the error.

Since we are dealing with two different approximations for the energy function, we have a choice of gradients for the integration on the patch. Although we use the quadratic approximation for the value of the energy, we will use the analytic gradient obtained from the cubic approximation for calculating the integration path. This method is called order extrapolation[52].

6.2.2 Patch Size

By having a good approximation of the error in our surrogates we can determine how large to make the next patch to ensure it remains accurate. The error estimate isn't the only metric we use for this decision. Each function evaluation is the result of an optimization as discussed in Section 2.6. We keep track of the maximum number of iterations it took for the function evaluations on the grid. If that number gets too high we shrink the patch irrespective of the error estimate. The function evaluations are extremely timely so our goal is to avoid ever having to restart a patch because of convergence failures within Gaussian.

Assuming that the iteration count remained below our tolerance we can derive a predictive formula for the next patch length. Let $\{X\}$ be the path of the simulation calculated as in section 5.2.1, then we take the error on the current patch to be

$$\epsilon_{cur} = \max_{x \in X} \frac{\|E_n^{S_{k+1}}(x) - E_n^{S_k}(x)\|}{\|E_n^{S_k}(x)\|}. \tag{6.6}$$

We calculate the size of the next grid using the common formula from actual Runge-Kutta methods. We specify a tolerance, δ, so ideally we have

$$\epsilon_{cur} = ch^{k+1} \leq \delta \tag{6.7}$$

$$c = \frac{\epsilon_{cur}}{h_{cur}^{k+1}}. \tag{6.8}$$

We would like our next patch to have a size, h_+, that is as large as possible while still keeping the error below our specified tolerance. Normally we choose the limit to be smaller than the chosen tolerance, $\rho\delta$ where $\rho < 1$

$$ch_+^{k+1} = \rho\delta < \delta \tag{6.9}$$

$$\epsilon_{cur} \left(\frac{h_+}{h_{cur}} \right)^{k+1} = \rho\delta. \tag{6.10}$$

$$h_+ = h_{cur} \left(\frac{\rho\delta}{\epsilon_{cur}} \right)^{\frac{1}{k+1}}. \tag{6.11}$$

In the case where $\epsilon > \delta$ we reject the patch because it is not accurate, but the formula will give us a smaller patch size at which to restart the calculations. If the error remains acceptable the formula will give us a new length for the next patch.

6.2.3 Example

We will continue with the prior example and choose a quadratic interpolation with a cubic error term, so $k = 2$. Let us choose $\rho = .8^3$, then we will have

$$h_+ = h_{cur} \left(\frac{.8^3 \delta}{\epsilon_{cur}} \right)^{\frac{1}{3}} = .8 h_{cur} \left(\frac{\delta}{\epsilon_{cur}} \right)^{\frac{1}{3}} \tag{6.12}$$

6.2.4 Termination Inside a Patch

When the integration terminates in the interior of a patch we will change states. We assume that the iteration tolerance has not been exceeded, because if it has we will shrink the patch. Using Formula 6.11 with the ϵ calculated on the integration path of the current state would be a mistake. When the state changes so does the function $E_n^S(x)$, but we would like to avoid having to perform more function evaluations. Since we are beginning the next integration on the interior of a domain where we already own a surrogate, there is no reason not to keep the current patch if it is accurate. This means we may use the Formula 6.11 if we can calculate an ϵ on the new state. We don't have an integration path, but since we own the surrogate we can cheaply query any points inside the domain.

For multi-dimensional functions a common sampling technique is to use a Latin Hypercube [57]. From the latin hypercube we get a set of points $\{\hat{X}\}$ just as we had a set of points defining a path on the previous state. We determine the ϵ in the new state the same way we used the path in the previous state so that

$$\epsilon_{cur} = \max_{x \in \hat{X}} \frac{\| E_n^{S_{k+1}}(x) - E_n^{S_k}(x) \|}{\| E_n^{S_k}(x) \|}. \tag{6.13}$$

If this error estimate is below our specified tolerance we may keep the patch and integrate to a steady-state or a boundary. If not we will have to construct a new patch of the size determined by Formula 6.11. The full Runge-Kutta approach error control algorithm is:

Algorithm 6.2 Runge-Kutta Approach Algorithm

```
input $\delta$, $h_{cur}$, max_iters, $\rho$, $X$, $k$, num_iters,at_min, $\Omega$, next_state
$\epsilon = \max_{x \in X} \frac{\|E_n^{S_{k+1}}(x) - E_n^{S_k}(x)\|}{\|E_n^{S_k}(x)\|}$
if $\epsilon > \delta$ then
    reject $X$
    $h_+ = h_{cur} \left(\frac{\rho\delta}{\epsilon}\right)^{\frac{1}{k+1}}$
else if num_iters $\geq$ max_iters then
    $h_+ = \frac{h_{cur}}{2}$
else if at_min=FALSE then
    $h_+ = h_{cur} \left(\frac{\rho\delta}{\epsilon}\right)^{\frac{1}{k+1}}$
else
    generate $\hat{X} \subset \Omega$
    $\epsilon = \max_{x \in \hat{X}} \frac{\|E_{next\_state}^{S_{k+1}}(x) - E_{next\_state}^{S_k}(x)\|}{\|E_{next\_state}^{S_k}(x)\|}$
    $h_+ = h_{cur} \left(\frac{\rho\delta}{\epsilon}\right)^{\frac{1}{k+1}}$
end if
return $h_+$
```

6.2.5 Runge-Kutta Approach Flow Chart

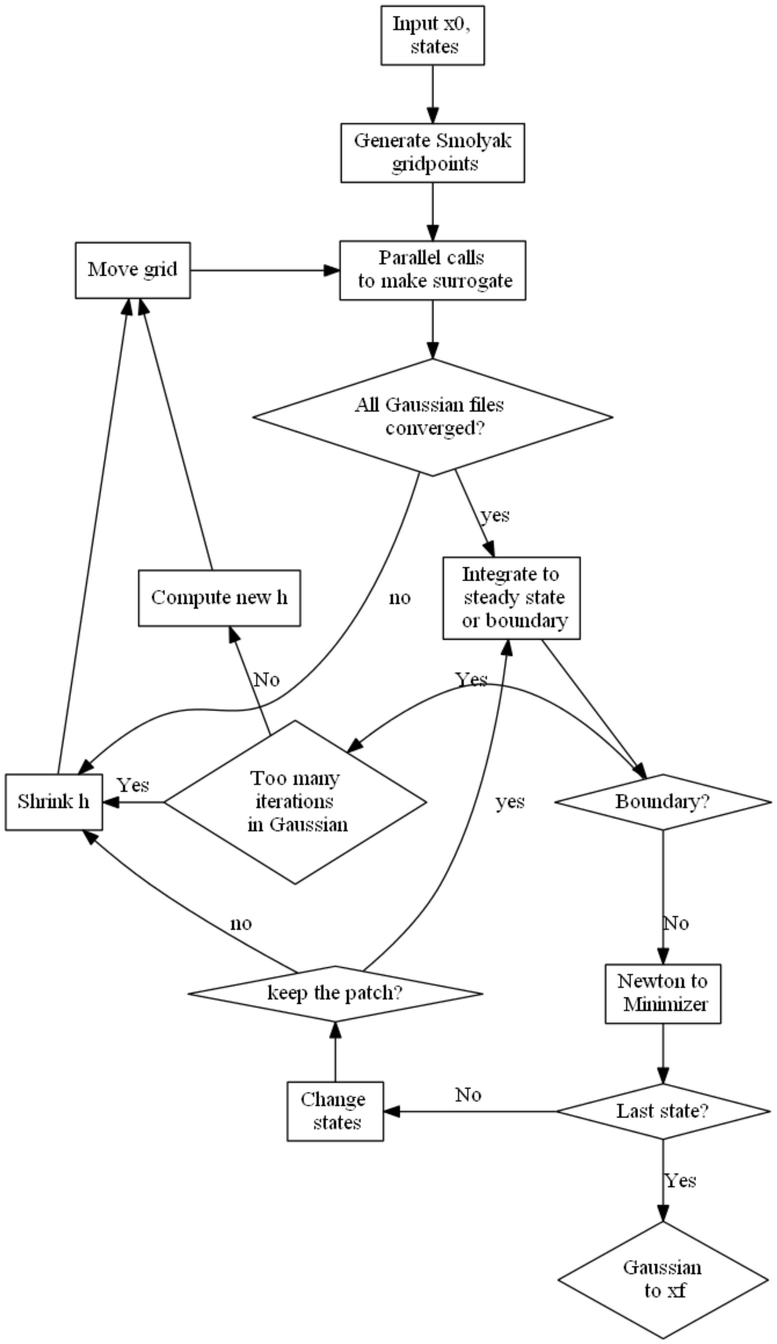

Input x0, states
Generate Smolyak gridpoints
Move grid
Parallel calls to make surrogate
All Gaussian files converged?
yes
no
Compute new h
Integrate to steady state or boundary
No
Yes
Shrink h
Yes
Too many iterations in Gaussian
yes
Boundary?
no
No
keep the patch?
Newton to Minimizer
Change states
No
Last state?
Yes
Gaussian to xf

Figure 6.2: Full flow chart for LITES with Runge-Kutta approach

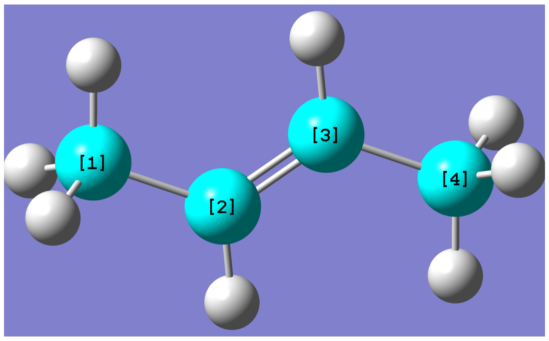
Chapter 7

Results

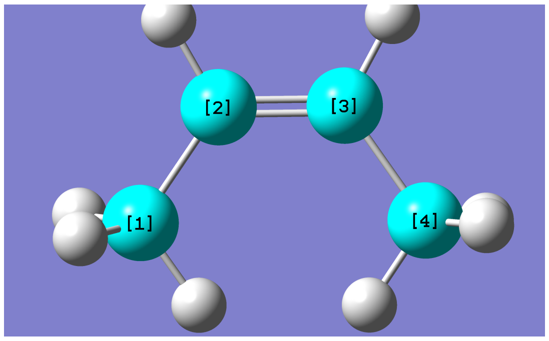
7.1 2-butene

2-butene serves as our test molecule because it has a well-known transition path between stable ground state geometries [29]. We have used the molecule to test every algorithm that we have developed, but here we will use it to show that the Runge-Kutta approach to error control is successful at managing the error. We begin in the cis configuration and the simulation will successfully terminate in the trans configuration as in Figure 7.1. We will use the same 3 degrees of freedom as in Figure 5.6, but we will display our results by only showing the individual changes in each variable at each integration step instead of showing surfaces.

For the Runge-Kutta approach we set the error tolerance of the simulation to be $\delta = 10^{-3}$ with a fudge factor $\rho = .7^3$. The higher order interpolation is cubic and thus the lower order is quadratic. In this example we will also calculate the actual error at each integration point to see how well our estimate is performing. This extra calculation is expensive since it doubles the simulation time and will not be done in simulation with larger molecules. Figure 7.2 displays the actual error on the integration path versus the approximation of the error on each of the 5 patches for the simulation. Neither error comes near the tolerated error of $\delta = 10^{-3}$. Figure 7.3 shows that the simulation successfully transitioned from cis to trans. This simulation required 1800.75 seconds. These results also appear in [33].



(a) Trans 2-Butene



(b) Cis 2-Butene

Figure 7.1: Butene molecule, C_4H_8

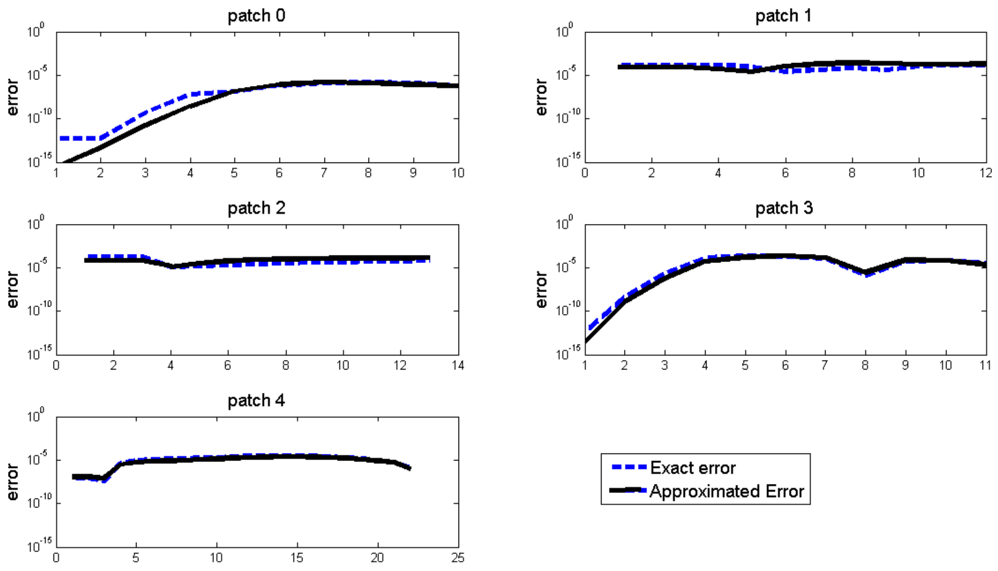


Figure 7.2: Actual vs. approximate error for 2-butene 3D simulation

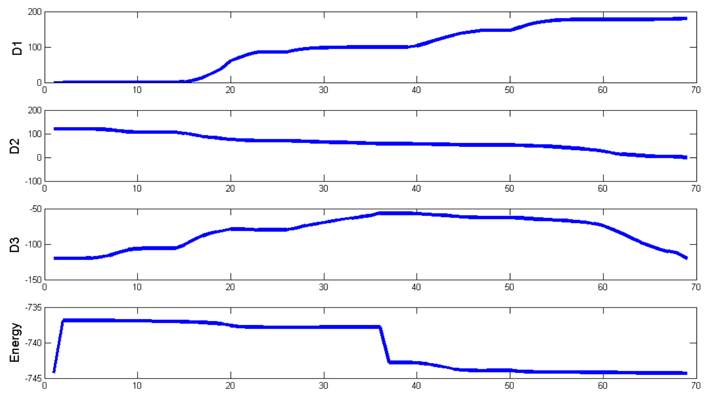
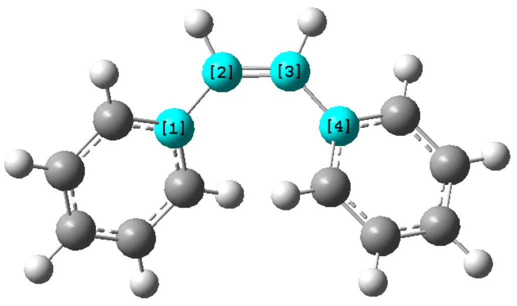


Figure 7.3: Simulation history for 2-butene with RK approach

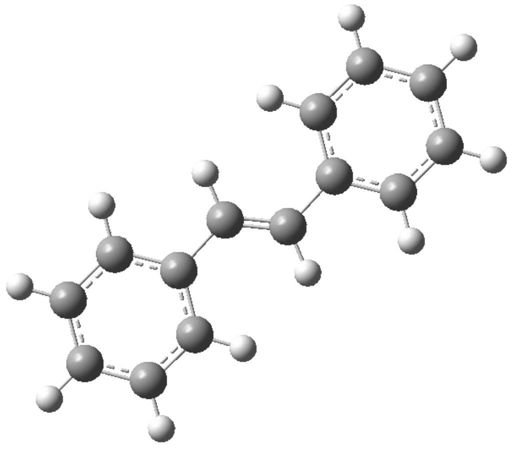
7.2 Stilbene

Stilbene is a molecule with two known stable conformations in the ground state. They are both displayed in Figure 7.4. We hope to use our simulation techniques to find an energy path through the excited states between these 2 geometries. Using just a single degree of freedom has not yielded a path [34] so this molecule is a perfect candidate for the algorithms which we developed.

We chose a simulation using 5 degrees of freedom. Each of those 5 coordinates are displayed in Figure 7.5. The simulation method was trust-region style (Section 6.1) and the Smolyak surrogates had quadratic exactness. The simulation was run in the following order of states, $2 \rightarrow 1 \rightarrow 0$. Simulation time was 95919.22 sec. A simulation history is displayed in Figure 7.6. The actual initial and final geometries of the simulation are displayed in Figure 7.7. The 2 geometries are clearly visually different, but they both seem to be cis-stilbene. While we are unsure whether there is chemical significance to this outcome, they indicate that the tool can find paths between local minima. Some discussion of these results appears in [6]

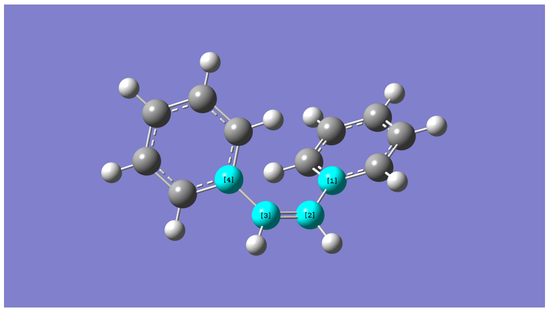


(a) cis-stilbene

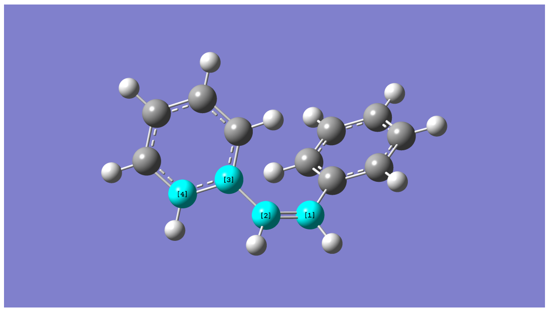


(b) trans-stilbene

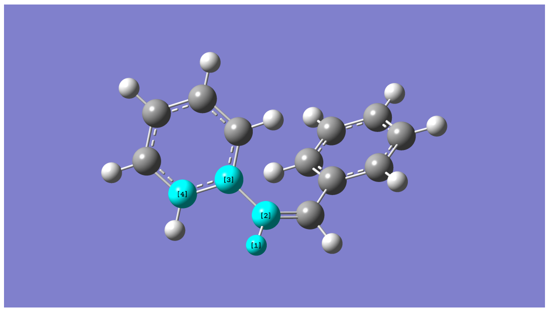
Figure 7.4: Both stable conformations of the molecule stilbene



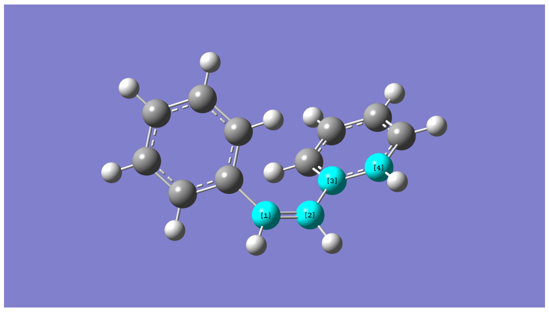
(a) D1



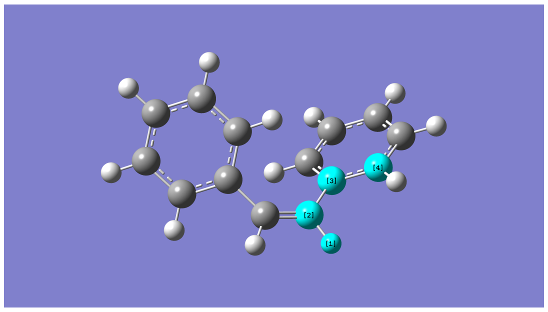
(b) D2



(c) D3



(d) D4



(e) D5

Figure 7.5: Each of the 5 coordinates in the stilbene simulation

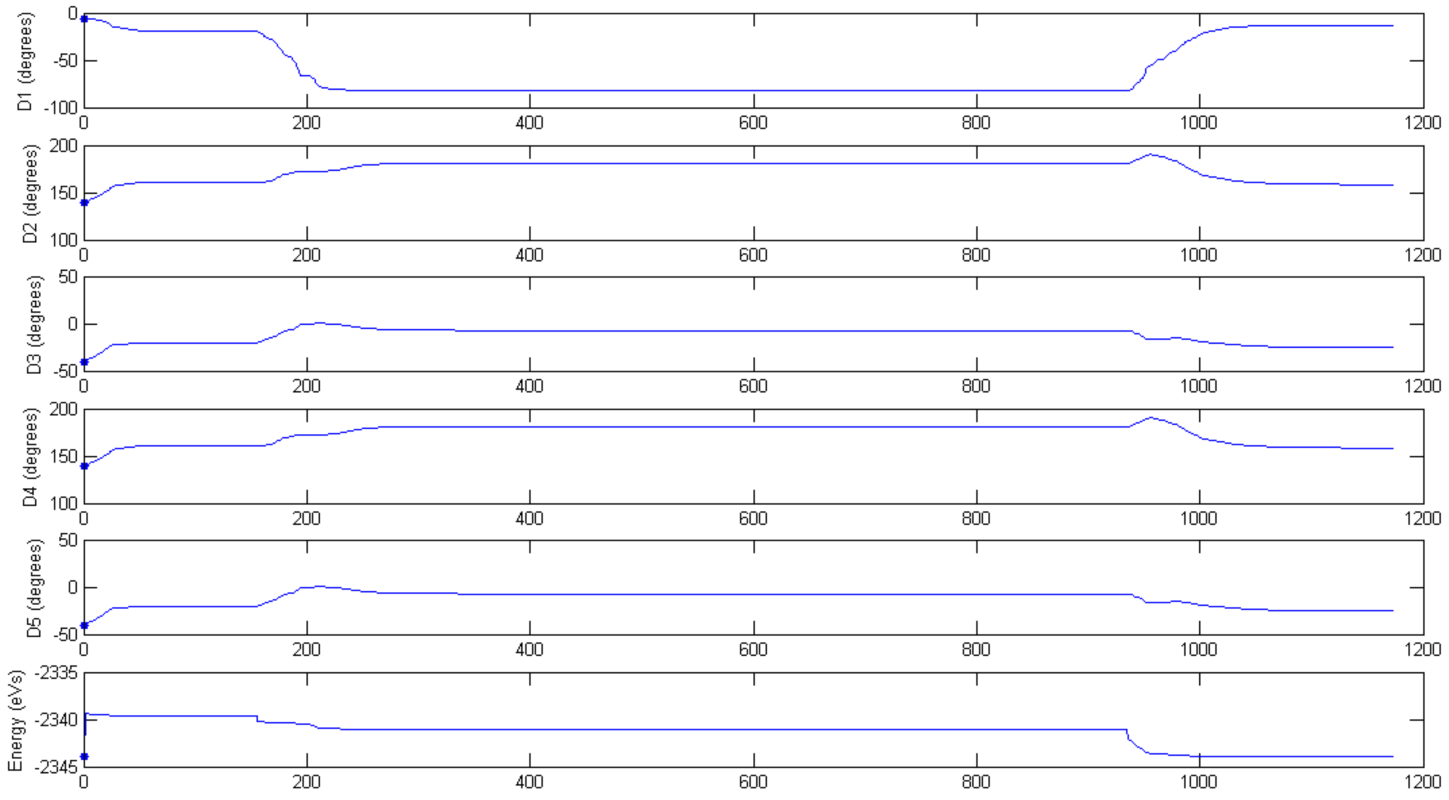
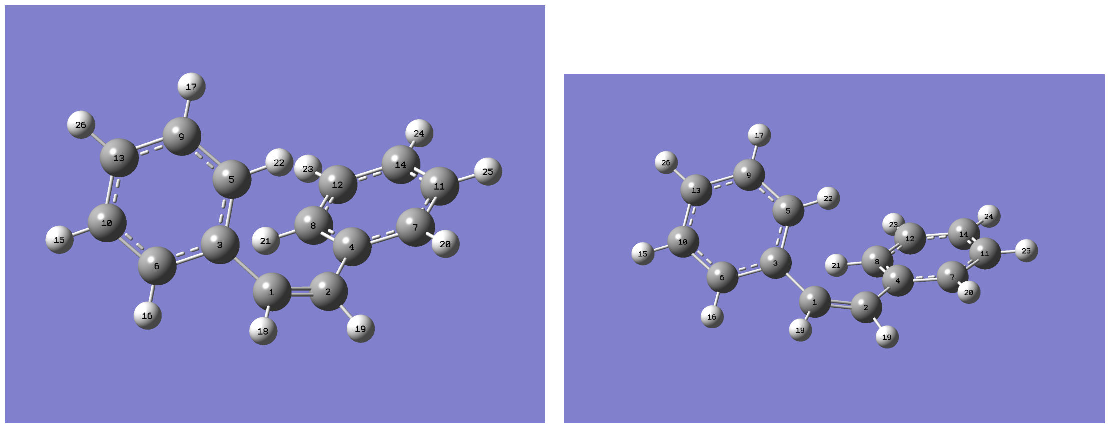


Figure 7.6: Results of 5 degree of freedom simulation of stilbene with quadratic exactness.



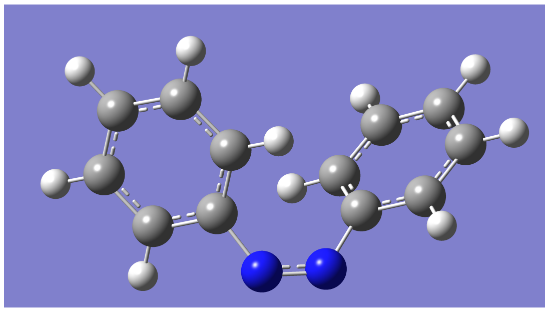
(a) Initial conformation (b) Final conformation

Figure 7.7: Initial and final geometries for 5 degree of freedom simulation of stilbene

7.3 Azobenzene

Azobenzene is another molecule with a known transition. Both stable geometries are displayed in Figure 7.8. There is some question as to how this transition occurs [54, 45, 11]. If the transition has a single intrinsic reaction coordinate, then there are different candidates for this coordinate. The first is a torsion angle similar to those for Stilbene and Butene. The other choice is a bond angle between the central torsion angle and a benzene ring. This angle was not a good candidate in the earlier molecules due to the fact that there were hydrogen bonds which we expect to rotate in symmetry with the rings. Both potential intrinsic reaction coordinates are displayed in Figure 7.9.

We know that a path does exist, but the geometric transition throughout the path is unknown. LITES is the perfect tool for resolving this question. Our methods do not require an intrinsic reaction coordinate so we may run simulations that include both coordinates as well as other coordinates to attempt to find a transition path. We have attempted simulations beginning in the cis and the trans confrormation, but neither of them have yielded a transition path. Figure 7.10 displays one simulation history for each starting conformation. We have yet to successfully find any path from cis to trans or vice versa. In either of the two images it is apparent that D1, which is the rotation of the torsion angle does not achieve a full 90° rotation before the molecule returns to the ground state. It does come within 10° of that barrier so it is possible that thermal effects could complete the transition.


(a) Cis Azobenzene

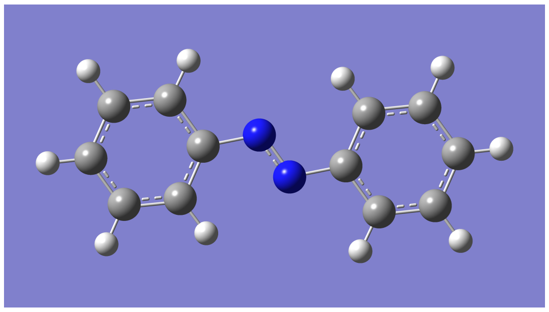
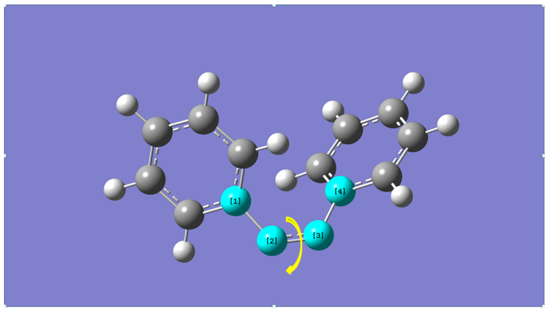
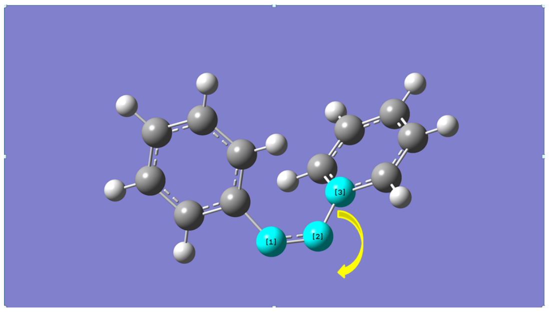

(b) Trans Azobenzene

Figure 7.8: Both stable geometries for Azobenzene, $C_{12}H_{10}N_2$

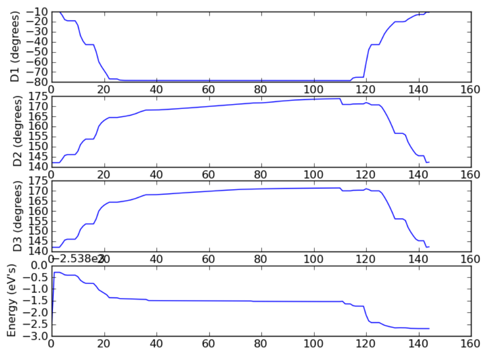


(a) Potential torsion angle rotation

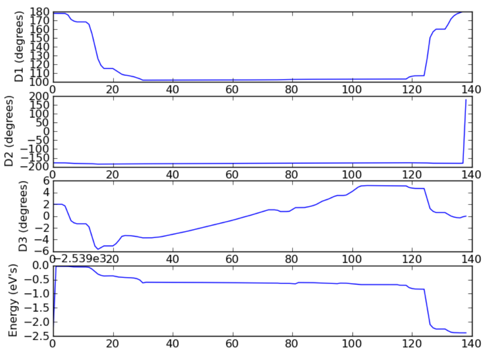


(b) Potential bond angle rotation

Figure 7.9: Potential intrinsic reaction coordinates for Azobenzene



(a) Simulation starting in the cis conformation



(b) Simulation starting in the trans conformation

Figure 7.10: Sample simulation history for Azobenzene

REFERENCES

[1] Uri M. Ascher and Linda R. Petzold. *Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations.* SIAM, Philadelphia, 1998.

[2] P. Y. Ayala and H. B. Schlegel. A combined method for determining reaction paths, minima and transition state geometries. *J. Chem. Phys.*, 107:375–84, 1997.

[3] Jon Baker and Warren J. Hehre. Geometry optimization in cartesian coordinates: The end of thez-matrix? *Journal of Computational Chemistry*, 12:606–610, 1991.

[4] Volker Barthelmann, Erich Novak, and Klaus Ritter. High dimensional polynomial interpolation on sparse grids. *Advances in Computational Mathematics*, 12:273–288, 2000.

[5] L Brutman. A note on polynomial interpolation at the chebyshev extrema nodes. *Journal of Approximation Theory*, 42:283–292, 1984.

[6] A. Bykhovski and D. Woolard. Physics and modeling of dna-derivative architectures for long-wavelength bio-sensing. 2011. to appear in Proceedings of CMOS Emerging Technologies 2011, Whistler, BC, Canada.

[7] James J. Callahan. *Advanced Calculus: A Geometric View.* Springer, 2010.

[8] R. Courant. Variational methods for the solution of problems of equilibrium and vibration. *Bull. Amer. Math. Soc.*, 49:1–43, 1943.

[9] Carl de Boor and Amos Ron. On multivariate polynomial interpolation. *Constructive Approximation*, 6:287–302, 1990.

[10] J. R. Dormand and P. J. Prince. A family of embedded runge-kutta formulae. *Journal of Computational and Applied Mathematics*, 6:19–26, 1980.

[11] R. H. El Halabieh, O. Mermut, and C. J. Barrett. Using light to control physical properties of polymers and surfaces with azobenzene chromophores. *Pure Appl. Chem.*

[12] Odon Farkas and H. Bernhard Schlegel. Methods for optimizing large molecules. part iii. an improved algorithm for geometry optimization using direct inversion in the iterative subspace (gdiis). *Phys. Chem. Chem. Phys.*, 4:11–15, 2002.

[13] Geza Fogarasi, Xuefeng Zhou, Patterson W. Taylor, and Peter Pulay. The calculation of ab initio molecular geometries: efficient optimization by natural internal coordinates and empirical correction by offset forces. *Journal of The American Chemical Society*, 114:8191–8201, 1992.

[14] J. B. Foresman and AE. Frisch. *Exploring Chemistry with Electronic Structure Methods.* Gaussian inc., Pittsburgh, Pa., second edition, 1996.

[15] F. Furche and R. Ahlrichs. Adiabatic time-dependent density functional methods for excited state properties. *J. Chem. Phys.*, 117:7433–47, 2002.

[16] Mariano Gasca and Thomas Sauer. Polynomial interpolation in several variables. *Advances in Computational Mathematics*, 12:377–410, 2000.

[17] S. Gasiorowicz. *Quantum Physics.* John Wiley & Sons, 1974.

[18] Walter Gautschi. *Numerical Analysis: An Introduction.* Birkhauser, Boston, 1997.

[19] Thomas Gerstner and Michael Griebel. Numerical integration using sparse grids. *Numer. Algorithms*, 18:209–232, 1998.

[20] Carlos Gonzalez and H. Bernhard. Schlegel. Reaction path following in mass-weighted internal coordinates. *The Journal of Physical Chemistry*, 94:5523–5527, 1990.

[21] Alan Hinchliffe. *Molecular Modeling for Beginners.* Wiley, West Sussex UK, second edition, 2008.

[22] E. Bright Wilson Jr., J.C. Decius, and Paul C. Cross. *Molecular Vibrations: The Theory of Infrared and Raman Vibrational Spectra.* Dover Publications, Philadelphia, Pa., 1980.

[23] C. T. Kelley. *Iterative Methods for Optimization.* SIAM, Philadelphia, Pa., 1999.

[24] Erwin Kreyszig. *Introductory Functional Analysis with Applications.* Wiley, 1989.

[25] J. D. Lambert. *Numerical Methods for Ordinary Differential Systems.* Wiley, New York, 2000.

[26] Ira N. Levine. *Quantum Chemistry.* Pearson Prentice Hall, Upper Saddle River, NJ, sixth edition, 2009.

[27] Xiaosong Li and Michael J. Frisch. Energy-represented direct inversion in the iterative subspace within a hybrid geometry optimization method. *Journal of Chemical Theory and Computation*, 2(3):835–839, 2006.

[28] X-L Luo, C. T. Kelley, L-Z. Liao, and H-W Tam. Combining trust region techniques and Rosenbrock methods for gradient systems. *J. Opt. Th. Appl.*, 140:265–286, 2009.

[29] Ying Luo, Boris L. Gelmont, and Dwight L. Woolard. Chapter 2 bio-molecular devices for terahertz frequency sensing. In J.M. Seminario, editor, *Molecular and Nano Electronics:Analysis, Design and Simulation*, volume 17 of *Theoretical and Computational Chemistry*, pages 55 – 81. Elsevier, 2007.

[30] F. R. Clemente M. J. Frisch, AE. Frisch and G. W. Trucks. *Gaussian 09 User's Reference*. Gaussian inc., Wallingford, CT, 2009.

[31] Robert C. McOwen. *Partial Differential Equations: Methods and Applications*. Pearson Education, Upper Saddle River, NJ, 2003.

[32] R. M. Minyaev. Molecular structure and global description of the potential energy surface. *Journal of Structural Chemistry*, 32:559–589, 1992.

[33] D. Mokrauer and C. T. Kelley. Sparse interpolatory reduced-order models for simulation of light-induced molecular transformations. 2012. submitted to Optimization Methods and Software.

[34] D. Mokrauer, C. T. Kelley, and A. Bykhovski. Efficient parallel computation of molecular potential energy surfaces for the study of light-induced transition dynamics in multiple coordinates. *Nanotechnology, IEEE Transactions on*, 10(1):70 –74, 2011.

[35] D. Mokrauer, C.T. Kelley, and A. Bykhovski. Parallel computation of surrogate models for potential energy surfaces. In *Distributed Computing and Applications to Business Engineering and Science (DCABES), 2010 Ninth International Symposium on*, pages 1 –4, 2010.

[36] Erich Novak and Klaus Ritter. High dimensional integration of smooth functions over cubes. *Numerische Mathematik*, 75:79–97, 1996.

[37] Erich Novak and Klaus Ritter. Simple cubature formulas with high polynomial exactness. *Constr. Approx.*, 15:499–522, 1999.

[38] Ian J. Palmer, Ioannis N. Ragazos, Femando Bemardi, Massimo Olivucci, and Michael A. Robb. An mc-scf study of the s1 and s2 photochemical reactions of benzene. *J. Am. Chem. Soc.*, (115):673–682, 1993.

[39] Chunyang Peng, Philippe Y. Ayala, H. Bernhard Schlegel, and Michael J. Frisch. Using redundant internal coordinates to optimize equilibrium geometries and transition states. *Journal of Computational Chemistry*, 17:49–56, 1996.

[40] L. Perko. *Differential Equations and Dynamical Systems*. Springer, New York, 3rd edition, 2001.

[41] Yehuda Pinchover and Jacob Rubinstein. *An Introduction to Partial Differential Equations*. Cambridge University Press, New York, NY, 2007.

[42] Peter Pulay. Convergence acceleration of iterative sequences. the case of scf iteration. *Chemical Physics Letters*, 73(2):393 – 398, 1980.

[43] Alfio Quarteroni, Riccardo Sacco, and Fausto Saleri. *Numerical Mathematics*. Springer, Heidelberg, 2007.

[44] Jason Quenneville and Todd J. Martinez. Ab initio study of cis-trans photoisomerization in stilbene and ethylene. *J. Phys. Chem.*, 107:829–837, 2003.

[45] Hermann Rau and Erik Lueddecke. On the rotation-inversion controversy on photoisomerization of azobenzenes. experimental proof of inversion. *Journal of the American Chemical Society*, 104(6):1616–1620, 1982.

[46] C. C. J. Roothaan. New developments in molecular orbital theory. *Reviews of Modern Physics*, 23:69–89, 1951.

[47] Jack Saltiel, Srinivasan Ganapathy, and Constance Werking. The delta h for thermal trans/cis-stilbene isomerization: do s0 and t1 potential energy curves cross? *The Journal of Physical Chemistry*, 91:2755–2758, 1987.

[48] Thomas Sauer. Lagrange interpolation on subgrids of tensor product grids. *Mathematics of Computation*, 73:181–190, 2004.

[49] Thomas Sauer and Yuan Xu. On multivariate lagrange interpolation. *Math. Comp*, 64:1147–1170, 1995.

[50] H. Bernhard Schlegel. Estimating the hessian for gradient-type geometry optimizations. *Theoretical Chemistry Accounts*, 66:333–340, 1984.

[51] H. Bernhard Schlegel. Geometry optimization. *Wiley Interdisciplinary Reviews: Computational Molecular Science*, pages n/a–n/a, 2011.

[52] L. F. Shampine. *Numerical Solution of Ordinary Differential Equations*. Chapman and Hall, 1994.

[53] Lawrence F. Shampine and Mark W. Reichelt. The matlab ode suite. *SIAM Journal on Scientific Computing*, 18.

[54] Dong Myung. Shin and David G. Whitten. Solvent-induced mechanism change in charge-transfer molecules. inversion versus rotation paths for the z .fwdarw. e isomerization of donor-acceptor substituted azobenzenes. *Journal of the American Chemical Society*, 110(15):5206–5208, 1988.

[55] Anton Simeonov, Masayuki Matsushita, Eric A. Juban, Elizabeth H. Z. Thompson, Timothy Z. Hoffman, Albert E. Beuscher IV, Matthew J. Taylor, Peter Wirsching, Wolfgang Rettig, James K. McCusker, Raymond C. Stevens, David P. Millar, Peter G. Schultz, Richard A. Lerner, and Kim D. Janda. Blue-fluorescent antibodies. *Science*, (290):307–313, 2000.

[56] S. Smolyak. Quadrature and interpolation formulas for tensor products of certain classes of functions. *Soviet Math. Dokl.*, 4:240–243, 1963.

[57] Michael Stein. Large sample properties of simulations using latin hypercube sampling. *Technometrics*, 29(2):pp. 143–151, 1987.

[58] R. E. Stratmann, G. E. Scuseria, and M. J. Frisch. An efficient implementation of time-dependent density-functional theory for the calculation of excitation energies of large molecules. *J. Chem. Phys.*, 109:8218–24, 1998.

[59] C. Van Caillie and R. D. Amos. Geometric derivatives of density functional theory excitation energies using gradient-corrected functionals. *Chem. Phys. Letters*, 317, 2000.

[60] Grzegorz W. Wasilkowski and Henryk Wozniakowski. Explicit cost bounds of algorithms for multivariate tensor product problems. *Journal of Complexity*, 11:1–56, 1995.

[61] Dongbin Xiu and Jan S. Hesthaven. High-order collocation methods for differential equations with random inputs. *Siam J. Sci. Comput.*, 27:1118–1139, 2005.

APPENDICES

Appendix A

Runge-Kutta Methods

A.1 Definition

We are solving the initial value problem

$$\begin{aligned} y' &= f(t, y) \qquad\qquad f : \mathbb{R} \times \mathbb{R}^n \to \mathbb{R}^n \\ y(t_0) &= y_0. \end{aligned} \tag{A.1}$$

Runge-Kutta methods approximate the solution to A.1 by taking single steps toward the solution [1, 25]. Each step consists of s stages defined by

$$y_{n+1} = y_n + h \sum_{i=1}^{s} b_i K_i \tag{A.2}$$

$$K_i = f(t_n + c_i h, y_n + h \sum_{j=1}^{s} a_{ij} K_j) \tag{A.3}$$

$$c_i = \sum_{j=1}^{s} a_{ij}. \tag{A.4}$$

The entire set of Runge-Kutta coefficients is collected into a Butcher array with the form:

$$\begin{array}{c|c} c & A \\ \hline & b^T \end{array}$$

Example

A 4-stage Runge-Kutta method is defined by $\begin{array}{c|cccc} 0 & 0 & 0 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\ \frac{1}{2} & 0 & \frac{1}{2} & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ \hline & \frac{1}{6} & \frac{1}{3} & \frac{1}{3} & \frac{1}{6} \end{array}$ This single step will take the form

$$y_{n+1} = y_n + h\sum_{i=1}^{4} b_i K_i. \tag{A.5}$$

Where

$$\begin{aligned} K_1 &= f(t_n, y_n) \\ K_2 &= f(t_n + \frac{1}{2}h, y_n + \frac{1}{2}hK_1) \\ K_3 &= f(t_n + \frac{1}{2}h, y_n + \frac{1}{2}hK_2) \\ K_4 &= f(t_n + h, y_n + K_3). \end{aligned}$$

Thus the full step will be

$$y_{n+1} = y_n + h\left(\frac{1}{6}K_1 + \frac{1}{3}K_2 + \frac{1}{3}K_3 + \frac{1}{6}K_4\right). \tag{A.6}$$

Calculating Runge-Kutta Coefficients

The coefficients for an RK method are chosen so that each single step matches the coefficients of the Taylor polynomial exactly up to a desired order.

Example

In a 2-stage explicit RK method we have

$$y_{n+1} = y_n + h(b_1K_1 + b_2K_2) \tag{A.7}$$
$$K_1 = f(t_n, y_n) \quad K_2 = f(t_n + hc_2, y_n + hc_2K_1).$$

Now we expand K_2 in taylor series about t_n

$$K_2 = f(t_n, y_n) + hc_2 f_t(t_n, y_n) + hc_2 K_1 f_y(t_n, y_n) + \mathcal{O}(h^2). \tag{A.8}$$

Substituting into our method we have

$$\begin{aligned} y_{n+1} &= y_n + hb_1 f(t_n, y_n) + hb_2(f(t_n, y_n) + hc_2 f_t(t_n, y_n) + hc_2 K_1 f_y(t_n, y_n) + \mathcal{O}(h^2)) \\ &= y_n + hf(t_n, y_n)(b_1 + b_2) + h^2 c_2 b_2 (f_t(t_n, y_n) + f(t_n, y_n) f_y(t_n, y_n)) + \mathcal{O}(h^3). \end{aligned} \tag{A.9}$$

The exact Taylor polynomial about the current point will be

$$\hat{y}_{n+1} = y_n + hf(t_n, y_n) + \frac{h^2}{2}(f_t(t_n, y_n) + f(t_n, y_n) f_y(t_n, y_n)) + \mathcal{O}(h^3). \tag{A.10}$$

Now we match the two equations and obtain the rules for our coefficients

$$b_1 + b_2 = 1 \quad c_2 b_2 = \frac{1}{2}. \tag{A.11}$$

For larger stage methods the same derivation will apply.

A.2 Embedding

The error in a single step of an RK method may be approximated by choosing two methods that have the same number of stages but different orders. For instance, an RK45 will approximate the error in a 4th order method using a 5th order method with the same number of stages. These two methods are embedded meaning that the same set of stages, K_i, are used for both methods. The local truncation error is approximated by taking the difference of the two methods

$$\hat{y}_{n+1} = y_n + h \sum_{i=1}^{s} \hat{b}_i K_i + \mathcal{O}(h^{p+1}) \tag{A.12}$$

$$y_{n+1} = y_n + h \sum_{i=1}^{s} b_i K_i + \mathcal{O}(h^p). \tag{A.13}$$

We assume that the higher order method is exact and thus the difference between the two methods will give us the error in the lower order method

$$\hat{y}_{n+1} - y_{n+1} = h\left(\sum_{i=1}^{s}(\hat{b}_i - b_i)K_i\right) = h(\sum_{i=1}^{s} E_i K_i) = \mathcal{O}(h^p). \tag{A.14}$$

The entire method is collected into a modified Butcher array: $\begin{array}{c|c} c & A \\ \hline & b^T \\ & \hat{b}^T \\ \hline & E^T \end{array}$

The method used in Matlab's RK45 [10, 53] was developed by Dormand and Prince with the following tableau

$$\begin{array}{c|ccccccc}
0 & & & & & & & \\
\frac{1}{5} & \frac{1}{5} & & & & & & \\
\frac{3}{10} & \frac{3}{40} & \frac{9}{40} & & & & & \\
\frac{4}{5} & \frac{44}{45} & -\frac{56}{15} & \frac{32}{9} & & & & \\
\frac{8}{9} & -\frac{19372}{6561} & -\frac{25360}{2187} & \frac{64448}{6561} & -\frac{212}{729} & & & \\
1 & \frac{9017}{3168} & -\frac{355}{33} & \frac{46732}{5247} & \frac{49}{176} & -\frac{5103}{18656} & & \\
1 & \frac{35}{384} & 0 & \frac{500}{1113} & \frac{125}{192} & -\frac{2187}{6784} & \frac{11}{84} & \\
\hline
 & \frac{5179}{57600} & 0 & \frac{7571}{16695} & \frac{393}{640} & -\frac{92097}{339200} & \frac{187}{2100} & \frac{1}{40} \\
 & \frac{35}{384} & 0 & \frac{500}{1113} & \frac{125}{192} & -\frac{2187}{6784} & \frac{11}{84} & 0 \\
\hline
 & \frac{71}{57600} & 0 & -\frac{71}{16695} & \frac{71}{1920} & -\frac{17253}{339200} & \frac{22}{525} & -\frac{1}{40}
\end{array}$$

It should be noted that the last row of A is also $\hat{b}^T$ which allows even simpler implementation.

A.3 Step-Size Control

The size of h is managed by comparing the error to some relative tolerance, δ. The relative error is given by

$$\epsilon = \frac{\|\hat{y}_{n+1} - y_{n+1}\|}{\|y_{n+1}\|} = \frac{\|h(\sum_{i=1}^{s} E_i K_i)\|}{\|y_{n+1}\|}. \tag{A.15}$$

If the relative error is greater than the specified tolerance, then we reject the current step-size and shrink. The methods for shrinking h vary, but the simplest would be to use a factor of 2. The formula found in ode45 in Matlab is

$$h = .8h\left(\frac{\delta}{\epsilon}\right)^{\frac{1}{5}}. \tag{A.16}$$

If h gets too small, then the current method cannot successfully integrate this differential equation.

A.4 Optimization

We use Runge-Kutta in order to simulate the natural descent of a molecule to a local minimum energy. In order to prevent increasing the value of the energy we added an extra criterion for accepting the stepsize, h. The step is rejected if the error tolerance is exceeded or if the step results in an increase in the objective function.

Appendix B

User's Manual

LITES (Light-Induced Transition Effects Simulator)

B.1 Overview

This software is designed to simulate the response of a single molecule to excitation. The phenomena in question is often induced by exposure of the molecule to energy in the form of light. This excitation occurs mathematically as a change in the quantum state occupied by the electrons. The change in states also forces a change in the geometry of the molecule and it relaxes to a different conformation in that same energy state. Once a stable conformation is found in the current state, the user may change states again and continue the process. This interactive simulation allows a researcher to anticipate feasible conformations of a molecule as well as find the energy required for a transition path between stable geometries.

B.2 Simulation Method

An N atom molecule is uniquely determined by $3N - 6$ internal coordinates, so the potential energy of the molecule is a function of every one of those coordinates. The computational difficulty of potential energy calculations forces us to simplify the model by assuming the energy is only a function of a few coordinates, the design variables. The selection of the design variables is left to the user.

In order to improve the performance of the computational chemistry software it may be necessary to specify more information about the coordinates than just the design variables. For example, if a design variable is the rotation of a ring about a bond then there may be an atom opposite the ring which will also rotate with the bond. The simulation allows the user to specify any angles which would benefit from this type of pre-processing. The sample simulation included in this document includes pre-processing.

The simulator determines the natural conformational changes likely to occur due to changes in the energy state of a molecule. In order to predict this transition the software builds a surrogate model (patch) of the potential energy of the molecule in a local area around the current values of the design variables. This means that the energy of the molecule is computed for a pre-specified set of gridpoints in the design space. The model is a polynomial interpolation between the gridpoints. This surrogate is only accurate inside the domain from which it was constructed. From the surrogate model we use steepest descent to predict the natural relaxation of the molecule. If the gradient flow leads to the boundary of the domain of the surrogate model, then a new surrogate model is built to continue the descent. Once the gradient flow concludes at a local minimum within the domain of the surrogate model, we may change energy states and continue building patches until the sequence of desired states specified by the user have concluded at a local minimum.

The simulation time is heavily dependent on the size of the molecule as well as the desired accuracy of the patches. Our test molecule 2-butene has 12 atoms and may requiree nearly 2 minutes for an energy computation. The molecule stilbene which has 26 atoms may require around 45 minutes for an energy computation. Each patch in a simulation with 5 design variables with a quadraticly exact surrogate requires an energy computation at each of 241 gridpoints.

This manual is separated into 2 halves. The first half is a full sample simulation and the second half is an explanation of the inputs and outputs.

B.3 Summary of Tasks for Running LITES

In order to run a simulation with LITES you must:

1. Obtain tarball LITES.tar from http://www4.ncsu.edu/ ctk/Lites.html

2. Obtain permission to use the HPC and Gaussian

3. Create a LITES directory
4. Create a directory for the LITES output files
5. Change to the LITES directory
6. Transfer LITES.tar to your LITES directory
7. Unpack the tarball LITES.tar
8. Edit the job submission file to reflect your directories
9. Edit the file chem_vars.py for the simulation you would like to perform
10. Make an initial Z-matrix file named zmat.gjf
11. Submit your simulation using bsub ¡ LITES_in
12. When complete copy and rename the files Simulation_Summary.txt and sim_plot.png from your data directory to a location where you may store and view them.

B.4 Example: 2-butene Simulation

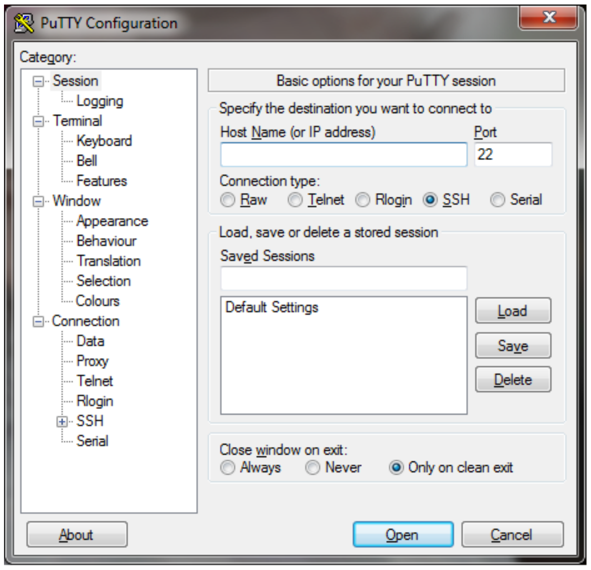
B.4.1 Permissions and HPC

The first requirement in order to use this simulation is that you have the proper permissions to use the hpc. In obtaining access, you must also gain specific access to use Gaussian 09. This requires signing papers, so please be aware.

Once all access has been granted you must ssh to the cluster. If you are using windows (as I do), then I recommend a program called putty. If you use a Mac, then you will use terminal. These instructions describe the use of putty. When you open the client it will look as in Figure B.1. Next fill in the textbox labeled Host Name with "login64.hpc.ncsu.edu" as in Figure B.2 and click Open. This will prompt you for your login name which is your unity id as in Figure B.3. Finally you will need your unity password as in Figure B.4. If your login has been successful it will look similar to Figure B.5.

In the next section you will create the directory LITES, but you must transfer the tarball to that directory. If you use a mac, then you will need to use Fugu. If you use windows, then I recommend downloading WinSCP. I will describe the process using this program.

When you open WinSCP you will see the screen in Figure B.6. Click "New" and you will see Figure B.7. Fill in the Host name, User name, and Password then click "Save" as in Figure B.8. Now the session will appear in the original screen like in Figure **??**. Now click "Login" and you will see 2 columns of directories. The left directories are on your computer and the right directories are on the hpc as in Figure B.9. Now find the file LITES.tar on your computer and drag it to the LITES directory on the hpc.



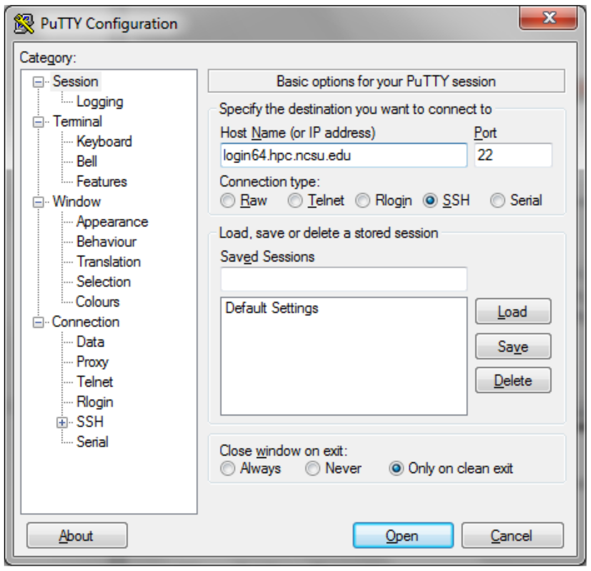
Port: 22

Connection type: Raw Telnet Rlogin [✓ SSH] Serial

Saved Sessions: Default Settings

Close window on exit: Always Never [✓ Only on clean exit]

Figure B.1: View upon opening putty



Specify the destination you want to connect to

Host Name (or IP address): login64.hpc.ncsu.edu
Port: 22

Connection type: Raw Telnet Rlogin [✓ SSH] Serial

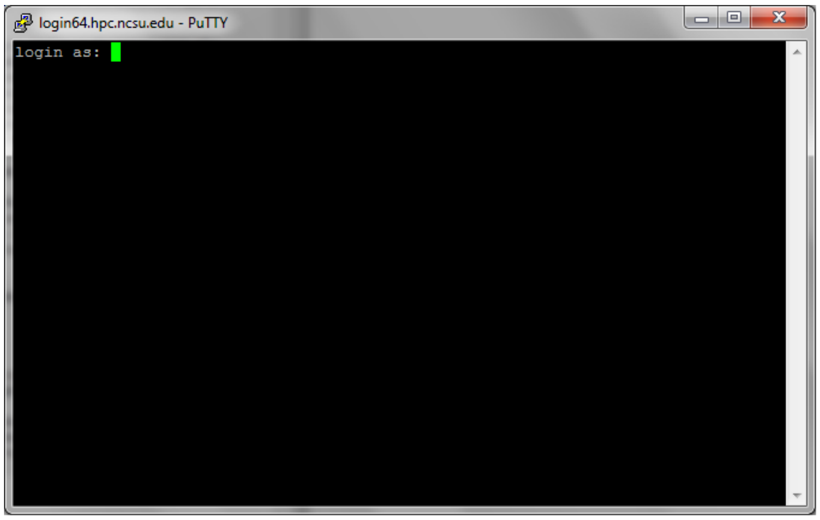
Load, save or delete a stored session

Saved Sessions

Default Settings

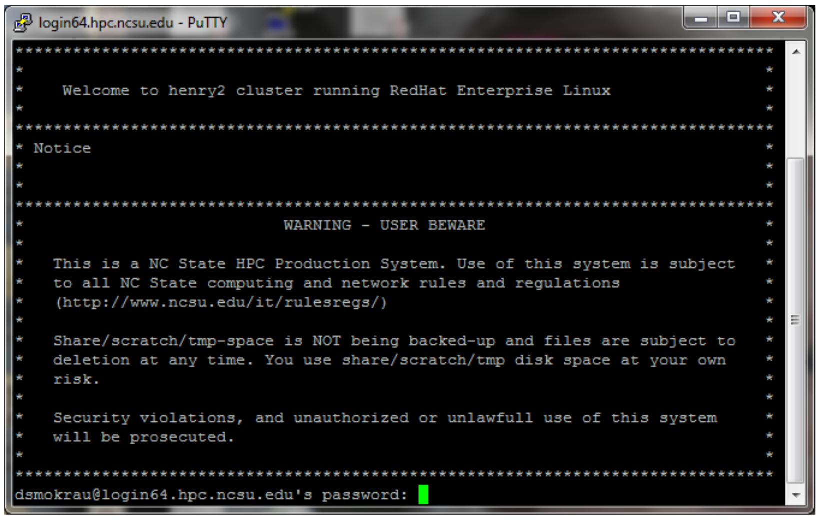
Close window on exit: Always Never [✓ Only on clean exit]

Figure B.2: putty with host queue



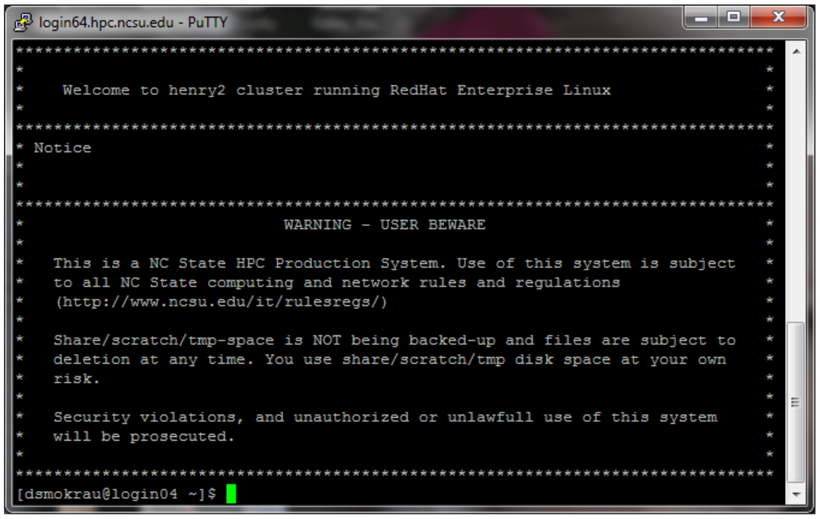
```
login as:
```

Figure B.3: putty login screen



```
login64.hpc.ncsu.edu - PuTTY
******************************************************************************
*                                                                            *
*    Welcome to henry2 cluster running RedHat Enterprise Linux               *
*                                                                            *
******************************************************************************
* Notice                                                                     *
*                                                                            *
*                                                                            *
******************************************************************************
*                           WARNING - USER BEWARE                            *
*                                                                            *
*   This is a NC State HPC Production System. Use of this system is subject  *
*   to all NC State computing and network rules and regulations              *
*   (http://www.ncsu.edu/it/rulesregs/)                                      *
*                                                                            *
*   Share/scratch/tmp-space is NOT being backed-up and files are subject to  *
*   deletion at any time. You use share/scratch/tmp disk space at your own   *
*   risk.                                                                    *
*                                                                            *
*   Security violations, and unauthorized or unlawfull use of this system    *
*   will be prosecuted.                                                      *
*                                                                            *
******************************************************************************
dsmokrau@login64.hpc.ncsu.edu's password:
```

Figure B.4: putty login/password screen



```
login64.hpc.ncsu.edu - PuTTY
******************************************************************************
*                                                                            *
*    Welcome to henry2 cluster running RedHat Enterprise Linux               *
*                                                                            *
******************************************************************************
* Notice                                                                     *
*                                                                            *
*                                                                            *
******************************************************************************
*                           WARNING - USER BEWARE                            *
*                                                                            *
*   This is a NC State HPC Production System. Use of this system is subject  *
*   to all NC State computing and network rules and regulations              *
*   (http://www.ncsu.edu/it/rulesregs/)                                      *
*                                                                            *
*   Share/scratch/tmp-space is NOT being backed-up and files are subject to  *
*   deletion at any time. You use share/scratch/tmp disk space at your own   *
*   risk.                                                                    *
*                                                                            *
*   Security violations, and unauthorized or unlawfull use of this system    *
*   will be prosecuted.                                                      *
*                                                                            *
******************************************************************************
[dsmokrau@login04 ~]$
```

Figure B.5: successful login

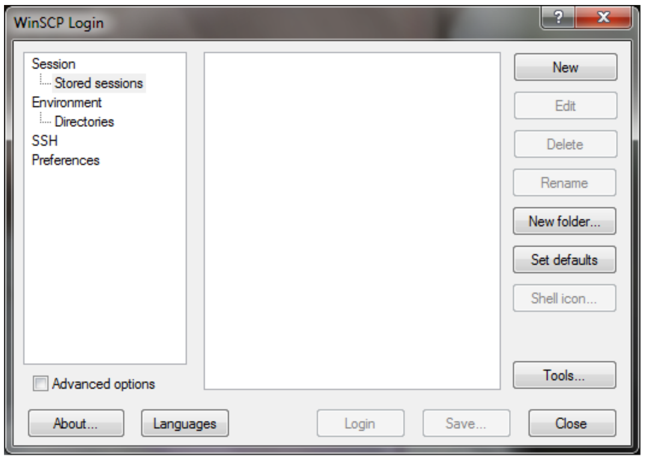
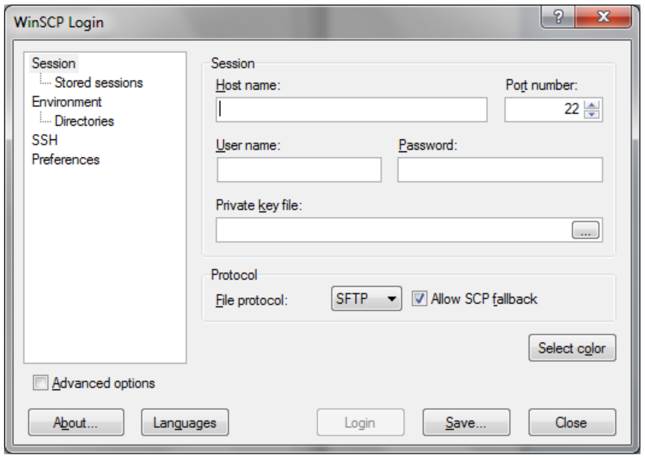


Figure B.6: Initial WinSCP screen

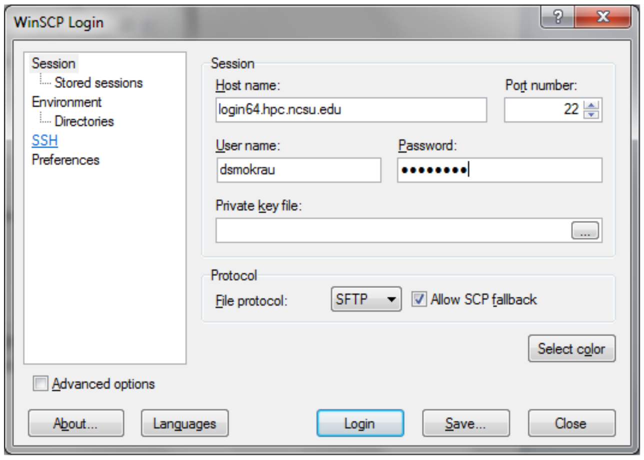


Port number: 22

File protocol: SFTP

[✓] Allow SCP fallback

Figure B.7: Result of clicking ”New”



Session

Host name: login64.hpc.ncsu.edu

Port number: 22

User name: dsmokrau

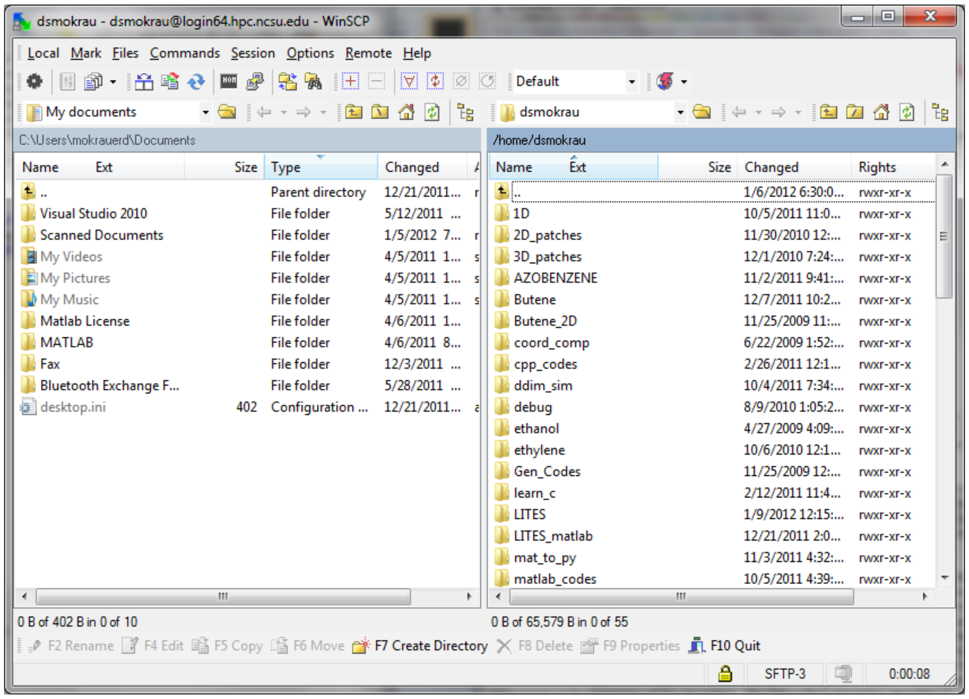
Password: ••••••••

Private key file:

File protocol: SFTP

[✓] Allow SCP fallback

Figure B.8: Filled out WinSCP session


dsmokrau - dsmokrau@login64.hpc.ncsu.edu - WinSCP
Local Mark Files Commands Session Options Remote Help
Default
My documents
dsmokrau
F2 Rename F4 Edit F5 Copy F6 Move F7 Create Directory F8 Delete F9 Properties F10 Quit
SFTP-3
0:00:08

C:\Users\mokrauerd\Documents

Name	Ext	Size	Type	Changed
..			Parent directory	12/21/2011...
Visual Studio 2010			File folder	5/12/2011 ...
Scanned Documents			File folder	1/5/2012 7...
My Videos			File folder	4/5/2011 1...
My Pictures			File folder	4/5/2011 1...
My Music			File folder	4/5/2011 1...
Matlab License			File folder	4/6/2011 1...
MATLAB			File folder	4/6/2011 8...
Fax			File folder	12/3/2011 ...
Bluetooth Exchange F...			File folder	5/28/2011 ...
desktop.ini		402	Configuration ...	12/21/2011...

0 B of 402 B in 0 of 10

/home/dsmokrau

Name	Ext	Size	Changed	Rights
..			1/6/2012 6:30:0...	rwxr-xr-x
1D			10/5/2011 11:0...	rwxr-xr-x
2D_patches			11/30/2010 12:...	rwxr-xr-x
3D_patches			12/1/2010 7:24:...	rwxr-xr-x
AZOBENZENE			11/2/2011 9:41:...	rwxr-xr-x
Butene			12/7/2011 10:2...	rwxr-xr-x
Butene_2D			11/25/2009 11:...	rwxr-xr-x
coord_comp			6/22/2009 1:52:...	rwxr-xr-x
cpp_codes			2/26/2011 12:1...	rwxr-xr-x
ddim_sim			10/4/2011 7:34:...	rwxr-xr-x
debug			8/9/2010 1:05:2...	rwxr-xr-x
ethanol			4/27/2009 4:09:...	rwxr-xr-x
ethylene			10/6/2010 12:1...	rwxr-xr-x
Gen_Codes			11/25/2009 12:...	rwxr-xr-x
learn_c			2/12/2011 11:4...	rwxr-xr-x
LITES			1/9/2012 12:15:...	rwxr-xr-x
LITES_matlab			12/21/2011 2:0...	rwxr-xr-x
mat_to_py			11/3/2011 4:32:...	rwxr-xr-x
matlab_codes			10/5/2011 4:39:...	rwxr-xr-x

0 B of 65,579 B in 0 of 55

Figure B.9: WinSCP session, your computer on the left and the hpc on the right

B.4.2 First Simulation

The following is a step by step example for setting up and simulating the molecule 2-butene with 3 degrees of freedom.

1. Create the Lites directory:

```
[dsmokrau@login04 ~]$ mkdir LITES
```

2. Create a directory for writing your simulation, I use the shared space /kelley_data

```
[dsmokrau@login04 ~]$ mkdir /kelley_data/dsmokrau/db_LITES
```

3. Change to the new directory

```
[dsmokrau@login04 ~]$ cd LITES
```

4. Transfer the file LITES.tar to the directory you just created as described in the previous section

5. Unpack the LITES files

```
[dsmokrau@login04 ~/LITES]$ tar -xf LITES.tar
```

6. Open the job submission file using your favorite editor (I use nano)

```
[dsmokrau@login04 ~/LITES]$ nano LITES_in
```

7. Edit the job submission file by replacing each of the my directories with the ones you created. Each line with dsmokrau should be replaced by your unity id and the directories you have just created

```
#!/bin/csh

#BSUB -W 10000
#BSUB -R em64t
#BSUB -x
#BSUB -n 1
#BSUB -q gto
#BSUB -o /kelley_data/dsmokrau/db_LITES/LITES.out
#BSUB -e /kelley_data/dsmokrau/db_LITES/LITES.err
```

```
rm -r /scratch/dsmokrau

mkdir /scratch/dsmokrau
cp /home/dsmokrau/LITES/* /scratch/dsmokrau
cd /scratch/dsmokrau

/usr/local/apps/python-2.6.5/bin/python LITES.py

cd
rm -r /scratch/dsmokrau
```

8. Save and exit with Ctrl -o followed by Ctrl -x, if you are using nano

9. Open the file chem_vars.py

```
[dsmokrau@login04 ~/LITES]$ nano chem_vars.py
```

10. Edit the file chem_vars.py. Each input will be described at length later in the manual. The file initially appears as

```
#! usr/local/apps/python-2.6.5/bin/python
#chemistry inputs for light-induced simulation

#number of design variables
d =

#name of your molecule
molecule = ''

#the number of atoms in your molecule
num_atoms =

#the number of cores for your gaussian jobs
n_proc =

#method of energy computation (hf,b3lyp,...)
method = ''

#basis-set for you gaussian jobs (cep-31g,CBSB7,...)
basis_set = ''
```

```
#Gaussian job type (opt=modredundant) and anything else you want in the header
calculation = ''

#state transitions for your simulation
state_order = []

#where will your gaussian jobs be run?
queue = ''

#set this to 1 if you want every z-matrix that was calculated during your sim
all_zmats =

#atom number, value, ...
bond_lengths = []

#for reading output files, i.e. 'C1-C2'
bond_names = []

#atom numbers, value, ...
valence_angles = []

#for reading output files, i.e. 'C1-C2-C6'
valence_names = []

#atom numbers, value, ...
torsion_angles = []

#for reading output files, i.e. 'C1-C2-C6-C8'
torsion_names = []

#pre-processed variable, pre-processed to whom, amount to pre-process
deps = []

#where your molecule sits, this directory must have initial zmat as zmat.gjf
home_dir = ''

#where to write all your files
share_dir =  ''
```

Once edited, it should appear as (note the differences between 1 and l):

```
#! usr/local/apps/python-2.6.5/bin/python
#chemistry inputs for light-induced simulation

#number of design variables
d = 3

#name of your molecule
molecule = 'Butene'

#the number of atoms in your molecule
num_atoms = 12

#the number of cores for your gaussian jobs
n_proc = 2

#method of energy computation (hf,b3lyp,...)
method = 'b3lyp'

#basis-set for you gaussian jobs (cep-31g,CBSB7,...)
basis_set = 'cep-31g'

#Gaussian job type (opt=modredundant) and anything else you want in the header
calculation = 'opt=modredundant'

#state transitions for your simulation
state_order = [1,0]

#where will your gaussian jobs be run?
queue = 'gto'

#set this to 1 if you want every z-matrix that was calculated during your sim
all_zmats = 0

#atom number, value, ...
bond_lengths = []

#for reading output files, i.e. 'C1-C2'
bond_names = []

#atom numbers, value, ...
```

```
valence_angles = []

#for reading output files, i.e. 'C1-C2-C6'
valence_names = []

#atom numbers, value, ...
torsion_angles = ['1 2 6 8', 0., '6 2 1 5', 120., '2 6 8 11', -120.]

#for reading output files, i.e. 'C1-C2-C6-C8'
torsion_names = ['C1-C2-C6-C8', 'C6-C2-C1-H5', 'C2-C6-C8-H11']

#pre-processed variable, pre-processed to whom, amount to pre-process
deps = ['1 2 6 9', 0, 180., '8 6 2 7', 0, 180., '9 6 2 7', 0, 0.]

#where your molecule sits, this directory must have initial zmat as zmat.gjf
home_dir = '/home/dsmokrau/LITES'

#where to write all your files
share_dir =  '/kelley_data/dsmokrau/db_LITES'
```

11. Save and exit with Ctrl -o followed by Ctrl -x, if you are using nano

12. Open zmat.gjf

```
[dsmokrau@login04 ~/LITES]$ nano zmat.gjf
```

13. The Z-matrix should be the only data in the file so edit the initial file from

```
%chk=/scratch/butene.chk
%mem=400MW
%NprocShared=1
# opt=modredundant b3lyp/cep-31g geom=(connectivity,nodistance) scf=tight

cis 2 butene D5=1-2-6-8

0 1
 C
 C                  1             B1
 H                  1             B2    2             A1
 H                  1             B3    2             A2    3               D1
 H                  1             B4    2             A3    3               D2
```

```
C                   2             B5    1              A4    4              D3
H                   2             B6    1              A5    6              D4
C                   6             B7    2              A6    1              D5
H                   6             B8    2              A7    1              D6
H                   8             B9    6              A8    2              D7
H                   8            B10    6              A9    2              D8
H                   8            B11    6             A10    2              D9

  B1              1.52880970
  B2              1.10731414
  B3              1.10124709
  B4              1.10730304
  B5              1.37283300
  B6              1.09857439
  B7              1.52880951
  B8              1.09857350
  B9              1.10730531
  B10             1.10731351
  B11             1.10124579
  A1            110.62562667
  A2            112.79042258
  A3            110.62199176
  A4            127.18758000
  A5            115.32977589
  A6            127.18763251
  A7            117.48266812
  A8            110.62184935
  A9            110.62555997
  A10           112.79080645
  D1            120.95156450
  D2           -118.09976127
  D3              0.07453914
  D4            179.99656673
  D5              0.00000000
  D6            180.00000000
  D7            121.02757116
  D8           -120.87281467
  D9              0.07914737

1 2 1.0 3 1.0 4 1.0 5 1.0
```

```
2 6 2.0 7 1.0
3
4
5
6 8 1.0 9 1.0
7
8 10 1.0 11 1.0 12 1.0
9
10
11
12

D 1 2 6 8 F
D 7 2 6 9
D 1 2 6 9
D 7 2 6 8
```

to contain only the Z-matrix as follows

```
C
C                  1             B1
H                  1             B2    2              A1
H                  1             B3    2              A2    3              D1
H                  1             B4    2              A3    3              D2
C                  2             B5    1              A4    4              D3
H                  2             B6    1              A5    6              D4
C                  6             B7    2              A6    1              D5
H                  6             B8    2              A7    1              D6
H                  8             B9    6              A8    2              D7
H                  8            B10    6              A9    2              D8
H                  8            B11    6             A10    2              D9

  B1              1.52880970
  B2              1.10731414
  B3              1.10124709
  B4              1.10730304
  B5              1.37283300
  B6              1.09857439
  B7              1.52880951
  B8              1.09857350
  B9              1.10730531
```

```
B10            1.10731351
B11            1.10124579
A1           110.62562667
A2           112.79042258
A3           110.62199176
A4           127.18758000
A5           115.32977589
A6           127.18763251
A7           117.48266812
A8           110.62184935
A9           110.62555997
A10          112.79080645
D1           120.95156450
D2          -118.09976127
D3             0.07453914
D4           179.99656673
D5             0.00000000
D6           180.00000000
D7           121.02757116
D8          -120.87281467
D9             0.07914737
```

14. Save and exit with Ctrl -o followed by Ctrl -x, if you are using nano

15. Submit the simulation using bsub

```
[dsmokrau@login04 ~/LITES]$ bsub < LITES_in
Job <491566> is submitted to queue <gto>.
[dsmokrau@login04~/LITES]$
```

16. Check the status of you simulation using the command bhist

```
[dsmokrau@login04 db_LITES]$ bhist
Summary of time in seconds spent in various states:
JOBID   USER    JOB_NAME  PEND    PSUSP   RUN     USUSP   SSUSP   UNKWN   TOTAL
491566  dsmokra *smokrau  5       0       57      0       0       0       62
491567  dsmokra E_files0  7       0       35      0       0       0       42
491568  dsmokra E_files1  7       0       35      0       0       0       42
491569  dsmokra E_files2  6       0       35      0       0       0       41
491570  dsmokra E_files3  6       0       35      0       0       0       41
.
```

```
.
.
491632  dsmokra *files65  30      0       0       0       0       0       30
491633  dsmokra *files66  30      0       0       0       0       0       30
491634  dsmokra *files67  30      0       0       0       0       0       30
491635  dsmokra *files68  30      0       0       0       0       0       30

[dsmokrau@login04 db_LITES]$
```

17. The simulation has completed when you receive the following

```
[dsmokrau@login04 db_LITES]$ bhist
No matching job found
[dsmokrau@login04 db_LITES]$
```

18. If you would like to end a running simulation, use the command

```
[dsmokrau@login04 db_LITES]$ bkill 0
```

19. When you begin a simulation the share directory you are using should be empty and the command to delete all files in a directory is

```
[dsmokrau@login04 db_LITES]$ rm *
```

B.4.3 Output

Once the simulation has begun to execute you may check the status of the simulation by checking the output file Simulation_summary.txt. This file will contain a full summary of the simulation once it is completed, but serves as a checkpoint file throughout the simulation. Here is the resulting file from the simulation we just submitted.

```
[dsmokrau@login04 ~/LITES]$ cd /kelley_data/dsmokrau/db_LITES/
[dsmokrau@login04 dd_debug]$ nano Simulation_Summary.txt
```

```
Thank you for using LITES
You will be simulating the molecule Butene using b3lyp with cep-31g as a basis set
We begin on state 1
with initial value of x = [   0.  120. -120.]
Initial patch size is 20.0
Beginning the simulation with the following Z-matrix
 C
 C                1             B1
```

```
H                  1              B2    2              A1
H                  1              B3    2              A2    3              D1
H                  1              B4    2              A3    3              D2
C                  2              B5    1              A4    4              D3
H                  2              B6    1              A5    6              D4
C                  6              B7    2              A6    1              D5
H                  6              B8    2              A7    1              D6
H                  8              B9    6              A8    2              D7
H                  8             B10    6              A9    2              D8
H                  8             B11    6             A10    2              D9

  B1            1.52880970
  B2            1.10731414
  B3            1.10124709
  B4            1.10730304
  B5            1.37283300
  B6            1.09857439
  B7            1.52880951
  B8            1.09857350
  B9            1.10730531
  B10           1.10731351
  B11           1.10124579
  A1          110.62562667
  A2          112.79042258
  A3          110.62199176
  A4          127.18758000
  A5          115.32977589
  A6          127.18763251
  A7          117.48266812
  A8          110.62184935
  A9          110.62555997
  A10         112.79080645
  D1          120.95156450
  D2         -118.09976127
  D3            0.07453914
  D4          179.99656673
  D5            0.00000000
  D6          180.00000000
  D7          121.02757116
  D8         -120.87281467
  D9            0.07914737


Now for the patch summaries!


PATCH NUMBER 0
Patch size was 20.0
Current state is state 1
Maximum Gaussian iterations was 5
```

```
Initial point for the patch was [   0.  120. -120.]
Patch terminated at [  2.64463711e-02   1.10427830e+02  -1.10386277e+02]
Terminal point hit a boundary of the patch, drawing new patch!

Now the Z-matrix, Patch, and Path
Z-matrix for patch 0 was
  C
 C                  1             B1
 H                  1             B2    2              A1
 H                  1             B3    2              A2    3              D1
 H                  1             B4    2              A3    3              D2
 C                  2             B5    1              A4    4              D3
 H                  2             B6    1              A5    6              D4
 C                  6             B7    2              A6    1              D5
 H                  6             B8    2              A7    1              D6
 H                  8             B9    6              A8    2              D7
 H                  8            B10    6              A9    2              D8
 H                  8            B11    6             A10    2              D9

   B1             1.52880970
   B2             1.10731414
   B3             1.10124709
   B4             1.10730304
   B5             1.37283300
   B6             1.09857439
   B7             1.52880951
   B8             1.09857350
   B9             1.10730531
   B10            1.10731351
   B11            1.10124579
   A1           110.62562667
   A2           112.79042258
   A3           110.62199176
   A4           127.18758000
   A5           115.32977589
   A6           127.18763251
   A7           117.48266812
   A8           110.62184935
   A9           110.62555997
   A10          112.79080645
   D1           120.95156450
   D2          -118.09976127
   D3             0.07453914
   D4           179.99656673
   D5             0.00000000
   D6           180.00000000
   D7           121.02757116
   D8          -120.87281467
   D9             0.07914737
```

```
Patch 0
1*2*6*8 6*2*1*5 2*6*8*11 state0 state1
-10.0000000,110.0000000,-130.0000000,-744.2032174,-736.9677174
-10.0000000,110.0000000,-120.0000000,-744.2079530,-736.9393530
-10.0000000,110.0000000,-110.0000000,-744.2080695,-736.9375695
-10.0000000,112.9289320,-120.0000000,-744.2097294,-736.9284294
-10.0000000,120.0000000,-130.0000000,-744.2085142,-736.9363142
-10.0000000,120.0000000,-127.0710680,-744.2101750,-736.9261750
-10.0000000,120.0000000,-120.0000000,-744.2126268,-736.9113268
-10.0000000,120.0000000,-112.9289320,-744.2127531,-736.9123531
-10.0000000,120.0000000,-110.0000000,-744.2121621,-736.9174621
-10.0000000,127.0710680,-120.0000000,-744.2132734,-736.9096734
-10.0000000,130.0000000,-130.0000000,-744.2093728,-736.9311728
-10.0000000,130.0000000,-120.0000000,-744.2128879,-736.9136879
-10.0000000,130.0000000,-110.0000000,-744.2119248,-736.9282248
-9.2387950,120.0000000,-120.0000000,-744.2152360,-736.9065360
-7.0710680,110.0000000,-120.0000000,-744.2176155,-736.9211155
-7.0710680,120.0000000,-130.0000000,-744.2182216,-736.9179216
-7.0710680,120.0000000,-120.0000000,-744.2215013,-736.8953013
-7.0710680,120.0000000,-110.0000000,-744.2204466,-736.9050466
-7.0710680,130.0000000,-120.0000000,-744.2211417,-736.9011417
-3.8268340,120.0000000,-120.0000000,-744.2277199,-736.8846199
0.0000000,110.0000000,-130.0000000,-744.2257746,-736.9202746
0.0000000,110.0000000,-127.0710680,-744.2268349,-736.9106349
0.0000000,110.0000000,-120.0000000,-744.2279236,-736.8977236
0.0000000,110.0000000,-112.9289320,-744.2269754,-736.9006754
0.0000000,110.0000000,-110.0000000,-744.2259811,-736.9064811
0.0000000,110.7612050,-120.0000000,-744.2282451,-736.8952451
0.0000000,112.9289320,-130.0000000,-744.2270136,-736.9096136
0.0000000,112.9289320,-120.0000000,-744.2290336,-736.8887336
0.0000000,112.9289320,-110.0000000,-744.2269754,-736.9006754
0.0000000,116.1731660,-120.0000000,-744.2298610,-736.8823610
0.0000000,120.0000000,-130.0000000,-744.2285819,-736.8940819
0.0000000,120.0000000,-129.2387950,-744.2288541,-736.8914541
0.0000000,120.0000000,-127.0710680,-744.2295090,-736.8861090
0.0000000,120.0000000,-123.8268340,-744.2301184,-736.8809184
0.0000000,120.0000000,-120.0000000,-744.2302755,-736.8791755
0.0000000,120.0000000,-116.1731660,-744.2298610,-736.8823610
0.0000000,120.0000000,-112.9289320,-744.2290336,-736.8887336
0.0000000,120.0000000,-110.7612050,-744.2282451,-736.8952451
0.0000000,120.0000000,-110.0000000,-744.2279236,-736.8977236
0.0000000,123.8268340,-120.0000000,-744.2301185,-736.8809185
0.0000000,127.0710680,-130.0000000,-744.2281151,-736.8940151
0.0000000,127.0710680,-120.0000000,-744.2295091,-736.8861091
0.0000000,127.0710680,-110.0000000,-744.2268352,-736.9105352
0.0000000,129.2387950,-120.0000000,-744.2288581,-736.8917581
0.0000000,130.0000000,-130.0000000,-744.2273047,-736.8989047
0.0000000,130.0000000,-127.0710680,-744.2281151,-736.8940151
0.0000000,130.0000000,-120.0000000,-744.2285819,-736.8940819
0.0000000,130.0000000,-112.9289320,-744.2270136,-736.9096136
0.0000000,130.0000000,-110.0000000,-744.2257747,-736.9201747
```

```
3.8268340,120.0000000,-120.0000000,-744.2277199,-736.8846199
7.0710680,110.0000000,-120.0000000,-744.2204466,-736.9050466
7.0710680,120.0000000,-130.0000000,-744.2211417,-736.9011417
7.0710680,120.0000000,-120.0000000,-744.2215013,-736.8953013
7.0710680,120.0000000,-110.0000000,-744.2176155,-736.9211155
7.0710680,130.0000000,-120.0000000,-744.2182216,-736.9179216
9.2387950,120.0000000,-120.0000000,-744.2152360,-736.9065360
10.0000000,110.0000000,-130.0000000,-744.2119232,-736.9300232
10.0000000,110.0000000,-120.0000000,-744.2121621,-736.9174621
10.0000000,110.0000000,-110.0000000,-744.2080695,-736.9375695
10.0000000,112.9289320,-120.0000000,-744.2127531,-736.9123531
10.0000000,120.0000000,-130.0000000,-744.2128879,-736.9136879
10.0000000,120.0000000,-127.0710680,-744.2132734,-736.9096734
10.0000000,120.0000000,-120.0000000,-744.2126268,-736.9113268
10.0000000,120.0000000,-112.9289320,-744.2097294,-736.9284294
10.0000000,120.0000000,-110.0000000,-744.2079530,-736.9393530
10.0000000,127.0710680,-120.0000000,-744.2101750,-736.9261750
10.0000000,130.0000000,-130.0000000,-744.2093728,-736.9311728
10.0000000,130.0000000,-120.0000000,-744.2085142,-736.9363142
10.0000000,130.0000000,-110.0000000,-744.2032174,-736.9677174
Path 0
0.0000000,120.0000000,-120.0000000,-744.2302755,0.0000000
0.0000000,120.0000000,-120.0000000,-736.8791755,0.0000000
-0.0000000,119.9999981,-119.9999981,-736.8791755,0.0000000
-0.0000000,119.9991230,-119.9991229,-736.8791759,0.0000000
0.0000003,119.8601736,-119.8601642,-736.8792331,0.0000000
0.0001121,118.4531587,-118.4527684,-736.8803288,0.0000001
0.0041719,114.7697414,-114.7618971,-736.8876443,0.0000000
0.0159476,111.8967140,-111.8708534,-736.8977915,0.0000004
0.0264464,110.4278301,-110.3862774,-736.9044682,0.0000004



PATCH NUMBER 1
Patch size was 40.310163591
Current state is state 1
Maximum Gaussian iterations was 5
Initial point for the patch was [  2.64463711e-02   1.10427830e+02  -1.10386277e+02]
Patch terminated at [ 11.6457299   77.02389331 -75.00212746]
Terminal point hit a boundary of the patch, drawing new patch!

Now the Z-matrix, Patch, and Path
Z-matrix for patch 1 was
 C
C,1,B1
H,1,B2,2,A1
H,1,B3,2,A2,3,D1,0
H,1,B4,2,A3,3,D2,0
C,2,B5,1,A4,4,D3,0
H,2,B6,1,A5,6,D4,0
C,6,B7,2,A6,1,D5,0
```

```
H,6,B8,2,A7,1,D6,0
H,8,B9,6,A8,2,D7,0
H,8,B10,6,A9,2,D8,0
H,8,B11,6,A10,2,D9,0
Variables:
B1=1.52919032
B2=1.10631918
B3=1.10162245
B4=1.10787371
B5=1.37292981
B6=1.09843946
B7=1.52919029
B8=1.09843953
B9=1.1063192
B10=1.10787373
B11=1.10162245
A1=110.70937289
A2=112.66207909
A3=110.68500888
A4=127.02642994
A5=115.44734914
A6=127.02645455
A7=117.52089398
A8=110.70934521
A9=110.68502299
A10=112.6620935
D1=121.11494029
D2=-118.4383603
D3=-10.44669938
D4=-179.13272096
D5=0.
D6=-179.11713503
D7=131.56165385
D8=-110.00000003
D9=10.4467318


Patch 1
1*2*6*8 6*2*1*5 2*6*8*11 state0 state1
-20.1286354,74.1486829,-114.4172937,-744.1221569,-737.1819569
-20.1286354,74.1486829,-94.2622119,-744.1180248,-737.2198248
-20.1286354,74.1486829,-74.1071301,-744.1001380,-737.2691380
-20.1286354,80.0519693,-94.2622119,-744.1206420,-737.1982420
-20.1286354,94.3037647,-114.4172937,-744.1356715,-737.1037715
-20.1286354,94.3037647,-108.5140073,-744.1369506,-737.1044506
-20.1286354,94.3037647,-94.2622119,-744.1321428,-737.1287428
-20.1286354,94.3037647,-80.0104165,-744.1193299,-737.1718299
-20.1286354,94.3037647,-74.1071301,-744.1130442,-737.1893442
-20.1286354,108.5555601,-94.2622119,-744.1449556,-737.0645556
-20.1286354,114.4588465,-114.4172937,-744.1549643,-737.0111643
-20.1286354,114.4588465,-94.2622119,-744.1492619,-737.0493619
```

```
-20.1286354,114.4588465,-74.1071301,-744.1288821,-737.1390821
-18.5944205,94.3037647,-94.2622119,-744.1434405,-737.1134405
-14.2253490,74.1486829,-94.2622119,-744.1571206,-737.1717206
-14.2253490,94.3037647,-114.4172937,-744.1765005,-737.0474005
-14.2253490,94.3037647,-94.2622119,-744.1704358,-737.0743358
-14.2253490,94.3037647,-74.1071301,-744.1533049,-737.1419049
-14.2253490,114.4588465,-94.2622119,-744.1844124,-737.0029124
-7.6865689,94.3037647,-94.2622119,-744.1970212,-737.0327212
0.0264464,74.1486829,-114.4172937,-744.2046089,-737.0795089
0.0264464,74.1486829,-108.5140073,-744.2027434,-737.0818434
0.0264464,74.1486829,-94.2622119,-744.1946492,-737.1057492
0.0264464,74.1486829,-80.0104165,-744.1858627,-737.1436627
0.0264464,74.1486829,-74.1071301,-744.1829420,-737.1593420
0.0264464,75.6828978,-94.2622119,-744.1954740,-737.0994740
0.0264464,80.0519693,-114.4172937,-744.2077409,-737.0523409
0.0264464,80.0519693,-94.2622119,-744.1980915,-737.0803915
0.0264464,80.0519693,-74.1071301,-744.1858693,-737.1436693
0.0264464,86.5907495,-94.2622119,-744.2024940,-737.0501940
0.0264464,94.3037647,-114.4172937,-744.2175157,-736.9709157
0.0264464,94.3037647,-112.8830788,-744.2172067,-736.9711067
0.0264464,94.3037647,-108.5140073,-744.2158326,-736.9746326
0.0264464,94.3037647,-101.9752272,-744.2126714,-736.9887714
0.0264464,94.3037647,-94.2622119,-744.2078848,-737.0160848
0.0264464,94.3037647,-86.5491967,-744.2024929,-737.0502929
0.0264464,94.3037647,-80.0104165,-744.1980956,-737.0805956
0.0264464,94.3037647,-75.6413450,-744.1954843,-737.0996843
0.0264464,94.3037647,-74.1071301,-744.1946623,-737.1059623
0.0264464,102.0167799,-94.2622119,-744.2126633,-736.9888633
0.0264464,108.5555601,-114.4172937,-744.2267095,-736.9031095
0.0264464,108.5555601,-94.2622119,-744.2158131,-736.9749131
0.0264464,108.5555601,-74.1071301,-744.2027449,-737.0820449
0.0264464,112.9246316,-94.2622119,-744.2171761,-736.9713761
0.0264464,114.4588465,-114.4172937,-744.2287977,-736.8891977
0.0264464,114.4588465,-108.5140073,-744.2266932,-736.9031932
0.0264464,114.4588465,-94.2622119,-744.2174809,-736.9712809
0.0264464,114.4588465,-80.0104165,-744.2077183,-737.0527183
0.0264464,114.4588465,-74.1071301,-744.2045955,-737.0797955
7.7394616,94.3037647,-94.2622119,-744.1968692,-737.0330692
14.2782418,74.1486829,-94.2622119,-744.1530100,-737.1423100
14.2782418,94.3037647,-114.4172937,-744.1841678,-737.0030678
14.2782418,94.3037647,-94.2622119,-744.1701496,-737.0748496
14.2782418,94.3037647,-74.1071301,-744.1568458,-737.1721458
14.2782418,114.4588465,-94.2622119,-744.1761579,-737.0480579
18.6473133,94.3037647,-94.2622119,-744.1430586,-737.1140586
20.1815282,74.1486829,-114.4172937,-744.1285066,-737.1393066
20.1815282,74.1486829,-94.2622119,-744.1126297,-737.1898297
20.1815282,74.1486829,-74.1071301,-744.0997332,-737.2696332
20.1815282,80.0519693,-94.2622119,-744.1189241,-737.1723241
20.1815282,94.3037647,-114.4172937,-744.1488942,-737.0497942
20.1815282,94.3037647,-108.5140073,-744.1445690,-737.0650690
20.1815282,94.3037647,-94.2622119,-744.1317260,-737.1294260
```

```
20.1815282,94.3037647,-80.0104165,-744.1202308,-737.1988308
20.1815282,94.3037647,-74.1071301,-744.1176269,-737.2203269
20.1815282,108.5555601,-94.2622119,-744.1364868,-737.1051868
20.1815282,114.4588465,-114.4172937,-744.1545183,-737.0119183
20.1815282,114.4588465,-94.2622119,-744.1351815,-737.1045815
20.1815282,114.4588465,-74.1071301,-744.1216941,-737.1823941
Path 1
0.0264464,110.4278300,-110.3862770,-736.9075595,0.0000041
0.0264465,110.4278052,-110.3862520,-736.9075596,0.0000041
0.0265357,110.4152905,-110.3736357,-736.9076222,0.0000041
0.0434732,108.5122972,-108.4525738,-736.9175607,0.0000046
0.3114732,99.0309144,-98.7937554,-736.9797642,0.0000011
1.1487851,90.8555402,-90.3187550,-737.0471117,0.0000004
3.6537189,83.1201261,-82.0715180,-737.1147700,0.0000041
8.4504014,78.3670757,-76.6850240,-737.1618668,0.0000016
11.6457299,77.0238933,-75.0021275,-737.1838528,0.0000051



PATCH NUMBER 2
Patch size was 35.213509848
Current state is state 1
Maximum Gaussian iterations was 6
Initial point for the patch was [ 11.6457299   77.02389331 -75.00212746]
Patch terminated at [ 28.29212708  73.21035762 -69.47580674]
Terminal point hit a boundary of the patch, drawing new patch!

Now the Z-matrix, Patch, and Path
Z-matrix for patch 2 was
 C
C,1,B1
H,1,B2,2,A1
H,1,B3,2,A2,3,D1,0
H,1,B4,2,A3,3,D2,0
C,2,B5,1,A4,4,D3,0
H,2,B6,1,A5,6,D4,0
C,6,B7,2,A6,1,D5,0
H,6,B8,2,A7,1,D6,0
H,8,B9,6,A8,2,D7,0
H,8,B10,6,A9,2,D8,0
H,8,B11,6,A10,2,D9,0
Variables:
B1=1.53257957
B2=1.10344567
B3=1.1054505
B4=1.10696393
B5=1.37420478
B6=1.09810856
B7=1.53502319
B8=1.09732867
B9=1.1029387
```

```
B10=1.10760667
B11=1.10512593
A1=111.59487019
A2=112.17594723
A3=110.19830346
A4=124.69807495
A5=116.19695124
A6=124.77674473
A7=117.88105925
A8=110.83909952
A9=112.37632052
A10=111.04681782
D1=121.29152698
D2=-119.83870601
D3=-44.72108401
D4=-166.9276729
D5=20.1815216
D6=-170.93372093
D7=165.65711997
D8=-74.10713
D9=45.5245615


Patch 2
1*2*6*8 6*2*1*5 2*6*8*11 state0 state1
-5.9610250,59.4171384,-78.5234784,-744.1753127,-737.1954127
-5.9610250,59.4171384,-60.9167235,-744.1696816,-737.2053816
-5.9610250,59.4171384,-43.3099686,-744.1711495,-737.1823495
-5.9610250,64.5740372,-60.9167235,-744.1695277,-737.2012277
-5.9610250,77.0238933,-78.5234784,-744.1793896,-737.1490896
-5.9610250,77.0238933,-73.3665797,-744.1763888,-737.1608888
-5.9610250,77.0238933,-60.9167235,-744.1716282,-737.1815282
-5.9610250,77.0238933,-48.4668674,-744.1707558,-737.1844558
-5.9610250,77.0238933,-43.3099686,-744.1718097,-737.1797097
-5.9610250,89.4737494,-60.9167235,-744.1778413,-737.1549413
-5.9610250,94.6306482,-78.5234784,-744.1900398,-737.0877398
-5.9610250,94.6306482,-60.9167235,-744.1811606,-737.1433606
-5.9610250,94.6306482,-43.3099686,-744.1821236,-737.1421236
-4.6207900,77.0238933,-60.9167235,-744.1749511,-737.1789511
-0.8041262,59.4171384,-60.9167235,-744.1772435,-737.1936435
-0.8041262,77.0238933,-78.5234784,-744.1863786,-737.1377786
-0.8041262,77.0238933,-60.9167235,-744.1802446,-737.1775446
-0.8041262,77.0238933,-43.3099686,-744.1819355,-737.1784355
-0.8041262,94.6306482,-60.9167235,-744.1902915,-737.1424915
4.9079171,77.0238933,-60.9167235,-744.1766828,-737.1911828
11.6457299,59.4171384,-78.5234784,-744.1490715,-737.2022715
11.6457299,59.4171384,-73.3665797,-744.1477031,-737.2141031
11.6457299,59.4171384,-60.9167235,-744.1475357,-737.2341357
11.6457299,59.4171384,-48.4668674,-744.1515827,-737.2336827
11.6457299,59.4171384,-43.3099686,-744.1546561,-737.2269561
11.6457299,60.7573734,-60.9167235,-744.1477935,-737.2351935
```

```
11.6457299,64.5740372,-78.5234784,-744.1511432,-737.1979432
11.6457299,64.5740372,-60.9167235,-744.1488021,-737.2364021
11.6457299,64.5740372,-43.3099686,-744.1554500,-737.2309500
11.6457299,70.2860805,-60.9167235,-744.1511743,-737.2346743
11.6457299,77.0238933,-78.5234784,-744.1588854,-737.1762854
11.6457299,77.0238933,-77.1832435,-744.1582525,-737.1813525
11.6457299,77.0238933,-73.3665797,-744.1567521,-737.1949521
11.6457299,77.0238933,-67.6545364,-744.1554179,-737.2129179
11.6457299,77.0238933,-60.9167235,-744.1551216,-737.2268216
11.6457299,77.0238933,-54.1789107,-744.1561117,-737.2311117
11.6457299,77.0238933,-48.4668674,-744.1582830,-737.2267830
11.6457299,77.0238933,-44.6502036,-744.1603664,-737.2200664
11.6457299,77.0238933,-43.3099686,-744.1611893,-737.2170893
11.6457299,83.7617061,-60.9167235,-744.1598752,-737.2133752
11.6457299,89.4737494,-78.5234784,-744.1684890,-737.1466890
11.6457299,89.4737494,-60.9167235,-744.1642328,-737.1988328
11.6457299,89.4737494,-43.3099686,-744.1703995,-737.1772995
11.6457299,93.2904132,-60.9167235,-744.1670718,-737.1879718
11.6457299,94.6306482,-78.5234784,-744.1721428,-737.1342428
11.6457299,94.6306482,-73.3665797,-744.1698162,-737.1554162
11.6457299,94.6306482,-60.9167235,-744.1680264,-737.1841264
11.6457299,94.6306482,-48.4668674,-744.1714586,-737.1724586
11.6457299,94.6306482,-43.3099686,-744.1744089,-737.1565089
18.3835427,77.0238933,-60.9167235,-744.1155219,-737.2766219
24.0955860,59.4171384,-60.9167235,-744.0536596,-737.3390596
24.0955860,77.0238933,-78.5234784,-744.0696238,-737.2836238
24.0955860,77.0238933,-60.9167235,-744.0687265,-737.3195265
24.0955860,77.0238933,-43.3099686,-744.0823828,-737.2890828
24.0955860,94.6306482,-60.9167235,-744.0850913,-737.2629913
27.9122498,77.0238933,-60.9167235,-744.0310591,-737.3496591
29.2524848,59.4171384,-78.5234784,-743.9940273,-737.3577273
29.2524848,59.4171384,-60.9167235,-743.9978488,-737.3848488
29.2524848,59.4171384,-43.3099686,-744.0160814,-737.3623814
29.2524848,64.5740372,-60.9167235,-744.0024581,-737.3833581
29.2524848,77.0238933,-78.5234784,-744.0149560,-737.3354560
29.2524848,77.0238933,-73.3665797,-744.0134391,-737.3496391
29.2524848,77.0238933,-60.9167235,-744.0166430,-737.3601430
29.2524848,77.0238933,-48.4668674,-744.0282786,-737.3394786
29.2524848,77.0238933,-43.3099686,-744.0342800,-737.3239800
29.2524848,89.4737494,-60.9167235,-744.0300452,-737.3194452
29.2524848,94.6306482,-78.5234784,-744.0318071,-737.2869071
29.2524848,94.6306482,-60.9167235,-744.0340778,-737.3012778
29.2524848,94.6306482,-43.3099686,-744.0529843,-737.2530843
Path 2
11.6457299,77.0238933,-75.0021275,-737.1891837,0.0000001
11.6457956,77.0238741,-75.0020916,-737.1891843,0.0000001
11.6816335,77.0134154,-74.9825472,-737.1895101,0.0000001
16.2168775,75.8550450,-72.8539279,-737.2323482,0.0000034
26.6538819,73.5891975,-69.7717321,-737.3375140,0.0000038
28.2921271,73.2103576,-69.4758067,-737.3540783,0.0000039
```

```
PATCH NUMBER 3
Patch size was 33.7237357244
Current state is state 1
Maximum Gaussian iterations was 7
Initial point for the patch was [ 28.29212708  73.21035762 -69.47580674]
Patch terminated at [ 57.68278356  65.69515353 -68.62004029]
Terminal point hit a boundary of the patch, drawing new patch!

Now the Z-matrix, Patch, and Path
Z-matrix for patch 3 was
 C
C,1,B1
H,1,B2,2,A1
H,1,B3,2,A2,3,D1,0
H,1,B4,2,A3,3,D2,0
C,2,B5,1,A4,4,D3,0
H,2,B6,1,A5,6,D4,0
C,6,B7,2,A6,1,D5,0
H,6,B8,2,A7,1,D6,0
H,8,B9,6,A8,2,D7,0
H,8,B10,6,A9,2,D8,0
H,8,B11,6,A10,2,D9,0
Variables:
B1=1.53319955
B2=1.10393204
B3=1.104974
B4=1.10675901
B5=1.37503825
B6=1.09927422
B7=1.53745679
B8=1.09814884
B9=1.10302603
B10=1.10738671
B11=1.10467511
A1=111.82444282
A2=112.31518015
A3=109.60884756
A4=125.00283563
A5=115.31112866
A6=125.1198104
A7=117.24333882
A8=110.61631361
A9=112.83661087
A10=110.78194259
D1=121.61080658
D2=-119.66877226
D3=-41.69652816
D4=-161.64985245
D5=29.25247076
```

```
D6=-167.20756639
D7=166.39220731
D8=-73.36658
D9=46.57882555


Patch 3
1*2*6*8 6*2*1*5 2*6*8*11 state0 state1
24.9197535,56.3484898,-86.3376746,-744.0452186,-737.2881186
24.9197535,56.3484898,-69.4758067,-744.0407020,-737.3358020
24.9197535,56.3484898,-52.6139389,-744.0499642,-737.3412642
24.9197535,61.2872162,-69.4758067,-744.0439245,-737.3365245
24.9197535,73.2103576,-86.3376746,-744.0621848,-737.2688848
24.9197535,73.2103576,-81.3989482,-744.0588321,-737.2887321
24.9197535,73.2103576,-69.4758067,-744.0551699,-737.3259699
24.9197535,73.2103576,-57.5526653,-744.0588541,-737.3342541
24.9197535,73.2103576,-52.6139389,-744.0625148,-737.3290148
24.9197535,85.1334990,-69.4758067,-744.0677353,-737.2963353
24.9197535,90.0722255,-86.3376746,-744.0798496,-737.2260496
24.9197535,90.0722255,-69.4758067,-744.0722128,-737.2811128
24.9197535,90.0722255,-52.6139389,-744.0796419,-737.2728419
26.2032873,73.2103576,-69.4758067,-744.0421336,-737.3377336
29.8584799,56.3484898,-69.4758067,-743.9839134,-737.3825134
29.8584799,73.2103576,-86.3376746,-744.0079150,-737.3212150
29.8584799,73.2103576,-69.4758067,-744.0017728,-737.3705728
29.8584799,73.2103576,-52.6139389,-744.0127130,-737.3653130
29.8584799,90.0722255,-69.4758067,-744.0203093,-737.3223093
35.3288644,73.2103576,-69.4758067,-743.9326902,-737.4196902
41.7816214,56.3484898,-86.3376746,-743.8101049,-737.4656049
41.7816214,56.3484898,-81.3989482,-743.8083230,-737.4772230
41.7816214,56.3484898,-69.4758067,-743.8122110,-737.4936110
41.7816214,56.3484898,-57.5526653,-743.8263473,-737.4908473
41.7816214,56.3484898,-52.6139389,-743.8339041,-737.4841041
41.7816214,57.6320236,-69.4758067,-743.8139329,-737.4935329
41.7816214,61.2872162,-86.3376746,-743.8180065,-737.4638065
41.7816214,61.2872162,-69.4758067,-743.8192521,-737.4918521
41.7816214,61.2872162,-52.6139389,-743.8405633,-737.4812633
41.7816214,66.7576007,-69.4758067,-743.8278846,-737.4852846
41.7816214,73.2103576,-86.3376746,-743.8381305,-737.4457305
41.7816214,73.2103576,-85.0541408,-743.8372687,-737.4490687
41.7816214,73.2103576,-81.3989482,-743.8355802,-737.4575802
41.7816214,73.2103576,-75.9285637,-743.8352968,-737.4670968
41.7816214,73.2103576,-69.4758067,-743.8382929,-737.4718929
41.7816214,73.2103576,-63.0230498,-743.8444909,-737.4698909
41.7816214,73.2103576,-57.5526653,-743.8518788,-737.4643788
41.7816214,73.2103576,-53.8974727,-743.8574714,-737.4583714
41.7816214,73.2103576,-52.6139389,-743.8595014,-737.4558014
41.7816214,79.6631145,-69.4758067,-743.8476846,-737.4539846
41.7816214,85.1334990,-86.3376746,-743.8537632,-737.4146632
41.7816214,85.1334990,-69.4758067,-743.8542539,-737.4371539
41.7816214,85.1334990,-52.6139389,-743.8764064,-737.4177064
```

```
41.7816214,88.7886917,-69.4758067,-743.8576707,-737.4258707
41.7816214,90.0722255,-86.3376746,-743.8576750,-737.4013750
41.7816214,90.0722255,-81.3989482,-743.8553247,-737.4125247
41.7816214,90.0722255,-69.4758067,-743.8586482,-737.4219482
41.7816214,90.0722255,-57.5526653,-743.8733372,-737.4115372
41.7816214,90.0722255,-52.6139389,-743.8814710,-737.4012710
48.2343783,73.2103576,-69.4758067,-743.7304298,-737.5202298
53.7047628,56.3484898,-69.4758067,-743.5968447,-737.5822447
53.7047628,73.2103576,-86.3376746,-743.6176540,-737.5475540
53.7047628,73.2103576,-69.4758067,-743.6287107,-737.5583107
53.7047628,73.2103576,-52.6139389,-743.6594435,-737.5432435
53.7047628,90.0722255,-69.4758067,-743.6475692,-737.5185692
57.3599554,73.2103576,-69.4758067,-743.5555712,-737.5820712
58.6434892,56.3484898,-86.3376746,-743.4795674,-737.6032674
58.6434892,56.3484898,-69.4758067,-743.4956629,-737.6116629
58.6434892,56.3484898,-52.6139389,-743.5282914,-737.6006914
58.6434892,61.2872162,-69.4758067,-743.5057651,-737.6080651
58.6434892,73.2103576,-86.3376746,-743.5128187,-737.5830187
58.6434892,73.2103576,-81.3989482,-743.5144355,-737.5872355
58.6434892,73.2103576,-69.4758067,-743.5289149,-737.5901149
58.6434892,73.2103576,-57.5526653,-743.5523918,-737.5821918
58.6434892,73.2103576,-52.6139389,-743.5626873,-737.5769873
58.6434892,85.1334990,-69.4758067,-743.5436107,-737.5657107
58.6434892,90.0722255,-86.3376746,-743.5273855,-737.5495855
58.6434892,90.0722255,-69.4758067,-743.5459539,-737.5564539
58.6434892,90.0722255,-52.6139389,-743.5817090,-737.5439090
Path 3
28.2921271,73.2103576,-69.4758067,-737.3564952,0.0000001
28.2922172,73.2103362,-69.4757930,-737.3564961,0.0000001
28.3317639,73.2009516,-69.4697579,-737.3568810,0.0000001
33.1086610,72.0351954,-68.8878871,-737.4025352,0.0000008
52.1554787,67.0751017,-68.4753557,-737.5601028,0.0000002
55.6148539,66.1996347,-68.5646880,-737.5837213,0.0000000
57.6827836,65.6951535,-68.6200403,-737.5971667,0.0000003



PATCH NUMBER 4
Patch size was 54.3519535599
Current state is state 1
Maximum Gaussian iterations was 10
Initial point for the patch was [ 57.68278356  65.69515353 -68.62004029]
Patch terminated at [ 101.16631937   74.27748311  -41.86523603]
Terminal point was a minimum, changing states!

Now the Z-matrix, Patch, and Path
Z-matrix for patch 4 was
 C
C,1,B1
H,1,B2,2,A1
H,1,B3,2,A2,3,D1,0
```

```
H,1,B4,2,A3,3,D2,0
C,2,B5,1,A4,4,D3,0
H,2,B6,1,A5,6,D4,0
C,6,B7,2,A6,1,D5,0
H,6,B8,2,A7,1,D6,0
H,8,B9,6,A8,2,D7,0
H,8,B10,6,A9,2,D8,0
H,8,B11,6,A10,2,D9,0
Variables:
B1=1.54128666
B2=1.10313363
B3=1.10666604
B4=1.10457706
B5=1.38416237
B6=1.10390879
B7=1.54401595
B8=1.10316813
B9=1.10355139
B10=1.10690602
B11=1.1032948
A1=111.37985854
A2=113.44686351
A3=108.62848978
A4=126.90047049
A5=111.18253485
A6=126.98394483
A7=113.78989835
A8=110.16757552
A9=114.055218
A10=109.52300643
D1=121.82948157
D2=-119.67998307
D3=-57.20331937
D4=-146.36314157
D5=58.64347464
D6=-155.10111992
D7=170.34739484
D8=-69.475807
D9=51.52986646


Patch 4
1*2*6*8 6*2*1*5 2*6*8*11 state0 state1
52.2475882,38.5191768,-95.7960171,-743.5948218,-737.5238218
52.2475882,38.5191768,-68.6200403,-743.6060123,-737.5592123
52.2475882,38.5191768,-41.4440635,-743.6536578,-737.5301578
52.2475882,46.4788356,-68.6200403,-743.6119081,-737.5699081
52.2475882,65.6951535,-95.7960171,-743.6403288,-737.5308288
52.2475882,65.6951535,-87.8363583,-743.6339131,-737.5482131
52.2475882,65.6951535,-68.6200403,-743.6441354,-737.5634354
52.2475882,65.6951535,-49.4037223,-743.6781776,-737.5436776
```

```
52.2475882,65.6951535,-41.4440635,-743.6915202,-737.5284202
52.2475882,84.9114715,-68.6200403,-743.6736452,-737.5195452
52.2475882,92.8711303,-95.7960171,-743.6713487,-737.4744487
52.2475882,92.8711303,-68.6200403,-743.6782544,-737.5009544
52.2475882,92.8711303,-41.4440635,-743.7301941,-737.4624941
54.3162371,65.6951535,-68.6200403,-743.6041151,-737.5769151
60.2072470,38.5191768,-68.6200403,-743.4362375,-737.6130375
60.2072470,65.6951535,-95.7960171,-743.4682165,-737.5924165
60.2072470,65.6951535,-68.6200403,-743.4832365,-737.6130365
60.2072470,65.6951535,-41.4440635,-743.5350336,-737.5853336
60.2072470,92.8711303,-68.6200403,-743.5138477,-737.5635477
69.0237698,65.6951535,-68.6200403,-743.2835803,-737.6551803
79.4235650,38.5191768,-95.7960171,-742.9171271,-737.6796271
79.4235650,38.5191768,-87.8363583,-742.9229322,-737.6819322
79.4235650,38.5191768,-68.6200403,-742.9604061,-737.6870061
79.4235650,38.5191768,-49.4037223,-742.9984668,-737.6910668
79.4235650,38.5191768,-41.4440635,-743.0060524,-737.6890524
79.4235650,40.5878257,-68.6200403,-742.9642681,-737.6883681
79.4235650,46.4788356,-95.7960171,-742.9349086,-737.6843086
79.4235650,46.4788356,-68.6200403,-742.9768543,-737.6915543
79.4235650,46.4788356,-41.4440635,-743.0235367,-737.6930367
79.4235650,55.2953583,-68.6200403,-742.9978368,-737.6938368
79.4235650,65.6951535,-95.7960171,-742.9778318,-737.6864318
79.4235650,65.6951535,-93.7273681,-742.9784106,-737.6871106
79.4235650,65.6951535,-87.8363583,-742.9829739,-737.6885739
79.4235650,65.6951535,-79.0198355,-742.9970537,-737.6901537
79.4235650,65.6951535,-68.6200403,-743.0203085,-737.6919085
79.4235650,65.6951535,-58.2202451,-743.0443341,-737.6954341
79.4235650,65.6951535,-49.4037223,-743.0604109,-737.6967109
79.4235650,65.6951535,-43.5127125,-743.0674860,-737.6958860
79.4235650,65.6951535,-41.4440635,-743.0691117,-737.6952117
79.4235650,76.0949487,-68.6200403,-743.0344848,-737.6861848
79.4235650,84.9114715,-95.7960171,-742.9906921,-737.6743921
79.4235650,84.9114715,-68.6200403,-743.0373674,-737.6795674
79.4235650,84.9114715,-41.4440635,-743.0863569,-737.6846569
79.4235650,90.8024814,-68.6200403,-743.0342673,-737.6748673
79.4235650,92.8711303,-95.7960171,-742.9836356,-737.6680356
79.4235650,92.8711303,-87.8363583,-742.9908632,-737.6693632
79.4235650,92.8711303,-68.6200403,-743.0322547,-737.6732547
79.4235650,92.8711303,-49.4037223,-743.0728177,-737.6802177
79.4235650,92.8711303,-41.4440635,-743.0805338,-737.6797338
89.8233602,65.6951535,-68.6200403,-742.7287972,-737.7108972
98.6398830,38.5191768,-68.6200403,-742.8616506,-737.8217506
98.6398830,65.6951535,-95.7960171,-742.8157790,-737.8027790
98.6398830,65.6951535,-68.6200403,-742.8092670,-737.8410670
98.6398830,65.6951535,-41.4440635,-742.7356079,-737.8583079
98.6398830,92.8711303,-68.6200403,-742.7544088,-737.8407088
104.5308928,65.6951535,-68.6200403,-742.9868467,-737.8202467
106.5995418,38.5191768,-95.7960171,-743.1196991,-737.7507991
106.5995418,38.5191768,-68.6200403,-743.1044312,-737.7831312
106.5995418,38.5191768,-41.4440635,-743.0340545,-737.8044545
```

```
106.5995418,46.4788356,-68.6200403,-743.0931552,-737.7947552
106.5995418,65.6951535,-95.7960171,-743.0632282,-737.7696282
106.5995418,65.6951535,-87.8363583,-743.0685889,-737.7789889
106.5995418,65.6951535,-68.6200403,-743.0467159,-737.8125159
106.5995418,65.6951535,-49.4037223,-742.9938814,-737.8343814
106.5995418,65.6951535,-41.4440635,-742.9741616,-737.8331616
106.5995418,84.9114715,-68.6200403,-743.0064140,-737.8138140
106.5995418,92.8711303,-95.7960171,-743.0152165,-737.7715165
106.5995418,92.8711303,-68.6200403,-743.0012280,-737.8085280
106.5995418,92.8711303,-41.4440635,-742.9264104,-737.8239104
Path 4
57.6827836,65.6951535,-68.6200403,-737.6028194,0.0000015
57.6828313,65.6951434,-68.6200400,-737.6028196,0.0000015
57.7037450,65.6906816,-68.6198949,-737.6029239,0.0000014
60.1597851,65.1055487,-68.6003047,-737.6134160,0.0000003
63.3836890,64.2061584,-68.5611737,-737.6236831,0.0000028
66.8270731,63.3154420,-68.4941200,-737.6332698,0.0000134
74.8256261,62.2547086,-68.2669524,-737.6643239,0.0000350
80.1232346,62.0150358,-68.0829522,-737.6982873,0.0000085
87.0131083,61.9148995,-67.7810924,-737.7567064,0.0000906
94.6185574,61.9894388,-67.2294750,-737.8210244,0.0000495
97.7169619,62.1195300,-66.7732695,-737.8386459,0.0000080
99.5352339,62.3055254,-66.2049553,-737.8446913,0.0000061
100.2307113,62.4732990,-65.7124255,-737.8461769,0.0000081
100.5760158,62.6554938,-65.1801390,-737.8469751,0.0000085
100.7448001,62.8570924,-64.5861014,-737.8476655,0.0000088
100.8280432,63.0852689,-63.9027608,-737.8484117,0.0000091
100.8738532,63.3488133,-63.0965299,-737.8492759,0.0000096
100.9070668,63.6601466,-62.1198930,-737.8503086,0.0000101
100.9392727,64.0362778,-60.9062855,-737.8515700,0.0000107
101.1663194,74.2774831,-41.8652360,-737.8644362,0.0000062



PATCH NUMBER 5
Patch size was 14.3008509726
Current state is state 0
Maximum Gaussian iterations was 11
Initial point for the patch was [ 101.16631937   74.27748311  -41.86523603]
Patch terminated at [ 107.96257975   73.72968385  -42.41160991]
Terminal point hit a boundary of the patch, drawing new patch!

Now the Z-matrix, Patch, and Path
Z-matrix for patch 5 was
 C
C,1,B1
H,1,B2,2,A1
H,1,B3,2,A2,3,D1,0
H,1,B4,2,A3,3,D2,0
C,2,B5,1,A4,4,D3,0
H,2,B6,1,A5,6,D4,0
```

```
C,6,B7,2,A6,1,D5,0
H,6,B8,2,A7,1,D6,0
H,8,B9,6,A8,2,D7,0
H,8,B10,6,A9,2,D8,0
H,8,B11,6,A10,2,D9,0
Variables:
B1=1.54283584
B2=1.1035385
B3=1.10563068
B4=1.10683613
B5=1.38470232
B6=1.11031082
B7=1.54555825
B8=1.10921133
B9=1.10146904
B10=1.1051482
B11=1.10842989
A1=110.42758828
A2=107.68323669
A3=114.94147169
A4=127.06255099
A5=108.44976778
A6=127.09478679
A7=112.71293055
A8=109.9941327
A9=108.42242655
A10=115.08561294
D1=118.18491632
D2=-120.99232902
D3=-55.12760066
D4=139.51153207
D5=98.6399062
D6=-40.8886822
D7=-161.18350351
D8=-41.444064
D9=77.22277821
```

```
Patch 5
1*2*6*8 6*2*1*5 2*6*8*11 state0 state1
94.0158939,67.1270576,-49.0156615,-742.6068907,-737.8664907
94.0158939,67.1270576,-41.8652360,-742.5856713,-737.8674713
94.0158939,67.1270576,-34.7148105,-742.5694012,-737.8629012
94.0158939,69.2213686,-41.8652360,-742.5796190,-737.8677190
94.0158939,74.2774831,-49.0156615,-742.5869474,-737.8677474
94.0158939,74.2774831,-46.9213505,-742.5804340,-737.8682340
94.0158939,74.2774831,-41.8652360,-742.5652688,-737.8681688
94.0158939,74.2774831,-36.8091215,-742.5524470,-737.8645470
94.0158939,74.2774831,-34.7148105,-742.5483260,-737.8623260
94.0158939,79.3335976,-41.8652360,-742.5521785,-737.8672785
94.0158939,81.4279086,-49.0156615,-742.5682611,-737.8663611
```

```
94.0158939,81.4279086,-41.8652360,-742.5471966,-737.8663966
94.0158939,81.4279086,-34.7148105,-742.5291226,-737.8640226
94.5601879,74.2774831,-41.8652360,-742.5826623,-737.8670623
96.1102049,67.1270576,-41.8652360,-742.6532323,-737.8657323
96.1102049,74.2774831,-49.0156615,-742.6529333,-737.8646333
96.1102049,74.2774831,-41.8652360,-742.6326881,-737.8653881
96.1102049,74.2774831,-34.7148105,-742.6160865,-737.8590865
96.1102049,81.4279086,-41.8652360,-742.6149585,-737.8632585
98.4299702,74.2774831,-41.8652360,-742.7057935,-737.8594935
101.1663194,67.1270576,-49.0156615,-742.8291077,-737.8527077
101.1663194,67.1270576,-46.9213505,-742.8229848,-737.8530848
101.1663194,67.1270576,-41.8652360,-742.8094127,-737.8527127
101.1663194,67.1270576,-36.8091215,-742.7991334,-737.8490334
101.1663194,67.1270576,-34.7148105,-742.7960714,-737.8463714
101.1663194,67.6713516,-41.8652360,-742.8078821,-737.8527821
101.1663194,69.2213686,-49.0156615,-742.8234236,-737.8532236
101.1663194,69.2213686,-41.8652360,-742.8035665,-737.8530665
101.1663194,69.2213686,-34.7148105,-742.7899976,-737.8463976
101.1663194,71.5411340,-41.8652360,-742.7972863,-737.8532863
101.1663194,74.2774831,-49.0156615,-742.8103397,-737.8537397
101.1663194,74.2774831,-48.4713675,-742.8087345,-737.8538345
101.1663194,74.2774831,-46.9213505,-742.8042156,-737.8540156
101.1663194,74.2774831,-44.6015852,-742.7976228,-737.8539228
101.1663194,74.2774831,-41.8652360,-742.7902856,-737.8532856
101.1663194,74.2774831,-39.1288869,-742.7838116,-737.8522116
101.1663194,74.2774831,-36.8091215,-742.7791469,-737.8504469
101.1663194,74.2774831,-35.2591045,-742.7765358,-737.8479358
101.1663194,74.2774831,-34.7148105,-742.7757555,-737.8470555
101.1663194,77.0138323,-41.8652360,-742.7837719,-737.8526719
101.1663194,79.3335976,-49.0156615,-742.7983058,-737.8525058
101.1663194,79.3335976,-41.8652360,-742.7785708,-737.8517708
101.1663194,79.3335976,-34.7148105,-742.7634916,-737.8478916
101.1663194,80.8836146,-41.8652360,-742.7752873,-737.8511873
101.1663194,81.4279086,-49.0156615,-742.7938370,-737.8519370
101.1663194,81.4279086,-46.9213505,-742.7877615,-737.8521615
101.1663194,81.4279086,-41.8652360,-742.7741825,-737.8509825
101.1663194,81.4279086,-36.8091215,-742.7628678,-737.8482678
101.1663194,81.4279086,-34.7148105,-742.7591776,-737.8468776
103.9026685,74.2774831,-41.8652360,-742.8732098,-737.8444098
106.2224339,67.1270576,-41.8652360,-742.9601928,-737.8356928
106.2224339,74.2774831,-49.0156615,-742.9600096,-737.8377096
106.2224339,74.2774831,-41.8652360,-742.9418152,-737.8362152
106.2224339,74.2774831,-34.7148105,-742.9294102,-737.8290102
106.2224339,81.4279086,-41.8652360,-742.9269733,-737.8338733
107.7724509,74.2774831,-41.8652360,-742.9868172,-737.8300172
108.3167449,67.1270576,-49.0156615,-743.0376211,-737.8290211
108.3167449,67.1270576,-41.8652360,-743.0204515,-737.8273515
108.3167449,67.1270576,-34.7148105,-743.0093371,-737.8186371
108.3167449,69.2213686,-41.8652360,-743.0149004,-737.8278004
108.3167449,74.2774831,-49.0156615,-743.0199149,-737.8298149
108.3167449,74.2774831,-46.9213505,-743.0143610,-737.8297610
```

```
108.3167449,74.2774831,-41.8652360,-743.0024618,-737.8277618
108.3167449,74.2774831,-36.8091215,-742.9936761,-737.8228761
108.3167449,74.2774831,-34.7148105,-742.9910843,-737.8194843
108.3167449,79.3335976,-41.8652360,-742.9919103,-737.8262103
108.3167449,81.4279086,-49.0156615,-743.0060328,-737.8276328
108.3167449,81.4279086,-41.8652360,-742.9883138,-737.8253138
108.3167449,81.4279086,-34.7148105,-742.9766473,-737.8177473
Path 5
101.1663194,74.2774831,-41.8652360,-742.7902856,0.0000000
101.1666251,74.2774583,-41.8652614,-742.7902951,0.0000000
101.2932515,74.2671865,-41.8757769,-742.7942161,0.0000000
105.0359129,73.9651507,-42.1811319,-742.9083253,0.0000001
107.3755771,73.7769921,-42.3659977,-742.9776640,0.0000001
107.9625797,73.7296838,-42.4116099,-742.9947675,0.0000001



PATCH NUMBER 6
Patch size was 7.15042548628
Current state is state 0
Maximum Gaussian iterations was 7
Initial point for the patch was [ 107.96257975   73.72968385  -42.41160991]
Patch terminated at [ 114.36577164   73.21287474  -42.87115485]
Terminal point hit a boundary of the patch, drawing new patch!

Now the Z-matrix, Patch, and Path
Z-matrix for patch 6 was
 C
C,1,B1
H,1,B2,2,A1
H,1,B3,2,A2,3,D1,0
H,1,B4,2,A3,3,D2,0
C,2,B5,1,A4,4,D3,0
H,2,B6,1,A5,6,D4,0
C,6,B7,2,A6,1,D5,0
H,6,B8,2,A7,1,D6,0
H,8,B9,6,A8,2,D7,0
H,8,B10,6,A9,2,D8,0
H,8,B11,6,A10,2,D9,0
Variables:
B1=1.54405258
B2=1.10323068
B3=1.10424895
B4=1.1074571
B5=1.38224916
B6=1.10734342
B7=1.54442397
B8=1.10700776
B9=1.10185864
B10=1.10446728
B11=1.10843202
```

```
A1=109.68971798
A2=108.9507749
A3=114.93782687
A4=126.60256995
A5=110.24314952
A6=126.63655044
A7=113.7928059
A8=110.01114528
A9=108.98109542
A10=114.72401534
D1=118.59495382
D2=-120.10816839
D3=-47.01939479
D4=143.90149994
D5=107.77247046
D6=-35.74618583
D7=-161.65635892
D8=-41.865236
D9=77.16752555


Patch 6
1*2*6*8 6*2*1*5 2*6*8*11 state0 state1
107.2475372,70.1544711,-45.9868227,-742.9914923,-737.8337923
107.2475372,70.1544711,-42.4116099,-742.9829971,-737.8324971
107.2475372,70.1544711,-38.8363972,-742.9760074,-737.8298074
107.2475372,71.2016266,-42.4116099,-742.9803234,-737.8326234
107.2475372,73.7296838,-45.9868227,-742.9826994,-737.8337994
107.2475372,73.7296838,-44.9396672,-742.9800649,-737.8335649
107.2475372,73.7296838,-42.4116099,-742.9741336,-737.8324336
107.2475372,73.7296838,-39.8835527,-742.9689543,-737.8306543
107.2475372,73.7296838,-38.8363972,-742.9670577,-737.8297577
107.2475372,76.2577411,-42.4116099,-742.9683675,-737.8319675
107.2475372,77.3048966,-45.9868227,-742.9747466,-737.8331466
107.2475372,77.3048966,-42.4116099,-742.9661243,-737.8316243
107.2475372,77.3048966,-38.8363972,-742.9589772,-737.8288772
107.5196842,73.7296838,-42.4116099,-742.9819913,-737.8313913
108.2946927,70.1544711,-42.4116099,-743.0130077,-737.8282077
108.2946927,73.7296838,-45.9868227,-743.0126141,-737.8297141
108.2946927,73.7296838,-42.4116099,-743.0042553,-737.8281553
108.2946927,73.7296838,-38.8363972,-742.9974212,-737.8253212
108.2946927,77.3048966,-42.4116099,-742.9963932,-737.8273932
109.4545754,73.7296838,-42.4116099,-743.0372632,-737.8231632
110.8227499,70.1544711,-45.9868227,-743.0919417,-737.8187417
110.8227499,70.1544711,-44.9396672,-743.0895211,-737.8184211
110.8227499,70.1544711,-42.4116099,-743.0841712,-737.8169712
110.8227499,70.1544711,-39.8835527,-743.0796237,-737.8147237
110.8227499,70.1544711,-38.8363972,-743.0779935,-737.8135935
110.8227499,70.4266181,-42.4116099,-743.0834984,-737.8169984
110.8227499,71.2016266,-45.9868227,-743.0894023,-737.8188023
110.8227499,71.2016266,-42.4116099,-743.0816075,-737.8170075
```

```
110.8227499,71.2016266,-38.8363972,-743.0754053,-737.8136053
110.8227499,72.3615093,-42.4116099,-743.0788496,-737.8169496
110.8227499,73.7296838,-45.9868227,-743.0835661,-737.8186661
110.8227499,73.7296838,-45.7146757,-743.0829211,-737.8186211
110.8227499,73.7296838,-44.9396672,-743.0811236,-737.8183236
110.8227499,73.7296838,-43.7797845,-743.0785507,-737.8177507
110.8227499,73.7296838,-42.4116099,-743.0757157,-737.8168157
110.8227499,73.7296838,-41.0434353,-743.0731173,-737.8157173
110.8227499,73.7296838,-39.8835527,-743.0711105,-737.8146105
110.8227499,73.7296838,-39.1085441,-743.0698729,-737.8137729
110.8227499,73.7296838,-38.8363972,-743.0694582,-737.8134582
110.8227499,75.0978584,-42.4116099,-743.0727238,-737.8166238
110.8227499,76.2577411,-45.9868227,-743.0782188,-737.8182188
110.8227499,76.2577411,-42.4116099,-743.0703118,-737.8163118
110.8227499,76.2577411,-38.8363972,-743.0639973,-737.8128973
110.8227499,77.0327496,-42.4116099,-743.0687702,-737.8160702
110.8227499,77.3048966,-45.9868227,-743.0761753,-737.8178753
110.8227499,77.3048966,-44.9396672,-743.0737109,-737.8175109
110.8227499,77.3048966,-42.4116099,-743.0682431,-737.8160431
110.8227499,77.3048966,-39.8835527,-743.0635769,-737.8137769
110.8227499,77.3048966,-38.8363972,-743.0619020,-737.8126020
112.1909245,73.7296838,-42.4116099,-743.1136419,-737.8102419
113.3508072,70.1544711,-42.4116099,-743.1535021,-737.8038021
113.3508072,73.7296838,-45.9868227,-743.1527104,-737.8060104
113.3508072,73.7296838,-42.4116099,-743.1453787,-737.8037787
113.3508072,73.7296838,-38.8363972,-743.1396868,-737.7999868
113.3508072,77.3048966,-42.4116099,-743.1383293,-737.8029293
114.1258157,73.7296838,-42.4116099,-743.1663703,-737.7997703
114.3979627,70.1544711,-45.9868227,-743.1887074,-737.8007074
114.3979627,70.1544711,-42.4116099,-743.1816776,-737.7983776
114.3979627,70.1544711,-38.8363972,-743.1762996,-737.7942996
114.3979627,71.2016266,-42.4116099,-743.1792426,-737.7984426
114.3979627,73.7296838,-45.9868227,-743.1808158,-737.8006158
114.3979627,73.7296838,-44.9396672,-743.1785765,-737.8000765
114.3979627,73.7296838,-42.4116099,-743.1737004,-737.7983004
114.3979627,73.7296838,-39.8835527,-743.1696556,-737.7955556
114.3979627,73.7296838,-38.8363972,-743.1682396,-737.7942396
114.3979627,76.2577411,-42.4116099,-743.1687111,-737.7977111
114.3979627,77.3048966,-45.9868227,-743.1740400,-737.7997400
114.3979627,77.3048966,-42.4116099,-743.1668322,-737.7974322
114.3979627,77.3048966,-38.8363972,-743.1612795,-737.7933795
Path 6
107.9625798,73.7296839,-42.4116099,-742.9947350,0.0000000
107.9628669,73.7296604,-42.4116315,-742.9947434,0.0000000
108.0579578,73.7218858,-42.4187784,-742.9975060,0.0000000
113.5024612,73.2815728,-42.8118379,-743.1512027,0.0000000
114.3657716,73.2128747,-42.8711548,-743.1747544,0.0000000


PATCH NUMBER 7
```

```
Patch size was 60.0
Current state is state 0
Maximum Gaussian iterations was 9
Initial point for the patch was [ 114.36577164   73.21287474  -42.87115485]
Patch terminated at [ 167.32685802   70.31509819  -43.62247643]
Terminal point hit a boundary of the patch, drawing new patch!

Now the Z-matrix, Patch, and Path
Z-matrix for patch 7 was
 C
C,1,B1
H,1,B2,2,A1
H,1,B3,2,A2,3,D1,0
H,1,B4,2,A3,3,D2,0
C,2,B5,1,A4,4,D3,0
H,2,B6,1,A5,6,D4,0
C,6,B7,2,A6,1,D5,0
H,6,B8,2,A7,1,D6,0
H,8,B9,6,A8,2,D7,0
H,8,B10,6,A9,2,D8,0
H,8,B11,6,A10,2,D9,0
Variables:
B1=1.54295879
B2=1.10333148
B3=1.10399057
B4=1.10746019
B5=1.38040069
B6=1.10575781
B7=1.54326837
B8=1.10544014
B9=1.10214411
B10=1.10413081
B11=1.10832557
A1=109.80353398
A2=109.28331735
A3=114.63345515
A4=126.24812413
A5=111.30407217
A6=126.27210677
A7=114.58514255
A8=110.05858034
A9=109.32114484
A10=114.45882952
D1=118.76441352
D2=-120.00708399
D3=-47.4988185
D4=146.90968945
D5=114.39797913
D6=-32.35080126
D7=-162.20816234
D8=-42.41161
```

```
D9=76.86756189


Patch 7
1*2*6*8 6*2*1*5 2*6*8*11 state0 state1
108.3657716,43.2128747,-72.8711548,-743.1579132,-737.7739132
108.3657716,43.2128747,-42.8711548,-743.0821266,-737.8047266
108.3657716,43.2128747,-12.8711548,-743.0857540,-737.7592540
108.3657716,51.9996707,-42.8711548,-743.0642263,-737.8168263
108.3657716,73.2128747,-72.8711548,-743.0872263,-737.8003263
108.3657716,73.2128747,-64.0843588,-743.0667226,-737.8157226
108.3657716,73.2128747,-42.8711548,-743.0084969,-737.8281969
108.3657716,73.2128747,-21.6579508,-742.9961687,-737.7921687
108.3657716,73.2128747,-12.8711548,-743.0120620,-737.7707620
108.3657716,94.4260787,-42.8711548,-742.9827165,-737.8144165
108.3657716,103.2128747,-72.8711548,-743.0705876,-737.7811876
108.3657716,103.2128747,-42.8711548,-742.9905460,-737.8017460
108.3657716,103.2128747,-12.8711548,-742.9879087,-737.7599087
110.6493866,73.2128747,-42.8711548,-743.0729736,-737.8180736
117.1525676,43.2128747,-42.8711548,-743.3215601,-737.7556601
117.1525676,73.2128747,-72.8711548,-743.3239717,-737.7546717
117.1525676,73.2128747,-42.8711548,-743.2484576,-737.7831576
117.1525676,73.2128747,-12.8711548,-743.2661012,-737.7174012
117.1525676,103.2128747,-42.8711548,-743.2445444,-737.7488444
126.8852696,73.2128747,-42.8711548,-743.4873564,-737.7199564
138.3657716,43.2128747,-72.8711548,-743.8448080,-737.5705080
138.3657716,43.2128747,-64.0843588,-743.8232225,-737.5941225
138.3657716,43.2128747,-42.8711548,-743.7912581,-737.5907581
138.3657716,43.2128747,-21.6579508,-743.8148731,-737.5262731
138.3657716,43.2128747,-12.8711548,-743.8356662,-737.4968662
138.3657716,45.4964897,-42.8711548,-743.7858627,-737.5962627
138.3657716,51.9996707,-72.8711548,-743.8241010,-737.5920010
138.3657716,51.9996707,-42.8711548,-743.7698920,-737.6096920
138.3657716,51.9996707,-12.8711548,-743.8144367,-737.5201367
138.3657716,61.7323727,-42.8711548,-743.7471787,-737.6220787
138.3657716,73.2128747,-72.8711548,-743.7866208,-737.6069208
138.3657716,73.2128747,-70.5875398,-743.7808391,-737.6133391
138.3657716,73.2128747,-64.0843588,-743.7642267,-737.6279267
138.3657716,73.2128747,-54.3516568,-743.7424428,-737.6363428
138.3657716,73.2128747,-42.8711548,-743.7291734,-737.6249734
138.3657716,73.2128747,-31.3906528,-743.7342518,-737.5951518
138.3657716,73.2128747,-21.6579508,-743.7514572,-737.5656572
138.3657716,73.2128747,-15.1547698,-743.7669532,-737.5469532
138.3657716,73.2128747,-12.8711548,-743.7727543,-737.5392543
138.3657716,84.6933767,-42.8711548,-743.7266859,-737.6126859
138.3657716,94.4260787,-72.8711548,-743.7973872,-737.5674872
138.3657716,94.4260787,-42.8711548,-743.7375727,-737.5910727
138.3657716,94.4260787,-12.8711548,-743.7791563,-737.5072563
138.3657716,100.9292597,-42.8711548,-743.7502070,-737.5745070
138.3657716,103.2128747,-72.8711548,-743.8150197,-737.5419197
138.3657716,103.2128747,-64.0843588,-743.7924775,-737.5656775
```

```
138.3657716,103.2128747,-42.8711548,-743.7553122,-737.5685122
138.3657716,103.2128747,-21.6579508,-743.7752171,-737.5096171
138.3657716,103.2128747,-12.8711548,-743.7964505,-737.4828505
149.8462736,73.2128747,-42.8711548,-743.9231894,-737.5240894
159.5789756,43.2128747,-42.8711548,-744.0858760,-737.4118760
159.5789756,73.2128747,-72.8711548,-744.0752480,-737.4458480
159.5789756,73.2128747,-42.8711548,-744.0469173,-737.4464173
159.5789756,73.2128747,-12.8711548,-744.1031768,-737.3509768
159.5789756,103.2128747,-42.8711548,-744.0942928,-737.3633928
166.0821566,73.2128747,-42.8711548,-744.1073808,-737.4046808
168.3657716,43.2128747,-72.8711548,-744.1597997,-737.3796997
168.3657716,43.2128747,-42.8711548,-744.1506887,-737.3599887
168.3657716,43.2128747,-12.8711548,-744.2080092,-737.2513092
168.3657716,51.9996707,-42.8711548,-744.1343749,-737.3823749
168.3657716,73.2128747,-72.8711548,-744.1387039,-737.3999039
168.3657716,73.2128747,-64.0843588,-744.1245775,-737.4147775
168.3657716,73.2128747,-42.8711548,-744.1242215,-737.3928215
168.3657716,73.2128747,-21.6579508,-744.1643500,-737.3220500
168.3657716,73.2128747,-12.8711548,-744.1819192,-737.2925192
168.3657716,94.4260787,-42.8711548,-744.1584600,-737.3334600
168.3657716,103.2128747,-72.8711548,-744.1924654,-737.2975654
168.3657716,103.2128747,-42.8711548,-744.1768462,-737.2981462
168.3657716,103.2128747,-12.8711548,-744.2332734,-737.1766734
Path 7
114.3657716,73.2128747,-42.8711549,-743.1747311,0.0000000
114.3660403,73.2128580,-42.8711667,-743.1747383,0.0000000
114.5074685,73.2040417,-42.8773868,-743.1785564,0.0000000
131.1843871,72.1922509,-43.5192083,-743.5849095,0.0000003
151.0082298,71.0676854,-43.9507110,-743.9413550,0.0000010
163.0427991,70.4783444,-43.8302201,-744.0816961,0.0000023
167.3268580,70.3150982,-43.6224764,-744.1167085,0.0000027



PATCH NUMBER 8
Patch size was 60.0
Current state is state 0
Maximum Gaussian iterations was 7
Initial point for the patch was [ 167.32685802   70.31509819  -43.62247643]
Patch terminated at [ 183.6925856    96.1353827   -16.41072805]
Terminal point hit a boundary of the patch, drawing new patch!

Now the Z-matrix, Patch, and Path
Z-matrix for patch 8 was
 C
C,1,B1
H,1,B2,2,A1
H,1,B3,2,A2,3,D1,0
H,1,B4,2,A3,3,D2,0
C,2,B5,1,A4,4,D3,0
H,2,B6,1,A5,6,D4,0
```

```
C,6,B7,2,A6,1,D5,0
H,6,B8,2,A7,1,D6,0
H,8,B9,6,A8,2,D7,0
H,8,B10,6,A9,2,D8,0
H,8,B11,6,A10,2,D9,0
Variables:
B1=1.53536405
B2=1.10316111
B3=1.10425545
B4=1.10687577
B5=1.37028103
B6=1.09837022
B7=1.53490307
B8=1.0984525
B9=1.10393484
B10=1.10379502
B11=1.10682661
A1=110.96505733
A2=111.03313144
A3=112.17446948
A4=124.36843004
A5=116.79260009
A6=124.37049736
A7=118.63328247
A8=110.58890115
A9=111.19334205
A10=112.36143204
D1=120.00911403
D2=-120.23968599
D3=-46.53832498
D4=173.77500424
D5=168.36578242
D6=-6.05625409
D7=-162.28405545
D8=-42.871155
D9=78.05077553


Patch 8
1*2*6*8 6*2*1*5 2*6*8*11 state0 state1
161.3268580,40.3150982,-73.6224764,-744.1298009,-737.4007009
161.3268580,40.3150982,-43.6224764,-744.1072864,-737.3940864
161.3268580,40.3150982,-13.6224764,-744.1631304,-737.2843304
161.3268580,49.1018942,-43.6224764,-744.0881352,-737.4200352
161.3268580,70.3150982,-73.6224764,-744.0911835,-737.4388835
161.3268580,70.3150982,-64.8356804,-744.0740249,-737.4567249
161.3268580,70.3150982,-43.6224764,-744.0638414,-737.4400414
161.3268580,70.3150982,-22.4092724,-744.1005767,-737.3702767
161.3268580,70.3150982,-13.6224764,-744.1197725,-737.3417725
161.3268580,91.5283022,-43.6224764,-744.0878769,-737.3962769
161.3268580,100.3150982,-73.6224764,-744.1362069,-737.3512069
```

```
161.3268580,100.3150982,-43.6224764,-744.1066720,-737.3626720
161.3268580,100.3150982,-13.6224764,-744.1611002,-737.2537002
163.6104730,70.3150982,-43.6224764,-744.0849807,-737.4243807
170.1136540,40.3150982,-43.6224764,-744.1644095,-737.3455095
170.1136540,70.3150982,-73.6224764,-744.1462209,-737.3948209
170.1136540,70.3150982,-43.6224764,-744.1326906,-737.3908906
170.1136540,70.3150982,-13.6224764,-744.1901537,-737.2916537
170.1136540,100.3150982,-43.6224764,-744.1823025,-737.3012025
179.8463560,70.3150982,-43.6224764,-744.1687615,-737.3665615
191.3268580,40.3150982,-73.6224764,-744.1300358,-737.3838358
191.3268580,40.3150982,-64.8356804,-744.1276481,-737.3888481
191.3268580,40.3150982,-43.6224764,-744.1552124,-737.3516124
191.3268580,40.3150982,-22.4092724,-744.1979095,-737.2762095
191.3268580,40.3150982,-13.6224764,-744.2081979,-737.2506979
191.3268580,42.5987132,-43.6224764,-744.1518494,-737.3589494
191.3268580,49.1018942,-73.6224764,-744.1211502,-737.4041502
191.3268580,49.1018942,-43.6224764,-744.1450246,-737.3763246
191.3268580,49.1018942,-13.6224764,-744.1986322,-737.2785322
191.3268580,58.8345962,-43.6224764,-744.1436832,-737.3909832
191.3268580,70.3150982,-73.6224764,-744.1359847,-737.4035847
191.3268580,70.3150982,-71.3388614,-744.1340908,-737.4068908
191.3268580,70.3150982,-64.8356804,-744.1318628,-737.4120628
191.3268580,70.3150982,-55.1029784,-744.1376090,-737.4080090
191.3268580,70.3150982,-43.6224764,-744.1558082,-737.3847082
191.3268580,70.3150982,-32.1419744,-744.1798940,-737.3471940
191.3268580,70.3150982,-22.4092724,-744.1987134,-737.3109134
191.3268580,70.3150982,-15.9060914,-744.2079042,-737.2889042
191.3268580,70.3150982,-13.6224764,-744.2102325,-737.2824325
191.3268580,81.7956002,-43.6224764,-744.1779441,-737.3525441
191.3268580,91.5283022,-73.6224764,-744.1798964,-737.3290964
191.3268580,91.5283022,-43.6224764,-744.1983931,-737.3099931
191.3268580,91.5283022,-13.6224764,-744.2521697,-737.1894697
191.3268580,98.0314832,-43.6224764,-744.2102401,-737.2814401
191.3268580,100.3150982,-73.6224764,-744.1945319,-737.2953319
191.3268580,100.3150982,-64.8356804,-744.1904880,-737.3060880
191.3268580,100.3150982,-43.6224764,-744.2137302,-737.2722302
191.3268580,100.3150982,-22.4092724,-744.2556705,-737.1811705
191.3268580,100.3150982,-13.6224764,-744.2670177,-737.1437177
202.8073600,70.3150982,-43.6224764,-744.0829630,-737.4438630
212.5400620,40.3150982,-43.6224764,-743.9408456,-737.5040456
212.5400620,70.3150982,-73.6224764,-743.9267473,-737.5400473
212.5400620,70.3150982,-43.6224764,-743.9752674,-737.5128674
212.5400620,70.3150982,-13.6224764,-744.0157775,-737.4175775
212.5400620,100.3150982,-43.6224764,-744.0274070,-737.4015070
219.0432430,70.3150982,-43.6224764,-743.8803749,-737.5597749
221.3268580,40.3150982,-73.6224764,-743.7341702,-737.6155702
221.3268580,40.3150982,-43.6224764,-743.7957544,-737.5833544
221.3268580,40.3150982,-13.6224764,-743.8259196,-737.5096196
221.3268580,49.1018942,-43.6224764,-743.7999761,-737.5990761
221.3268580,70.3150982,-73.6224764,-743.7853311,-737.6104311
221.3268580,70.3150982,-64.8356804,-743.7960194,-737.6119194
```

```
221.3268580,70.3150982,-43.6224764,-743.8428118,-737.5770118
221.3268580,70.3150982,-22.4092724,-743.8755109,-737.5120109
221.3268580,70.3150982,-13.6224764,-743.8751988,-737.4928988
221.3268580,91.5283022,-43.6224764,-743.8840748,-737.5017748
221.3268580,100.3150982,-73.6224764,-743.8305200,-737.5234200
221.3268580,100.3150982,-43.6224764,-743.8886594,-737.4791594
221.3268580,100.3150982,-13.6224764,-743.9194762,-737.3908762
Path 8
167.3268580,70.3150982,-43.6224764,-744.1145505,0.0000000
167.3269293,70.3151060,-43.6224630,-744.1145510,0.0000000
167.3649981,70.3193081,-43.6152667,-744.1148351,0.0000000
172.5742307,71.1354925,-42.3107778,-744.1486841,0.0000006
177.0488989,72.5453928,-40.2794141,-744.1712228,0.0000010
179.8758139,74.3595700,-37.8695717,-744.1849768,0.0000011
181.6683666,76.6613653,-35.0236136,-744.1970128,0.0000013
182.7626781,79.5108577,-31.7508039,-744.2100451,0.0000014
183.3953685,82.9703561,-28.0847033,-744.2250433,0.0000016
183.7118025,87.0417205,-24.1341961,-744.2417399,0.0000020
183.7927444,91.5677455,-20.1267553,-744.2586271,0.0000021
183.6925856,96.1353827,-16.4107281,-744.2732188,0.0000016



PATCH NUMBER 9
Patch size was 60.0
Current state is state 0
Maximum Gaussian iterations was 6
Initial point for the patch was [ 183.6925856    96.1353827   -16.41072805]
Patch terminated at [ 180.04690164  120.64145274   -0.25729983]
Terminal point on the last state! Gaussian performing final optimization

Now the Z-matrix, Patch, and Path
Z-matrix for patch 9 was
 C
C,1,B1
H,1,B2,2,A1
H,1,B3,2,A2,3,D1,0
H,1,B4,2,A3,3,D2,0
C,2,B5,1,A4,4,D3,0
H,2,B6,1,A5,6,D4,0
C,6,B7,2,A6,1,D5,0
H,6,B8,2,A7,1,D6,0
H,8,B9,6,A8,2,D7,0
H,8,B10,6,A9,2,D8,0
H,8,B11,6,A10,2,D9,0
Variables:
B1=1.5279681
B2=1.10529698
B3=1.10400429
B4=1.10763518
B5=1.36929843
```

```
B6=1.09874964
B7=1.52725966
B8=1.09883829
B9=1.10646069
B10=1.10345453
B11=1.1073178
A1=112.09493438
A2=111.16636081
A3=110.43329925
A4=124.3490444
A5=116.67193263
A6=124.38928993
A7=118.75846003
A8=111.75109562
A9=111.31131277
A10=110.59591393
D1=121.22102819
D2=-119.86570908
D3=-18.59816473
D4=-173.78308717
D5=-168.67316751
D6=5.74298397
D7=-134.28020334
D8=-13.622476
D9=106.65401107


Patch 9
1*2*6*8 6*2*1*5 2*6*8*11 state0 state1
153.6925856,66.1353827,-22.4107281,-744.0129447,-737.4299447
153.6925856,66.1353827,7.5892719,-744.0662984,-737.3773984
153.6925856,66.1353827,37.5892719,-744.0367903,-737.4578903
153.6925856,74.9221787,7.5892719,-744.0631729,-737.3727729
153.6925856,96.1353827,-22.4107281,-744.0342289,-737.3754289
153.6925856,96.1353827,-13.6239321,-744.0547071,-737.3442071
153.6925856,96.1353827,7.5892719,-744.0899135,-737.3092135
153.6925856,96.1353827,28.8024759,-744.0801083,-737.3579083
153.6925856,96.1353827,37.5892719,-744.0635090,-737.3890090
153.6925856,117.3485867,7.5892719,-744.1337466,-737.2232466
153.6925856,126.1353827,-22.4107281,-744.0924690,-737.2861690
153.6925856,126.1353827,7.5892719,-744.1452571,-737.2071571
153.6925856,126.1353827,37.5892719,-744.1172207,-737.3056207
155.9762006,96.1353827,7.5892719,-744.1198735,-737.2865735
162.4793816,66.1353827,7.5892719,-744.1572225,-737.3124225
162.4793816,96.1353827,-22.4107281,-744.1453491,-737.2965491
162.4793816,96.1353827,7.5892719,-744.1916843,-737.2257843
162.4793816,96.1353827,37.5892719,-744.1559395,-737.3181395
162.4793816,126.1353827,7.5892719,-744.2381960,-737.1261960
172.2120836,96.1353827,7.5892719,-744.2604474,-737.1584474
183.6925856,66.1353827,-22.4107281,-744.2096625,-737.2990625
183.6925856,66.1353827,-13.6239321,-744.2234652,-737.2703652
```

```
183.6925856,66.1353827,7.5892719,-744.2268629,-737.2565629
183.6925856,66.1353827,28.8024759,-744.1909810,-737.3221810
183.6925856,66.1353827,37.5892719,-744.1727860,-737.3530860
183.6925856,68.4189977,7.5892719,-744.2292772,-737.2529772
183.6925856,74.9221787,-22.4107281,-744.2202444,-737.2851444
183.6925856,74.9221787,7.5892719,-744.2386441,-737.2362441
183.6925856,74.9221787,37.5892719,-744.1853393,-737.3352393
183.6925856,84.6548807,7.5892719,-744.2572576,-737.1944576
183.6925856,96.1353827,-22.4107281,-744.2615400,-737.1926400
183.6925856,96.1353827,-20.1271131,-744.2656255,-737.1823255
183.6925856,96.1353827,-13.6239321,-744.2754524,-737.1552524
183.6925856,96.1353827,-3.8912301,-744.2834435,-737.1297435
183.6925856,96.1353827,7.5892719,-744.2801593,-737.1321593
183.6925856,96.1353827,19.0697739,-744.2646968,-737.1674968
183.6925856,96.1353827,28.8024759,-744.2458066,-737.2106066
183.6925856,96.1353827,35.3056569,-744.2324355,-737.2395355
183.6925856,96.1353827,37.5892719,-744.2279340,-737.2490340
183.6925856,107.6158847,7.5892719,-744.2967696,-737.0793696
183.6925856,117.3485867,-22.4107281,-744.2862874,-737.1160874
183.6925856,117.3485867,7.5892719,-744.3020168,-737.0609168
183.6925856,117.3485867,37.5892719,-744.2491243,-737.1960243
183.6925856,123.8517677,7.5892719,-744.3001223,-737.0657223
183.6925856,126.1353827,-22.4107281,-744.2842024,-737.1185024
183.6925856,126.1353827,-13.6239321,-744.2970927,-737.0805927
183.6925856,126.1353827,7.5892719,-744.2984474,-737.0706474
183.6925856,126.1353827,28.8024759,-744.2624413,-737.1652413
183.6925856,126.1353827,37.5892719,-744.2449309,-737.2057309
195.1730876,96.1353827,7.5892719,-744.2334187,-737.1705187
204.9057896,66.1353827,7.5892719,-744.0824536,-737.3701536
204.9057896,96.1353827,-22.4107281,-744.1545292,-737.2718292
204.9057896,96.1353827,7.5892719,-744.1420606,-737.2445606
204.9057896,96.1353827,37.5892719,-744.0828320,-737.3655320
204.9057896,126.1353827,7.5892719,-744.1274122,-737.2381122
211.4089706,96.1353827,7.5892719,-744.0551147,-737.3075147
213.6925856,66.1353827,-22.4107281,-743.9868443,-737.4639443
213.6925856,66.1353827,7.5892719,-743.9623153,-737.4444153
213.6925856,66.1353827,37.5892719,-743.9025219,-737.5385219
213.6925856,74.9221787,7.5892719,-743.9836092,-737.4155092
213.6925856,96.1353827,-22.4107281,-744.0445113,-737.3432113
213.6925856,96.1353827,-13.6239321,-744.0470211,-737.3196211
213.6925856,96.1353827,7.5892719,-744.0197897,-737.3314897
213.6925856,96.1353827,28.8024759,-743.9728666,-737.4131666
213.6925856,96.1353827,37.5892719,-743.9610868,-737.4450868
213.6925856,117.3485867,7.5892719,-744.0089954,-737.3180954
213.6925856,126.1353827,-22.4107281,-744.0214356,-737.3434356
213.6925856,126.1353827,7.5892719,-743.9919903,-737.3392903
213.6925856,126.1353827,37.5892719,-743.9333392,-737.4438392
Path 9
183.6925856,96.1353827,-16.4107280,-744.2716513,0.0000001
183.6925840,96.1354010,-16.4107133,-744.2716514,0.0000001
183.6917626,96.1446359,-16.4032636,-744.2716794,0.0000001
```

```
183.5656674,97.4826165,-15.3356565,-744.2756383,0.0000000
183.0597074,101.9856368,-11.9028546,-744.2874462,0.0000002
182.5774497,105.6572761,-9.2717353,-744.2952510,0.0000005
182.0813492,109.0603691,-6.9593284,-744.3009590,0.0000006
181.6304007,111.9056736,-5.1192371,-744.3045830,0.0000008
180.0469016,120.6414527,-0.2572998,-744.3092285,0.0000009


Final Path
-180.   , 120.51 ,  -0.06 ,-744.309903152
```

B.5 Simulation Inputs

All of the inputs for the simulation except the initial Z-matrix are in the file chem_vars.py which will lie in the directory in which you unzip the simulation files. Below is a blank sample of that file:

```
#! usr/local/apps/python-2.6.5/bin/python
#chemistry inputs for light-induced simulation

#number of design variables
d =

#name of your molecule
molecule = ''

#the number of atoms in your molecule
num_atoms =

#the number of cores for your gaussian jobs
n_proc =

#method of energy computation (hf,b3lyp,...)
method = ''

#basis-set for you gaussian jobs (cep-31g,CBSB7,...)
basis_set = ''

#Gaussian job type (opt=modredundant) and anything else you want in the header
calculation = ''

#state transitions for your simulation
```

```
state_order = []

#where will your gaussian jobs be run?
queue = ''

#set this to 1 if you want every z-matrix that was calculated during your sim
all_zmats =

#atom number, value, ...
bond_lengths = []

#for reading output files, i.e. 'C1-C2'
bond_names = []

#atom numbers, value, ...
valence_angles = []

#for reading output files, i.e. 'C1-C2-C6'
valence_names = []

#atom numbers, value, ...
torsion_angles = []

#for reading output files, i.e. 'C1-C2-C6-C8'
torsion_names = []

#pre-processed variable, pre-processed to whom, amount to pre-process
deps = []

#where your molecule sits, this directory must have initial zmat as zmat.gjf
home_dir = ''

#where to write all your files
share_dir =  ''
```

We now describe each input consecutively

d The number of design variables for your simulation

molecule This name will be written in the simulation summary file for future reference

num_atoms The number of atoms in the molecule

n_proc The number of cores for each individual gaussian job

method Energy computation method within gaussian

basis_set Basis set for the energy computations

calculation Not only should you specify "opt=modredundant", but any other statements you would like to add to the gaussian header should be placed here. ie geom=connectivity

state_order The quantum states through which your simulation will transition

queue The queue in which the gaussian jobs will run

all_zmats Setting this option to 1 will cause the simulator to save every z-matrix on every patch. Otherwise the simulator will only save the original on each patch.

design_angles Design angles for the simulation specified by atom numbers in a string followed by the value of the angle

angle_names Each of the angles in the design angle portion must be paired with the atom type and separated by hyphens

deps These are the angles to be pre-processed as the design angles are changed. Each angle requires 3 inputs: atom numbers in a string, design angle to be pre-processed with (counted by 2 from 0), and the value to be pre-processed by.

home_dir Directory containing zmat.gjf

share_dir Where do you want to write all the files?

B.6 Simulation Output

Once the simulation has begun, there will be a txt file summarizing the progress of the simulation in the share_dir specified in your input file chem_vars.py. The file is named Simulation_summary.txt. The file begins with some information identifying the simulation for future reference (molecule, basis set, energy computation method, initial point, initial z-matrix) along with some information about the initial patch as shown below

```
Thank you for using LITES
You will be simulating the molecule ____ using ____ with ____ as a basis set
We begin on state ____
with initial value of x = [   ____   ]
Initial patch size is ____
Beginning the simulation with the following Z-matrix
```

Following the Z-matrix will be summaries of every patch. The patch summary will give the number of the patch, the size of the patch, the state the simulation was run on, the maximum number of gaussian iterations, the initial and terminal points, and the result of the patch (failed to converge, hit a boundary, found a minimum,..). A sample of this section is provided below

```
PATCH NUMBER 0
Patch size was 20.0
Current state is state 1
Maximum Gaussian iterations was 5
Initial point for the patch was [   0.  120. -120.]
Patch terminated at [  2.64463711e-02   1.10427830e+02  -1.10386277e+02]
Terminal point hit a boundary of the patch, drawing new patch!
```

Next the summary provides the Z-matrix used as the initial iterate for constructing that patch. If the patch converged, then each interpolation node is provided and the simulation path is provided. The patch is provided following 2 lines. The first line numbers the patch and the second is a key for reading the patch columns as shown below

```
Patch 0
1*2*6*8 6*2*1*5 2*6*8*11 state0 state1
-10.0000000,110.0000000,-130.0000000,-744.2032174,-736.9677174
-10.0000000,110.0000000,-120.0000000,-744.2079530,-736.9393530
-10.0000000,110.0000000,-110.0000000,-744.2080695,-736.9375695
.
.
.
```

The numbers separated by * are the torsion angles chosen for the simulation, while the later columns just give the energy at each requested state. The simulation path is provided after the patch following the line stating the path number. The columns of this table are the value of the design variables, the energy at that point, and the predicted error at that point:

```
Path 0
0.0000000,120.0000000,-120.0000000,-744.2302755,0.0000000
0.0000000,120.0000000,-120.0000000,-736.8791755,0.0000000
-0.0000000,119.9999981,-119.9999981,-736.8791755,0.0000000
-0.0000000,119.9991230,-119.9991229,-736.8791759,0.0000000
0.0000003,119.8601736,-119.8601642,-736.8792331,0.0000000
.
.
.
```

Appendix C

Codes

The codes in this section are the application of the algorithm described in Section 6.2.5 and are exclusively written for use with Python 2.6.5 or Linux C-Shell.

The file LITES_defs.py contains classes, script editors, and error controls for the simulation:

```
#! usr/local/apps/python-2.6.5/bin/python
#module of script editors and classes for LITES

#import modules we will use
from numpy import *
import os, time, shutil as si
import chem_vars as cv, math_vars as mv, LITES_math as lm, lhs


#Simulation class of values to be carried along throughout the simulation
class Simulation:
   def __init__(self):
self.molecule = cv.molecule
self.num_atoms = cv.num_atoms
self.method = cv.method
self.basis_set = cv.basis_set
self.headfoot = make_headfoot(cv.n_proc, cv.method, cv.basis_set, cv.molecule, cv.calculation,\
 len(cv.state_order))
mids = []
for angle in cv.torsion_angles[::2]:
   m=angle.split()
   mid = ' '.join(['D','*',m[1],m[2],'*','R','\n'])
   mids.append(mid)
self.mids = ''.join(mids)
self.bond_lengths = cv.bond_lengths
self.valence_angles = cv.valence_angles
self.torsion_angles = cv.torsion_angles
self.share_dir = cv.share_dir
self.deps = cv.deps
self.state_order = cv.state_order
self.queue = cv.queue
self.n_proc = cv.n_proc
self.bond_names = cv.bond_names
self.valence_names = cv.valence_names
self.torsion_names = cv.torsion_names
set = lm.make_index(mv.k,cv.d)
self.point_list = array(lm.gen_grid(set)[0])
bls = array([float(i) for i in cv.bond_lengths[1::2]])
vas = array([float(i) for i in cv.valence_angles[1::2]])
tas = array([float(i) for i in cv.torsion_angles[1::2]])
self.init_point = mat(hstack([bls,vas,tas]))
self.conv_patches = []
self.q_size = mv.q_size
self.H0 = mv.H0
bd,bu = self.init_point.transpose()-mv.H0, self.init_point.transpose()+mv.H0
self.bounds0 = bmat('bd bu')
self.gau_time = mv.gau_time
self.sim_time = mv.sim_time
self.ec = mv.error_control



#makes the header and footer that are in every gjf file
#n_proc=number of processors, method=energy computation method,
#basis_set=gaussian basis set, molecule=duh, calculation= type of
#calculation(i.e. opt=modredundant), num_states=number of excited states
#both header and footer are long strings
def make_headfoot(n_proc, method, basis_set, molecule, calculation, num_states):
   header = ''
   header = header + '%%chk=%s\n' % molecule
   header = header + '%MEM=2GB\n'  #unnecessary in g03, but seem to be for g09
   header = header + '%%NProcShared=%s\n' % n_proc
   header = header + '# %s %s/%s IOp(99/14=1) scf=tight test\n' % (calculation, method, basis_set)
   header = header + '\n'
   header = header + 'header made by Dave Mokrauer\n\n'
   header = header + '0 1\n'
   footer = ''
   footer = footer + '\n--Link1--\n'
   footer = footer + '%%chk=%s\n' % molecule
   footer = footer + '%MEM=2GB\n'   #unnecessary in g03, but seem to be for g09
   footer = footer + '%NoSave\n'
   footer = footer + '%%NProcShared=%s\n' % n_proc
   footer = footer + '# td(NStates=%s) %s/%s scf=tight Geom=(AllCheck,CAngle,CDihedral) Guess=\
Read test\n' % (num_states, method, basis_set)
   return header, footer


#Patch class for each step of the simulation, this passes to function evaluations and RK45
class Patch:
   def __init__(self, patch_number, bounds, state, init_point, zmat, point_list):
```

```
self.p = patch_number
self.k = mv.k - 1
self.d = cv.d
self.b = bounds
self.s = state
self.i = init_point
self.GTOL_RK = mv.GTOL_RK
self.ETOL_RK = mv.ETOL_RK
self.ETOL_sim = mv.ETOL_sim
self.Fudge_sim = mv.Fudge_sim
self.gau_max = mv.gau_max
self.RK_hmin = mv.RK_hmin
self.H = (bounds[0,1]-bounds[0,0])/2.
if patch_number == 0:
   self.zmat = read_initial_zmat()
else:
   self.zmat = zmat
self.at_min = 0
self.dist = mv.shift_dist
self.grid = zeros(point_list.shape)
for i in range(0,self.grid.shape[1]):
   self.grid[:,i] = (bounds[i,1] - bounds[i,0])*point_list[:,i]/2. + bounds[i,0]+self.H
self.Data=mat(zeros(self.grid.shape))
self.Path=mat(zeros((3,3)))



#reads the initial z-matrix from your zmat.gjf file
def read_initial_zmat():
   file = open('zmat.gjf', 'r') #open the file
   zmat = file.read()
   file.close
   zmat = zmat.split('\r\n') #remove the end lines
   zmat = '\n'.join(zmat)
   while zmat[-1] == '\n': #remove all the blank lines at the end
zmat=zmat[:-1]
   return zmat

#runs the gaussian jobs on the patch, checks for convergence, and returns max gaussian iterations
def run_patch(Sim, Patch):
   patch_fin = 1  #indicator that the jobs finished
   max_iters = 0  #most iterations of any job on the patch

   #make and submit a gaussian file for every point
   for i in range(0,Patch.grid.shape[0]):
        make_gjf(Patch.zmat, Patch.grid[i], Sim, i)
        make_sub_file(Sim, i)
        os.system('bsub < g_sub1')
   sub_files = range(0,Patch.grid.shape[0])

   #wait for gaussian files to return, check if they converged, kill those running if there is a failure
   while len(sub_files) > 0:
        for i in sub_files:
           if os.path.exists('%s/%s_sp%s.log' % (Sim.share_dir, Sim.molecule, i)) == True:
                sub_files.remove(i)
                if convergence_check('%s/%s_sp%s.log' % (Sim.share_dir, Sim.molecule, i))==0:
                   patch_fin = 0
if patch_fin == 0:
   sub_files = []
   #kills the runing gaussian jobs
   os.system('bkill -J E_files*')

   # if they all converged write the files
   if patch_fin == 1:
        time.sleep(5)
        max_iters = write_patch(Sim.share_dir,Patch.grid, Patch.p, Sim.molecule, max(Sim.state_order))
   return max_iters, patch_fin



#makes the file that submits gaussian jobs by replacing place holders in file g_sub
def make_sub_file(Sim, file_number):
   f = open('g_sub','r')
   s = f.read()
   f.close()
   s = s.split('\n')
   scratch = '/scratch/dsmokrau%s' % file_number
   gjf = '%s_sp%s.gjf' % (Sim.molecule, file_number)
   log = '%s_sp%s.log' % (Sim.molecule, file_number)
   for line in s:
      line1 = line.split(' ')
      if 'log' in line1:
         line1[line1.index('log')] = log
      if 'gjf' in line1:
         line1[line1.index('gjf')] = gjf
      if 'scratch' in line1:
         line1[line1.index('scratch')] = scratch
      if 'n_proc' in line1:
         line1[line1.index('n_proc')] = str(Sim.n_proc)
      if 'queue' in line1:
         line1[line1.index('queue')] = Sim.queue
      if 'time' in line1:
         line1[line1.index('time')] = str(Sim.gau_time)
      if 'share' in line1:
         line1[line1.index('share')] = Sim.share_dir
      if 'out' in line1:
         line1[line1.index('out')] = '%s_sp%s.out' % (Sim.molecule, file_number)
      if 'err' in line1:
         line1[line1.index('err')] = '%s_sp%s.err' % (Sim.molecule, file_number)
      if 'E_files' in line1:
 line1[line1.index('E_files')] = 'E_files%s' % file_number
      line1 = ' '.join(line1)
      line1 = line1 + '\n'
      s[s.index(line)]=line1
   f = open('g_sub1','w')
   f.writelines(s)
   f.close()


#makes the gjf file with the header and footer for the simulation
#need to add bond length
def make_gjf(zmat, x, Sim, file_number):
   angle_edits = ''
   for i in Sim.valence_angles[::2]:
angle_edits = angle_edits + 'A %s %f F\n' % (i, x[Sim.valence_angles.index(i)/2])
   for i in Sim.torsion_angles[::2]:
        angle_edits = angle_edits + 'D %s %f F\n' % (i, x[len(Sim.valence_angles)/2\
+Sim.torsion_angles.index(i)/2])
   for i in Sim.deps[::3]:
        angle_edits = angle_edits + 'D %s %f\n'  % (i, float(x[Sim.deps[Sim.deps.index(i)+1]]\
+Sim.deps[Sim.deps.index(i)+2]))
   tzmat = zmat+'\n\n'
   gjf_file = open("%s_sp%s.gjf" % (Sim.molecule, file_number), 'w')
   gjf_file.write(Sim.headfoot[0] + tzmat + Sim.mids + angle_edits + Sim.headfoot[1])
   gjf_file.close()



#collects all the patch's data into a function so that the smolyak algorithm can evaluate
#pretty standard, read lines and write them
#also counts the iterations
def write_patch(share_dir,X,patch_number,molecule,num_states):
   X=mat(X)
   max_iters = 0
   energy = mat(zeros((X.shape[0],num_states+1)))
   for i in range(0,X.shape[0]):
#################
        [evals, iters] = get_energy_vals(share_dir, molecule, i, num_states)
        if iters > max_iters:
           max_iters = iters
#################
        energy[i]=evals
   Data=bmat('X energy')
   savetxt('%s/Data%s.txt' % (share_dir,patch_number), Data)
   return max_iters


#just extracts energy from the log file and counts the iterations
def get_energy_vals(share_dir, molecule, file_number, num_states):
   inf = open("%s/%s_sp%s.log" % (share_dir, molecule, file_number), 'r')
   F = inf.read()
   inf.close()
   F = F.split('\n')
   Z = []
   iters = 0
   for line in F:
        line = line.split()
        Z.append(line)
   matrix_line=[]
   E_spots=[]
   exc_line = ['Excited', 'State']
   for line in Z:
        if line[0:2] == ['SCF', 'Done:']:
           E_spots.append(Z.index(line))
           iters = iters + 1
        for i in range(1, num_states+1):
           exc_line.append('%s:' % i)
           if (line[0:3] == exc_line):
#                if (i == num_states):
#                   matrix_line.append('%s\n' % line[4])
#                else:
                matrix_line.append(line[4])
           exc_line = exc_line[:-1]
   matrix_line.insert(0,Z[E_spots[-1]][4])
   matrix_line[0]=float(matrix_line[0])*27.2112
   for i in range(1,len(matrix_line)):
matrix_line[i]=float(matrix_line[i])+matrix_line[0]
   iters = iters - 1
   return matrix_line, iters


#checks if the gaussian file converged by looking for a zmatrix and a statement of convergence
#input is the file to check
def convergence_check(log_file):
   log = open('%s' % log_file, 'r')
   log_str = log.read()
   log.close()
   log_str = log_str.split('\n')
   chk1 = '    -- Stationary point found.' in log_str
   chk2 = ' Final structure in terms of initial Z-matrix:' in log_str
   chk = 0
   if (chk1 == 1) and (chk2 == 1):
        chk=1
   return chk


#runs the integrator and a newton if a minimum is found.  The rest is data for passing
def Patch_Path(Patch,Sim):
   #integrator
   [path,f,edge,reject_patch]=lm.RK45(Patch)

   #do a newton if steady state is reached
   at_min=0
   if all(edge==0):
```

```
at_min=1
[pn,fn] = lm.newton(path[-1],Patch)
    path=bmat('path;pn')
    f=bmat('f;fn')

   #RK error needs the higher order function values
   Patch.k = Patch.k+1
   f1=lm.my_fun(path,Patch)

   #Relative error for RK approach
   int_err=max(abs(f-f1)/abs(f))[0,0]

   #Values for TR approach
   o_energy=f[0]
   p_energy=f[-1]

   #next initial point
   init_point=path[-1]
   Patch.Data=mat(genfromtxt('Data%s.txt' % Patch.p ,delimiter=' '))
   Patch.Path=path

   #Find the nearest zmat to the initial point on the next patch
   nearby,dist,p_count=0,10000,0
   for i in Patch.Data[:,0:Patch.d]:
if linalg.norm(init_point[0]-i[0])<dist:
   dist=linalg.norm(init_point[0]-i[0])
   nearby=p_count
p_count=p_count+1

   # file with values for checking the exact error
   savetxt('check_error.py',path)
   path=bmat('path f')
   savetxt('Path%s.txt' % Patch.p, path)
   init_point = array(init_point)
   return init_point, edge, at_min, nearby, p_energy, o_energy, int_err



#Calculates the new patch size using number of gaussian iterations and the error estimate
def RK_approach(Patch, int_err, max_iters, init_point, edge):
   if max_iters > Patch.gau_max:
H = Patch.H/2.
   else:
print('Patch.k = %s' % Patch.k)
H = Patch.Fudge_sim*Patch.H*(Patch.ETOL_sim/int_err)**(1.0/Patch.k)
   x0 = init_point+(Patch.dist*2-1)*edge*H
   bd,bu = x0.transpose()-H, x0.transpose()+H
   new_bounds = bmat('bd bu')
   return new_bounds
```

```
#reads the z-matrix from the log file
#i.e. file = 'TMS_sp3.log'
def read_zmat(file,num_atoms):
   zmat=[]
   inf = open("%s" % file, 'r')
   F = inf.read()
   inf.close()
   F = F.split('\n') # we have to split the file twice in order to read the first line
   Z = []
   for line in F:
        line = line.split() # 2nd split
        Z.append(line)
   for line in Z:
        if len(line)!=0 and line[0] == 'Final': #signals the end of the zmatrix
           locale = Z.index(line)
   num_vars = 4*num_atoms-4
   zmat=zmat+Z[locale+1:locale+num_vars][:]
   zmat.append([''])
   zmat1=''
   for line in zmat:
        zmat1=zmat1+line[0]+'\n'
   zmat = zmat1
   while zmat[-1] == '\n':
zmat=zmat[:-1]
   return zmat


#When the integrator finds a minimum, we change states and estimate
#a global error from a latin hypercube
#If the error is unacceptable on the new state we will drop the patch
def keep_patch(Patch,next_state):
   lhc=mat(lhs.lhsFromSample(Patch.b.transpose(),siz=1000))
   Patch.s = next_state
   F1 = lm.my_fun(lhc,Patch)
   Patch.k = Patch.k-1
   F2 = lm.my_fun(lhc,Patch)
   int_err = max(abs(F2-F1)/abs(F2))[0,0]
   return int_err


#Makes or continues summary of simulation file
#Just a script to write to the file
def Sim_Sum(Sim,Patch,patch_fin,at_min,gau_iters):
   if os.path.exists('Simulation_Summary.txt') == False:
sim_file = open('%s/Simulation_Summary.txt' % cv.share_dir,'w')
sim_file.write('Thank you for using LITES\n')
sim_file.write('You will be simulating the molecule %s using\
   %s with %s as a basis set\n' % (Sim.molecule,Sim.method,Sim.basis_set))
sim_file.write('We begin on state %s\n' % Sim.state_order[0])
sim_file.write('with initial value of x = %s' % Patch.i)
```

```
H = 2*Patch.H
sim_file.write('\nInitial patch size is %s\n' % H)
sim_file.write('Beginning the simulation with the following Z-matrix\n')
si.copyfileobj(open('zmat.gjf','r'),sim_file)
sim_file.write('\n\nNow for the patch summaries!')
sim_file.close()
   else:
sim_file = open('%s/Simulation_Summary.txt' % cv.share_dir, 'a')
sim_file.write('\n\n\nPATCH NUMBER %s\n' % Patch.p)
H = 2*Patch.H
sim_file.write('Patch size was %s\n' % H)
if patch_fin == 0:
   sim_file.write('Convergence Failure! Shrinking the Patch!\n\n')
#    if .5*H < mv.H_min:
# sim_file.write('Simulation Failure!  H too small!')
else:
   sim_file.write('Current state is state %s\n' % Patch.s)
   sim_file.write('Maximum Gaussian iterations was %s\n' % gau_iters)
   sim_file.write('Initial point for the patch was %s\n' % Patch.i)
   sim_file.write('Patch terminated at %s\n' % Patch.Path[-1])
   if (at_min == 1) and (Patch.s != Sim.state_order[-1]):
sim_file.write('Terminal point was a minimum, changing states!\n\n')
   elif (at_min == 1) and (Patch.s == Sim.state_order[-1]):
sim_file.write('Terminal point on the last state! Gaussian performing final optimization\n\n')
   else:
        sim_file.write('Terminal point hit a boundary of the patch, drawing new patch!\n\n')
           sim_file.write('Now the Z-matrix, Patch, and Path\n')
   sim_file.write('Z-matrix for patch %s was\n %s\n' % (Patch.p,Patch.zmat))
#    vars=[]
#    for line in Sim.torsion_angles:
# line1 = line.split()
#         line1 = '*'.join(line1)
#         Sim.torsion_angles[Sim.torsion_angles.index(line)]=line1
#    for i in range(0,len(Sim.state_order)+1):
#         Sim.torsion_angles.append('state%s' % i)
#    vars.append('\n')
#    vars = ' '.join(vars)
   sim_file.write('Patch %s \n' % Patch.p)
#    sim_file.write(vars)
   si.copyfileobj(open('Data%s.txt' % Patch.p,'r'),sim_file)
   sim_file.write('Path %s \n' % Patch.p)
   si.copyfileobj(open('Path%s.txt' % Patch.p,'r'),sim_file)
   sim_file.close()
   return
```

The file LITES_math.py contains the math codes for the simulation. These include Smolyak interpolation codes, Runge-Kutta 45, and a Newton's method code as well as function evaluations from Gaussian.

```
#! usr/local/apps/python-2.6.5/bin/python


#module of math codes for LITES all converted from matlab
```

```
from math import *
from numpy import *



#makes every index for the Smolyak interpolation
#k is the degree of exactness
#d is the degree
def make_index(k,d):
   set = ones((1,d))
   #generate all the possible indices
   for i in range(0,k):
for j in range(set.shape[0]-d**i+1,set.shape[0]+1):
   set = vstack([set,tile(set[j-1],(d,1))+eye(d)])

   #remove the redundancies
   set = unique(set.view([('',set.dtype)]*set.shape[1])).view(set.dtype).reshape(-1,set.shape[1])
   lines=[]

   #remove elements whose sums are too small
   for i in range(0,set.shape[0]):
if sum(set[i]) < k+1:
   lines.append(i)
   set=delete(set,lines,0)
   return set



#Generates the set of points for the patches and writes to a file
def gen_grid(set):
   max_size = max(set[:,0])
   nodes = zeros((max_size,2**(max_size-1)+1))

   #make the nodes
   for i in range(1,int(max_size)):
j=range(1,2**i+2)
nodes[i,range(0,2**i+1)]=-cos(pi*(j-ones(len(j)))/(2**i))

   #make sure that 0 is 0
   nodes[nonzero(abs(nodes)<10**(-15))]=0
   nodes=mat(nodes)
   grid = zeros((1,set.shape[1]))

   #now the grid from the cartesian product of each dimension for the index
   for i in set:
w = 'cartesian(('
for k in range(0,len(i)):
   if i[k] == 1:
w = w +'[nodes[0,0]]'
   else:
```

```
w = w + 'nodes[' + str(int(i[k]-1)) + ',range(0,' + str(int(2**(i[k]-1)+1)) + ')]'
   if k != len(i)-1:
w = w + ','
w = w + '))'
grid = vstack([grid,eval(w)])

   #remove the redundancies
   grid = unique(grid.view([('',grid.dtype)]*grid.shape[1])).view(grid.dtype).reshape(-1,grid.shape[1])
   grid = mat(grid)
   return grid, nodes



#cartesian product of arrays
def cartesian(arrays, out=None):
   arrays = [asarray(x) for x in arrays]
   dtype = arrays[0].dtype

   n = prod([x.size for x in arrays])
   if out is None:
        out = zeros([n, len(arrays)], dtype=dtype)

   m = n / arrays[0].size
   out[:,0] = repeat(arrays[0], m)
   if arrays[1:]:
        cartesian(arrays[1:], out=out[0:m,1:])
        for j in xrange(1, arrays[0].size):
           out[j*m:(j+1)*m,1:] = out[0:m,1:]
   return out


#d-dimensional Smolyak interpolation with degree of exactness k
#x is the points you are evaluating, where each row is a point, x is a MATRIX!
def ddim_smol(x,Patch):
   set = make_index(Patch.k,Patch.d)
   max_size=max(set[:,0])
   nodes = zeros((max_size,2**(max_size-1)+1))

   #compute the nodes
   for i in range(1,int(max_size)):
j=range(1,2**i+2)
nodes[i,range(0,2**i+1)]=-cos(pi*(j-ones(len(j)))/(2**i))
   nodes[nonzero(abs(nodes)<10**(-15))]=0

   g = zeros(x.shape)
   f = zeros((x.shape[0],1))

   #Linear combinations for Smolyak
   for i in set:
      [f_cur,g_cur] = vec_lagrange(i,nodes,x,Patch)
      coeffs = (-1)**(Patch.d+Patch.k-sum(i))*factorial(Patch.d-1)/(factorial(Patch.d-1\
   -Patch.d-Patch.k+sum(i))*factorial(Patch.d+Patch.k-sum(i)))
      f=f+f_cur*coeffs
      g=g+coeffs*g_cur.transpose()
   f=mat(f)
   g=mat(g)
   return f, g


#multi-dimensional lagrange interpolation for vector i of indices with associated
#smolyak nodes. Takes the product of 1d lagrange polynomials
#x is columns of points
def vec_lagrange(i,nodes,x,Patch):

   #generate the grid for the current index i
   w = 'cartesian(('
   for k in range(0,len(i)):
      if i[k] == 1:
         w = w +'[nodes[0,0]]'
      else:
         w = w + 'nodes[' + str(int(i[k]-1)) + ',range(0,' + str(int(2**(i[k]-1)+1)) + ')]'
      if k != len(i)-1:
         w = w + ','
   w = w + '))'
   grid = mat(eval(w))

   #move the grid to the current domain
   bounds1=tile((Patch.b[:,1]-Patch.b[:,0]).transpose()/2,(grid.shape[0],1))
   grid1=multiply(grid,bounds1)
   grid1 = grid1 + tile((Patch.b[:,1]+Patch.b[:,0]).transpose()/2,(grid.shape[0],1))

   #evaluate the function at each gridpoint
   f = Gau_Evals(grid1,Patch)

   #pre-allocate
   lg_poly = ones((grid.shape[0],x.shape[0]))
   grad = ones((grid.shape[0],x.shape[0],grid.shape[1]))
   z=0
   grid=array(grid)
   for j in grid:
      for k in range(0,len(j)):
         if i[k] != 1:
            cur_nodes=nodes[i[k]-1,range(0,int(2**(i[k]-1)+1))] #row
            cur_nodes=cur_nodes[cur_nodes!=j[k]] #row

    #move the nodes
            cur_nodes=cur_nodes*(Patch.b[k,1]-Patch.b[k,0])/2+(Patch.b[k,1]+Patch.b[k,0])/2; #row
            oth_node=j[k]*(Patch.b[k,1]-Patch.b[k,0])/2+(Patch.b[k,1]+Patch.b[k,0])/2  #row

    #evaluate the lagrange poly
            cur_den=tile(oth_node-cur_nodes,(x.shape[0],1))
            cur_num=tile(x[:,k],(1,len(cur_nodes)))-tile(cur_nodes,(x.shape[0],1))
            cur_dim=prod(cur_num/cur_den,1)
            lg_poly[z,:]=multiply(lg_poly[z,:],cur_dim.transpose());

    #differentiate the same way
            other_vars=mat(range(0,len(j)))
            other_vars=other_vars[other_vars!=k] #only diff the variable k
            cur_dim = tile(cur_dim.transpose(),(1,other_vars.shape[1]))
            cur_dim = array(cur_dim)
            cur_dim = cur_dim.reshape(1,x.shape[0],other_vars.shape[1],order='F')
    #product with only other variables
            grad[z,:,other_vars]=multiply(grad[z,:,other_vars][0],cur_dim[0].transpose())
            int_num = zeros((1,x.shape[0]))
            for m in range(0,cur_num.shape[1]):
               inds=array(cur_num)
               inds[:,m]=1

       #product rule
               int_num=int_num+prod(inds,1).transpose()
            grad[z,:,k]=multiply(grad[z,:,k],int_num)/(prod(cur_den,1))
         else:

    #constant terms zero out
            grad[z,:,k]=0


      z=z+1


   g=tile(f,(1,grad.shape[1]*grad.shape[2]))
   g=array(g)
   g=g.reshape(f.shape[0],grad.shape[1],grad.shape[2],order='F')

   #sum the polynomials
   grad=sum(multiply(g,grad),0)
   grad=grad.reshape(x.shape[0],x.shape[1],order='F')
   grad=grad.transpose()
   f=sum(multiply(tile(f,(1,x.shape[0])),lg_poly),0).transpose()
   return f,grad




#Runge-Kutta 4/5 with Dormand=Prince coefficients for use in continuous steepest
#descent optimization on a closed and bounded surface
#you are solving x'=grad f(x)
#my_mod is the module with all of your functions
#grad is the NEGATIVE gradient of your function
#mf is the function you are minimizing
#Hess is the function that gives you a Hessian
#all in this module
#y is a matrix of a single row!
```

```
def RK45(Patch):
   h=.01
   y = Patch.i
   f = my_fun(y,Patch)
   A= mat([[0., 0., 0., 0., 0., 0., 0.],
           [1./5., 0., 0., 0., 0., 0., 0.],
           [3./40., 9./40., 0., 0., 0., 0., 0.],
           [44./45., -56./15., 32./9., 0., 0., 0., 0.],
           [19372./6561., -25360./2187., 64448./6561., -212./729., 0., 0., 0.],
           [9017./3168., -355./33., 46732./5247., 49./176., -5103./18656., 0., 0.],
           [35./384., 0., 500./1113., 125./192., -2187./6784., 11./84., 0.]])
   B=mat([[5179./57600., 0., 7571./16695., 393./640., -92097./339200., 187./2100., 1./40.],
          [35./384., 0., 500./1113., 125./192., -2187./6784., 11./84., 0.]])
   E=B[1,:]-B[0,:]

   Y=mat(zeros((7,y.shape[1])))
   Y[0,:] = y
   g0=linalg.norm(my_grad(y,Patch))
   done=0
   path=mat(y)
   reject_patch = 0
   while done !=1:
      accept=0
      while accept==0:
         Y[1,:]=Y[0,:]+h*A[1,0]*my_grad(Y[0,:],Patch)
         Y[2,:]=Y[0,:]+h*A[2,0]*my_grad(Y[0,:],Patch)+h*A[2,1]*my_grad(Y[1,:],Patch)
         Y[3,:]=Y[0,:]+h*A[3,0]*my_grad(Y[0,:],Patch)+h*A[3,1]*my_grad(Y[1,:],Patch
                )+h*A[3,2]*my_grad(Y[2,:],Patch)
         Y[4,:]=Y[0,:]+h*A[4,0]*my_grad(Y[0,:],Patch)+h*A[4,1]*my_grad(Y[1,:],Patch
                )+h*A[4,2]*my_grad(Y[2,:],Patch)+h*A[4,3]*my_grad(Y[3,:],Patch)
         Y[5,:]=Y[0,:]+h*A[5,0]*my_grad(Y[0,:],Patch)+h*A[5,1]*my_grad(Y[1,:],Patch
                )+h*A[5,2]*my_grad(Y[2,:],Patch)+h*A[5,3]*my_grad(Y[3,:],Patch)+h*A[
                5,4]*my_grad(Y[4,:],Patch)
         Y[6,:]=Y[0,:]+h*A[6,0]*my_grad(Y[0,:],Patch)+h*A[6,1]*my_grad(Y[1,:],Patch
                )+h*A[6,2]*my_grad(Y[2,:],Patch)+h*A[6,3]*my_grad(Y[3,:],Patch)+h*A[
                6,4]*my_grad(Y[4,:],Patch)+h*A[6,5]*my_grad(Y[5,:],Patch)

         y_err=y+h*B[0,0]*my_grad(Y[0,:],Patch)
         y_new=y+h*B[1,0]*my_grad(Y[0,:],Patch)
         for i in range(1,7):
            y_err = y_err+h*B[0,i]*my_grad(Y[i,:],Patch)
            y_new = y_new+h*B[1,i]*my_grad(Y[i,:],Patch)
         f_cur = my_fun(y_new,Patch)
         eps=linalg.norm(y_new-y_err)/linalg.norm(y_new);
         reject=0;

         if eps > Patch.ETOL_RK:
            h=.8*h*(Patch.ETOL_RK/eps)**(1./5.)
            reject=1
         else:
            [shrink,done,edge]=check_bounds(y_new,Patch.b)
            if shrink == 1:
               h=.5*h
               reject=1

         if reject == 0:
            accept = 1
            f=bmat('f;f_cur')
            path=bmat('path;y_new')
            y=mat(y_new)
            Y[0,:]=y
            h=.8*h*(Patch.ETOL_RK/eps)**(1./5.)
            gcur=linalg.norm(my_grad(y,Patch))
            if gcur < Patch.GTOL_RK:
               [f1,g1,H] = f_newt(y_new,Patch)
               [D,V]=linalg.eig(H)
               if all(D>0):
                  done=1
               else:
                  done=0

         if h < Patch.RK_hmin:
    done = 1
    accept = 1
    reject_patch = 1

   return path, f, edge, reject_patch


#checks that the RK step doesn't cross the boundary or that it should terminate
def check_bounds(y_new,bounds):
   shrink=0
   done=0
   edge=mat(zeros((1,y_new.shape[1])))
   past_bounds=mat(zeros((1,y_new.shape[1])))
   b1=.05*(bounds[:,1]-bounds[:,0])
   bottom, top = bounds[:,0]+b1, bounds[:,1]-b1
   bounds1 = bmat('bottom top')
   for i in range(0,y_new.shape[1]):
if y_new[0,i]<bounds1[i,0]:
   edge[0,i]=-1
if y_new[0,i]>bounds1[i,1]:
   edge[0,i]=1
if y_new[0,i]<bounds[i,0]:
   past_bounds[0,i]=-1
if y_new[0,i]>bounds[i,1]:
   past_bounds[0,i]=1
   if all(past_bounds==0)==False:
      shrink=1
   elif all(edge==0)==False:
      done=1
   return shrink,done,edge
```

```
#braindead Newton’s method
#other functions are in this module
#Hess is a function whose definition is in the module mf
#[fout,gout,hout]=Hess(x)
#x is a row vector!
def newton(x,Patch):
   tol=10.**-12
   [fout,gout,hout] = f_newt(x,Patch)
   while linalg.norm(gout) > tol:
      s=linalg.solve(hout,-gout)
      x=x+s.transpose()
      [fout,gout,hout] = f_newt(x,Patch)
   return x, fout



#Gives the proper value of the energy at each gridpoint
def Gau_Evals(x,Patch):
    F=mat(genfromtxt('Data%s.txt' % Patch.p, delimiter=' '))
    f=mat(zeros((x.shape[0],1)))
    kk=0
    for j in x:
        ind=-3
        cc=0
        for i in F:
            if linalg.norm(i[0,:Patch.d]-j[0])<10.**-4:
                ind = cc
            cc = cc + 1
        f[kk] = F[ind,Patch.d+Patch.s]
        kk = kk+1
    return f

#interpolation of the energy function
def my_fun(x,Patch):
    [f1,g1]=ddim_smol(x,Patch)
    return f1

#-grad of the energy function
def my_grad(x,Patch):
    Patch.k=Patch.k+1
    [f1,g1]=ddim_smol(x,Patch)
    g1=-g1
    Patch.k=Patch.k-1
    return g1
```

```
#adds a finite difference Hessian to a single point call of the Smolyak interpolation
#also converts the gradient to a column vector.
def f_newt(x,Patch):
   [fout]=my_fun(x,Patch)
   [gout]=my_grad(x,Patch)
   gout=gout.transpose()

   h=10.0**-6
   hout= h*mat(eye(3))
   for i in range(0,x.shape[1]):
      gout1=my_grad((x+hout[:,i].transpose()),Patch)
      hout[:,i]=gout1.transpose()
   hout=hout-tile(gout,(1,x.shape[1]))
   hout=-hout/h
   gout=-gout
   return fout,gout,hout
```

The file math_vars.py contains the math variables that a mathematician may want to edit in the simulation, but these are not intended for all users.

```
#! usr/local/apps/python-2.6.5/bin/python
#Math variables for light-induced simulation




#This is the order of the interpolation error. The surrogate will be exact for degree k-1
k = 3

#This is the gradient value where RK45 switches to Newton/Quasi-Newton
GTOL_RK = 10**(-3.0)

#Tolerance for RK45 accuracy
ETOL_RK = 10**(-5.0)

#1/2 the Maximum size of a patch
H_max = 100

#1/2 the minimum size of a patch
H_min = 1

#minimum h for RK45
RK_hmin = 10**(-10)

#Maximum number of Gaussian iterations before shrinking the patch
gau_max = 12

#Error tolerance for patch accuracy
ETOL_sim = 10.0**(-4.0)

#Time requested from LSF for full simulation
sim_time = 10000
```

```
#Time requested from LSF for gaussian jobs
gau_time = 1000

#number of available processors for jobs. Allows for scaling studies
q_size = 240

#Parameter for growing the trust-region
TR_grow = 2

#Parameter for shrinking the trust-resion
TR_shrink = .5

#accuracy limit to grow the trust-region
TR_accg = .95

#accuracy limit to shrink the trust-region
TR_accs = .85

#Fudge factor for determining the next patch-size
Fudge_sim = .7

#Check the exact error along the simulation path (this doubles the simulation time!)
Check_Exact = 0

#1/2 the initial patch size
H0 = 15

#parameter for shifting the patch when a boundary is hit
shift_dist = .8

#1 for trust-region, 2 for Runge-Kutta
error_control = 2
```

The file chem_vars.py contains the inputs that any user should be prepared to edit in order to run this simulation.

```
#! usr/local/apps/python-2.6.5/bin/python
#chemistry inputs for light-induced simulation

#number of design variables
d = 3

#name of your molecule
molecule = 'Butene'

#the number of atoms in your molecule
num_atoms = 12

#the number of cores for your gaussian jobs
n_proc = 2

#method of energy computation (hf,b3lyp,...)
```

```
method = 'b3lyp'

#basis-set for you gaussian jobs (cep-31g,CBSB7,...)
basis_set = 'cep-31g'

#Gaussian job type (opt=modredundant) and anything else you want in the header
calculation = 'opt=modredundant'

#state transitions for your simulation
state_order = [1,0]

#where will your gaussian jobs be run?
queue = 'gto'

#set this to 1 if you want every z-matrix that was calculated during your sim
all_zmats = 0

#atom number, value, ...
bond_lengths = []

#for reading output files, i.e. 'C1-C2'
bond_names = []

#atom numbers, value, ...
valence_angles = []

#for reading output files, i.e. 'C1-C2-C6'
valence_names = []

#atom numbers, value, ...
torsion_angles = ['1 2 6 8', 0., '6 2 1 5', 120., '2 6 8 11', -120.]

#for reading output files, i.e. 'C1-C2-C6-C8'
torsion_names = ['C1-C2-C6-C8', 'C6-C2-C1-H5', 'C2-C6-C8-H11']

#pre-processed variable, pre-processed to whom, amount to pre-process
deps = ['1 2 6 9', 0, 180., '8 6 2 7', 0, 180., '9 6 2 7', 0, 0.]

#where your molecule sits, this directory must have initial zmat as zmat.gjf
home_dir = '/home/dsmokrau/%s' % molecule

#where to write all your files
share_dir =  '/kelley_data/dsmokrau/db_LITES'
```

The file LITES.py is the main simulation which calls the above modules.

```
#! usr/local/apps/python-2.6.5/bin/python
# LIGHT INDUCED TRANSITION EFFECTS SIMULATOR

from numpy import *
import shutil, os, sys, time, string
import LITES_defs as ld

Sim = ld.Simulation()  # all the initial values and sim stuff is in this class

shutil.copyfile('g_sub', '%s/g_sub' % Sim.share_dir)
shutil.copyfile('zmat.gjf', '%s/zmat.gjf' % Sim.share_dir)
sys.path.insert(0,'%s' % Sim.share_dir)
os.chdir('%s' % Sim.share_dir)


cur_Patch = ld.Patch(0, Sim.bounds0, Sim.state_order[0], Sim.init_point, 0, Sim.point_list)
print('cur_Patch is number %s' % cur_Patch.p)
keep = 0
ld.Sim_Sum(Sim,cur_Patch,0,0,0)

for state in Sim.state_order:
   at_min = 0
   if state == Sim.state_order[-1]:
next_state = 0
   else:
next_state = Sim.state_order[Sim.state_order.index(state)+1]

   while at_min == 0:
if keep == 0:
   [max_iters, patch_fin] = ld.run_patch(Sim,cur_Patch)
else:
   patch_fin = 1

    if patch_fin == 0:
   ld.Sim_Sum(Sim,cur_Patch,patch_fin,0,max_iters)
   bd, bu = cur_Patch.b[:,0]+(cur_Patch.b[:,1]-cur_Patch.b[:,0])/4., cur_Patch.b[:,1]-(cur_Patch.b[:,1]-\
cur_Patch.b[:,0])/4.
   bounds = bmat('bd bu')
   cur_Patch = ld.Patch(cur_Patch.p+1, bounds, state, cur_Patch.i, cur_Patch.zmat, Sim.point_list)
   print('cur_Patch is number %s' % cur_Patch.p)
   os.system('rm %s_sp*' % Sim.molecule)
   keep = 0

else:
   Sim.conv_patches.append(cur_Patch.p)
   [init_point, edge, at_min, nearby, p_energy, o_energy, int_err] = ld.Patch_Path(cur_Patch,Sim)
   zmat = ld.read_zmat('%s_sp%s.log' % (Sim.molecule, nearby), Sim.num_atoms)

   keep = 0
   if at_min == 0:
if Sim.ec == 1:
   pass
    elif Sim.ec == 2:
   new_bounds = ld.RK_approach(cur_Patch,int_err,max_iters, init_point, edge)
   ld.Sim_Sum(Sim,cur_Patch,patch_fin,at_min,max_iters)
   cur_Patch = ld.Patch(cur_Patch.p+1, new_bounds, state, init_point, zmat, Sim.point_list)
   print('cur_Patch is number %s' % cur_Patch.p)
   os.system('rm %s_sp*' % Sim.molecule)


   if at_min == 1 and state!=Sim.state_order[-1] and Sim.ec == 2:
edge = mat(zeros((1,cur_Patch.d)))
print('checking if we keep the patch')
int_err = ld.keep_patch(cur_Patch,next_state)
new_bounds = ld.RK_approach(cur_Patch,int_err,max_iters, init_point, edge)
print('state = %s' % cur_Patch.s)
ld.Sim_Sum(Sim,cur_Patch,patch_fin,at_min,max_iters)
if (new_bounds[0,1]-new_bounds[0,0])/2. > cur_Patch.H:
   keep = 1
   print('Keep the Patch')
   cur_Patch = ld.Patch(cur_Patch.p+1, cur_Patch.b, next_state, init_point, zmat, Sim.point_list)
   print('cur_Patch is number %s' % cur_Patch.p)
   os.system('cp Data%s.txt Data%s.txt' % (cur_Patch.p-1,cur_Patch.p))
else:
   keep = 0
   print('discarding the patch, changing to state %s' % next_state)
   cur_Patch = ld.Patch(cur_Patch.p+1, new_bounds, next_state, init_point, zmat, Sim.point_list)
   print('cur_Patch is number %s' % cur_Patch.p)
```

The file g_sub is the generic submission file for Gaussian 09 jobs.

```
#!/bin/csh
source /usr/local/apps/env/g09.csh

#BSUB -W time
#BSUB -R em64t
#BSUB -J E_files
#BSUB -n n_proc
#BSUB -q queue
#BSUB -o out
#BSUB -e err

mkdir scratch
cp gjf scratch
cd scratch

g09 < gjf > log

cp log share
cd
rm -r scratch
```

The file LITES_in submits the entire simulation.

```
#!/bin/csh
```

```
#BSUB -W 10000
#BSUB -R em64t
#BSUB -x
#BSUB -n 1
#BSUB -q gto
#BSUB -o /kelley_data/dsmokrau/db_LITES/LITES.out
#BSUB -e /kelley_data/dsmokrau/db_LITES/LITES.err

rm -r /scratch/dsmokrau

mkdir /scratch/dsmokrau
cp /home/dsmokrau/LITES/* /scratch/dsmokrau
cd /scratch/dsmokrau

/usr/local/apps/python-2.6.5/bin/python LITES.py

cd
rm -r /scratch/dsmokrau
```