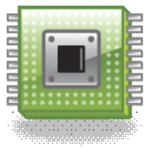
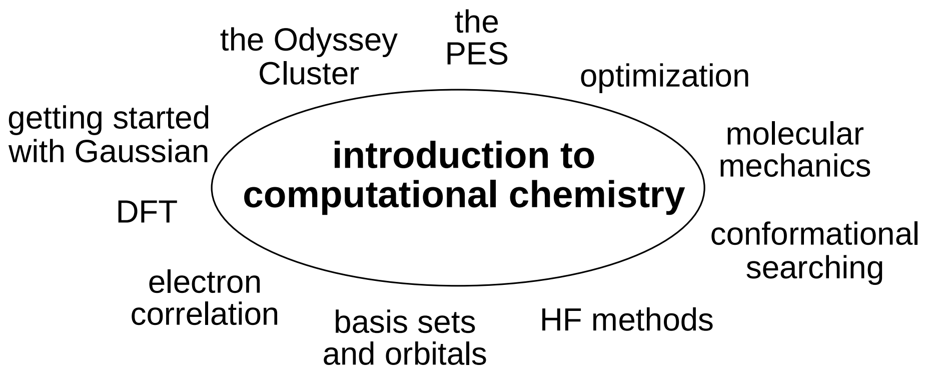
Introduction to Computational Chemistry

Eugene E. Kwan

February 22, 2010.



Scope of Lecture



Key References

1. Molecular Modeling Basics Jensen, J.H. CRC Press, **2009**.

2. Computational Organic Chemistry Bachrach, S.M. Wiley, **2007**.

3. Essentials of Computational Chemistry: Theories and Models (2nd ed.) Cramer, C.J. Wiley, **2004**.

4. Calculation of NMR and EPR Parameters: Theory and Applications Kaupp, M.; Buhl, M.; Malkin, V.G., eds. Wiley-VCH, **2004**.

I thank Professor Jensen (Copenhagen) for providing some useful material for this lecture.

Key Questions

(1) What kinds of questions can computational chemistry help us answer?

- **Mechanistic:**
  What are the intermediates and transition states along the reaction coordinate?

  What factors are responsible for selectivity?

  As molecules pass from reactants to products, do they stay along the minimum energy path?

- **Physical:**
  What is the equilibrium geometry of a molecule?

  What will the spectra of a molecule look like (IR, UV-vis **NMR**, etc.)? What do the lines represent?

- **Conceptual:**
  Where are the charges in a molecule? What do its orbitals look like? Where are the electrons?

  Why are some molecules more stable than others? Is a molecule aromatic? What are the important hyperconjugative interactions in a molecule?

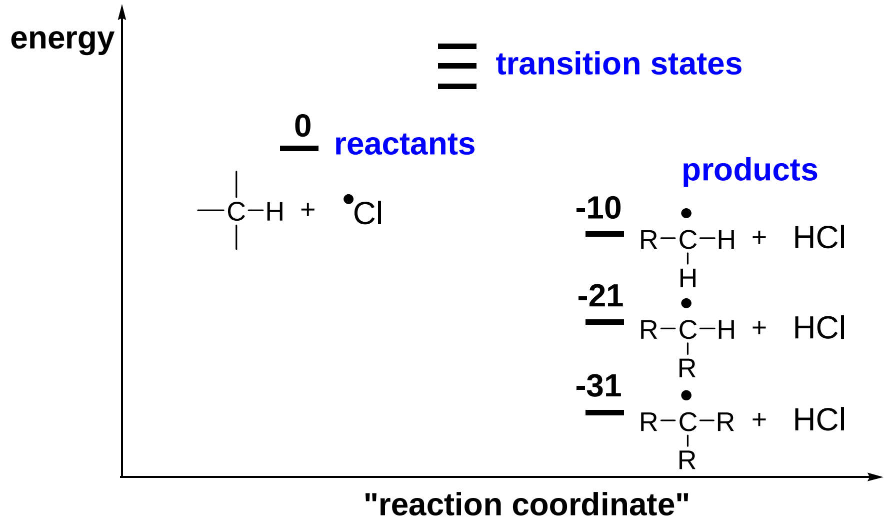
(2) How are potential energy surfaces (PESs) studied?
How do I locate ground states and transition states?

(3) How is energy evaluated?
What are the differences between HF, DFT, MP2, etc?
When is each method applicable?

(4) How do I perform calculations here at Harvard?
How do I use Gaussian?
What do I need to do to get started on the Odyssey Cluster?

The Potential Energy Surface (PES)

In every introductory organic textbook, you see diagrams like:

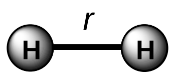


Here are some important questions that you should ask:

- What is a "reaction coordinate," exactly?
- What does a transition state look like?
- Where do the numbers for these energies come from?

I can't answer all of these questions here, but **the primary purpose of computational chemistry is to connect experimental results with a theoretical potential energy surface** so we can understand nature. One can ask static questions like "What is the equilibrium geometry for this molecule?" or dynamic ones like "What is the mechanism of this reaction?".

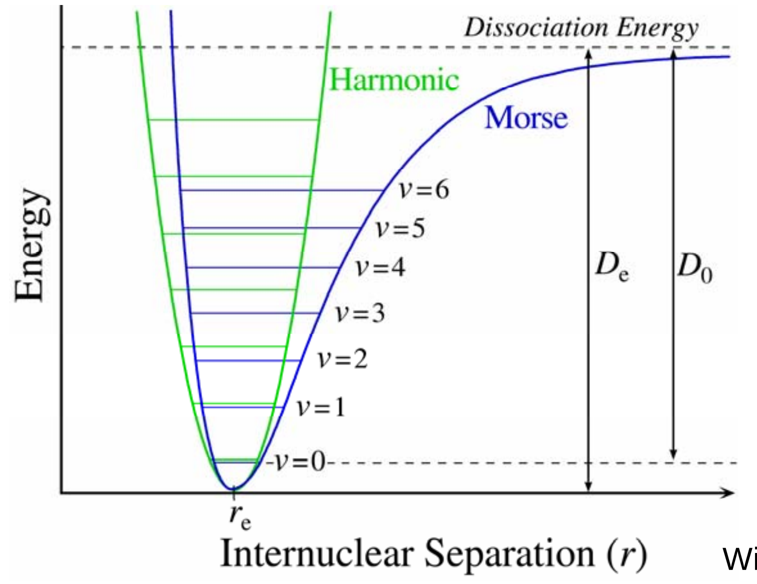
What exactly is a potential energy surface (PES)? Consider dihydrogen:



How many variables do I need to describe the geometry of dihydrogen? Six. In Cartesian coordinates, I could say that the energy is a function of all the nuclear coordinates:

$$E(q) = E(x_1, y_1, z_1, x_2, y_2, z_2)$$

Now, if you're sharp, you can point out that we don't really care where the center of mass is, but what we really care about in terms of chemistry, is simply the bond length *r*. So in some sense, the PES is single-dimensional. The typical appearance of such potentials for bonds is something like:



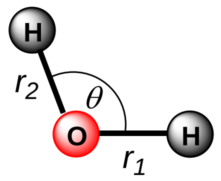
Wikipedia

Note that at the bottom of the well, the PES is described well by a **harmonic oscillator**--a quadratic function. This is an important approximation that is made in many calculations.

But, wait! Where are the electrons? Why aren't we giving them coordinates? It is a basic assumption of standard computational techniques that because the nuclei are much heavier than the electrons, the nuclei are essentially "frozen" from the point of view of the electrons. This is the **Born-Oppenheimer approximation.** From quantum mechanics, we know that the electrons are described by wave functions, rather than specific position and momentum coordinates as in classical mechanics.

The Potential Energy Surface (PES)

Now, a more complicated molecule will necessarily require more coordinates. For example, you can think of water as needing two O-H bond lengths and the H-O-H angle:



If you want to characterize the *entire* PES, and you want to do 10 points per coordinate, that would make a thousand points total. Since evaluating the energy as a function of geometry requires minutes if not hours per point, this is clearly going to be impractical for all but the exceptionally patient (and we're not).

To summarize, the PES is a multidimensional function. It takes the molecular geometry as input, and gives the energy as output. It has a lot of dimensions and is very, very complicated.

Fortunately, **transition state theory** tells us that we only need to characterize a few **stationary points** on the PES, rather than the entire thing. A stationary point is anywhere the **gradient** is zero. For dihydrogen:

$$\frac{\partial E}{\partial x_1} = \frac{\partial E}{\partial x_2} = \frac{\partial E}{\partial y_1} = \cdots = 0$$

This is the multidimensional analog of the derivative. For example, consider this function:

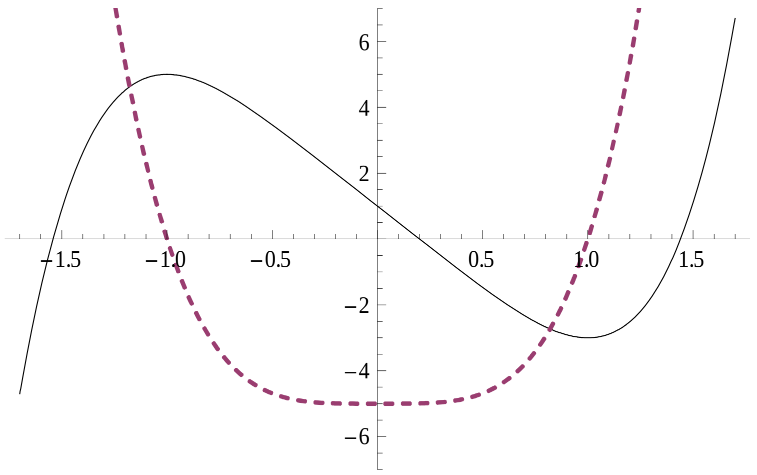
$$y = x^5 - 5x + 1$$

From high school calculus, you know that the derivative is:

$$\frac{dy}{dx} = 5x^4 - 5$$

The function has a local maximum and a local minimum, which occur when dy/dx is zero:

solid = function; dashed = derivative



If we want to know whether a particular stationary point is a local minimum or maximum, we compute the second derivative, which is positive for minima, 0 for asymptotes, and negative for maxima. For multidimensional functions, the analog of the second derivative is called the **Hessian**:

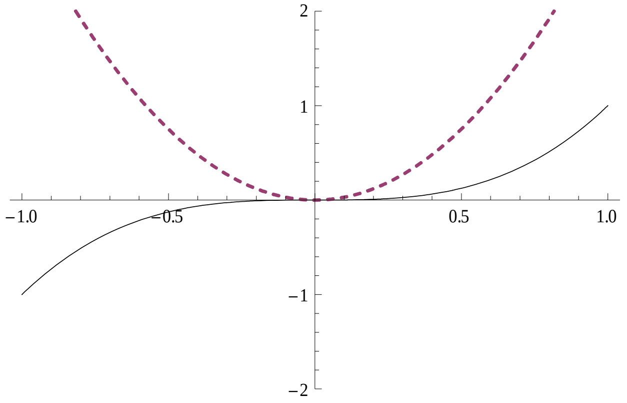
$$H(f) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1\,\partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1\,\partial x_n} \\ \frac{\partial^2 f}{\partial x_2\,\partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2\,\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n\,\partial x_1} & \frac{\partial^2 f}{\partial x_n\,\partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix}.$$

In chemistry, we are particularly interested in energy minima. A stationary point is a local minimum if the Hessian is postive-definite there. To evaluate this, one transforms from Cartesian to normal coordinates, which amounts to diagonalizing the Hessian or "performing a frequency analysis." The "lack of any imaginary frequencies" means that the matrix is positive definite.

Lecture 9: Introduction to Computational Chemistry

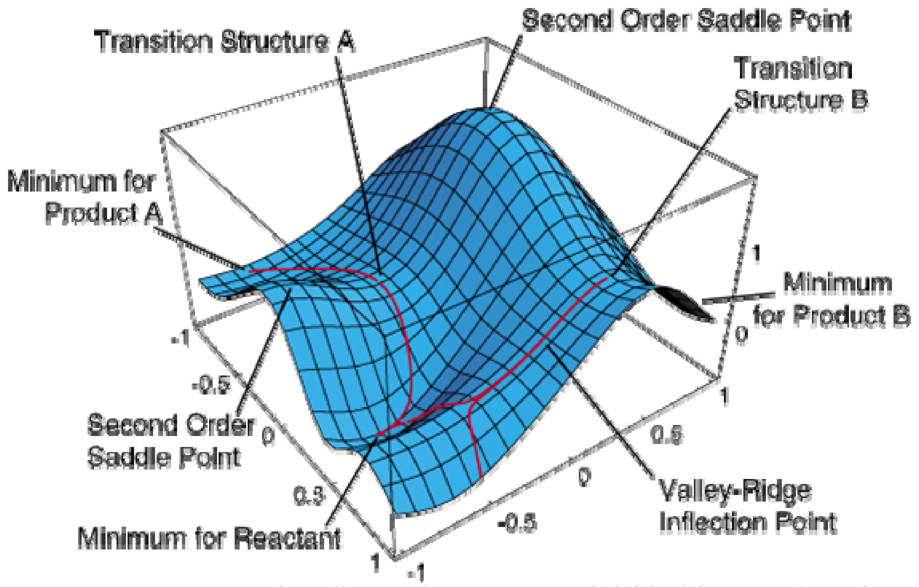
The Potential Energy Surface (PES)

Can one have a stationary point that is neither a local maximum nor a local minimum? Certainly: **saddle points**. Consider $y=x^3$:



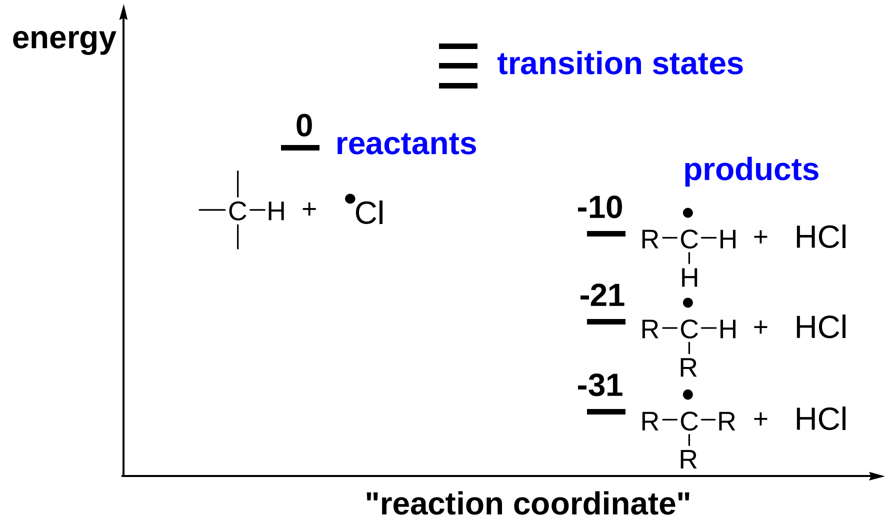
starting materials and products correspond to local minima (stationary points with zero imaginary frequencies) while

transition states correspond to first-order saddle points (stationary points with exactly one imaginary frequency).



http://www.chem.wayne.edu/~hbs/chm6440/PES.html

The **minimum energy path**, **reaction coordinate**, or **intrinsic reaction coordinate** connect the starting materials and products. Thus, to understand a reaction, we will need to locate a minimum of three stationary points: one each for the starting material, transition state, and product.



The energy difference between the reactant and transition state will tell us about the rate of the reaction; the difference between the reactant and product will tell us about how exothermic the reaction is. We can also examine the geometry of the molecules at all the stationary points, and then draw some conclusions about reactivity, selectivity, or other trends.

Q: How do I know if the answers that the computer gives actually mean something real?

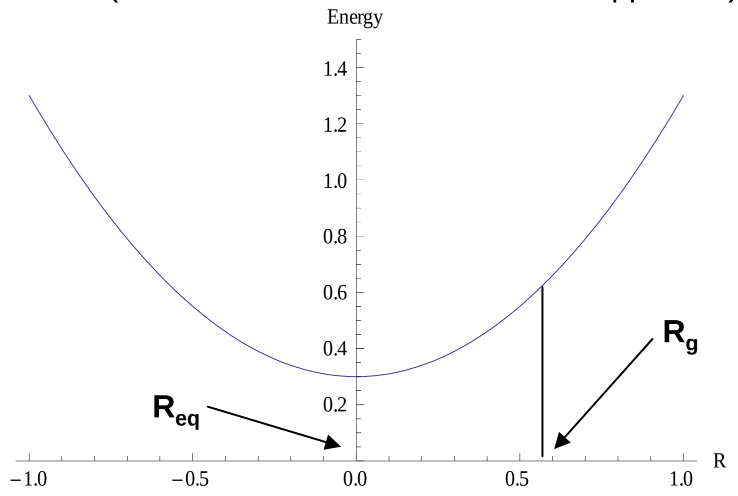
It is **very important** to be considering this question at all times when dealing with computations. To fit into the scientific method, a computation must make a tangible prediction that can be verified by experiment. One can now predict geometries (X-ray), reaction barriers (kinetics), spectroscopic properties (IR and NMR), etc. As it turns out, in many cases, the agreement between theory and experiment is now **very good**.

Energy Minimization

Q: How do we locate the stationary points?

The general approach is: (1) Make a guess for the geometry of a molecule at its stationary point; (2) Use a computer program to move the atoms in such a way that the gradient drops to zero; and (3) Perform a frequency analysis to verify the nature of the stationary point is correct.

For now, let us consider step (2); the optimization process. Imagine we have a one-dimensional, quadratic PES with coordinate *R* (discussion borrowed from Jensen, pp. 8-12):



The energy E of this harmonic oscillator potential is given by:

$$E = \frac{1}{2}k(R - R_{eq})^2$$

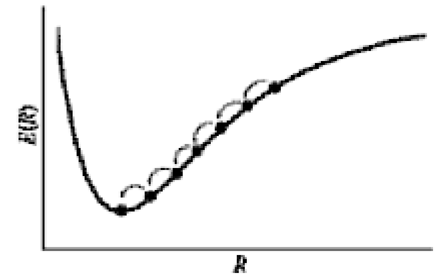
Taking the derivative, we have:

$$\frac{dE}{dR} = k(R - R_{eq})$$

Clearly, if R is not the equilibrium value, R_{eq}, then the gradient is non-zero, and we are not at a stationary point. But we can rearrange this equation to see how to get there:

$$R_e = R_g - \frac{1}{k}\left(\frac{dE}{dR}\right)_{R_g} = R_g + \frac{1}{k}F$$

Here, R_g is some guess for the value of R_{eq}. F is the **force**, or negative gradient. That means that if we know what the spring constant *k* is, we can get to the stationary point in one step. However, we usually don't know this, so we take small scaled steps in the direction of the gradient until it falls below some value. This is the **method of steepest descent**.



Clearly, we want to be taking as few steps as possible. For a harmonic oscillator, *k* is the derivative of the gradient:

$$R_e = R_g - \left(\frac{d^2E}{dR^2}\right)_{R_g}\left(\frac{dE}{dR}\right)_{R_g}$$

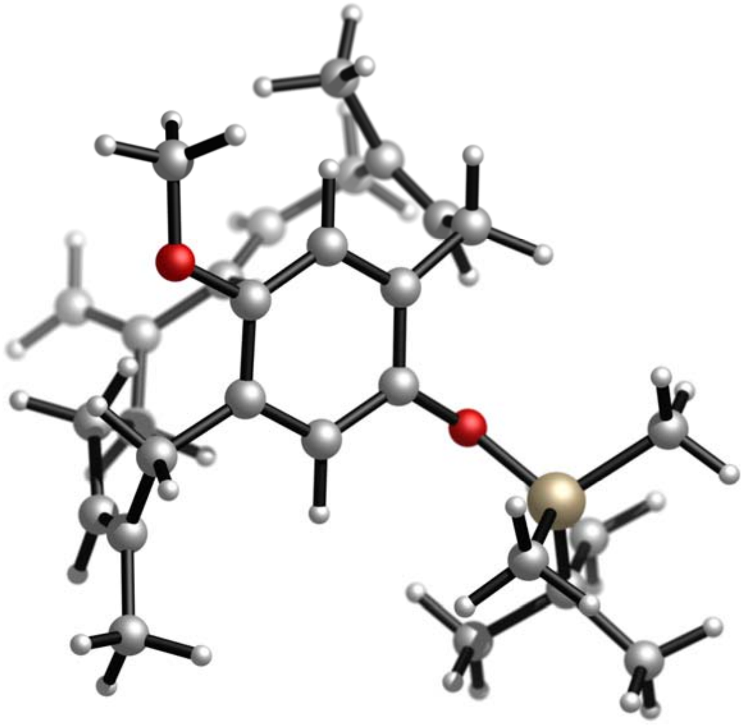
This works great if we are near a quadratic portion of the PES. In reality, many PESs have very flat regions where this kind of **Newton-Raphson** step will be too large. (An excellent animation of how this works can be found on Wikipedia at http://en.wikipedia.org/wiki/File:NewtonIteration_Ani.gif.) Thus, most programs will scale back quadratic steps when their size exceeds some pre-determined threshold.

Note that this method requires the Hessian, which is clearly expensive to calculate. In many cases, one can use approximate methods to make a good guess at what the Hessian looks like, and then update the guess on every iteration.

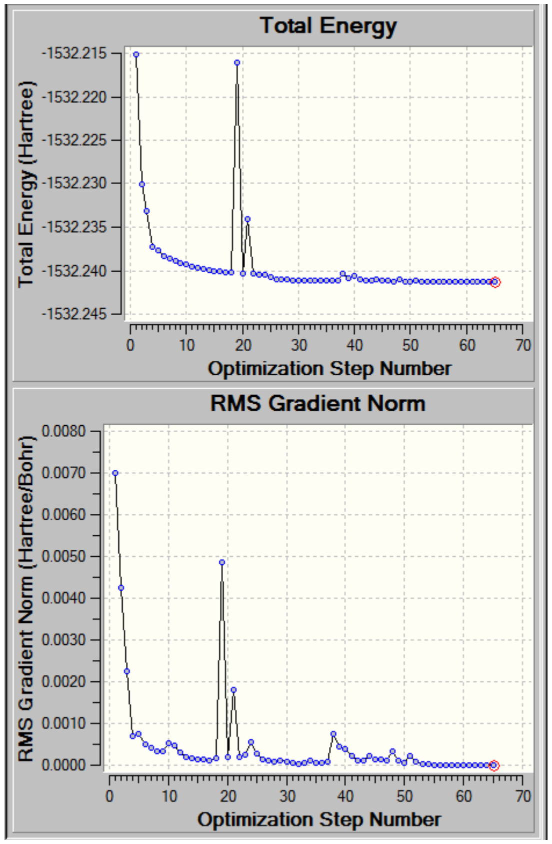
Energy Minimization

In practice, neither of these methods is satisfactory and various improvements have been made. In Gaussian 09, the Berny algorithim using GEDIIS in internally redundant coordinates is used by default. For details, see http://www.gaussian.com/g_tech/g_ur/k_opt.htm. Internally redundant coordinates means that instead of using the Cartesian coordinate system, which has certain mathematical pathologies, a more natural set of coordinates involving bond lengths, bond angles, and dihedral angles is used. This has been shown to be faster for many organic molecules.

Let's take a look at how this works in real life. My co-worker Joe Wzorek and I are interested in the conformations of macrocycles. One structure we need the equilibrium geometry and energy of is a conformer of an intermediate used by Professor Shair along his route towards longithorone A:



Gaussian plots the energy and gradient of each geometry optimization step:



(1) Energy is given in hartrees. 1 hartree = 627.509 469 kcal.

(2) The energy drops quickly at first and then slowly converges. Seeing the energy spike in the middle is common; sometimes the optimizer will get off track. By its nature, optimization is a chaotic phenomenon, and can show high sensitivity to initial conditions, oscillatory behavior, or other pathologies.

(3) The gradient usually tracks with the energy, but not always. When it reaches a certain threshold, the job is finished.

Energy Minimization

By the way, one of the nicest programs for rendering structures, as shown on the previous slide is CYLView, is available from Professor Legault (Sherbrooke) at www.cylview.org.

We can learn more by looking at the raw output file (.out). The fundamental file format in computational chemistry is text, so it is useful to learn some techniques for handling large amounts of text. Virtually all scientific computing is done in the magical land of Linux, so we will start there:

```
[ekwan@iliadaccess03 output]$ grep -A 3 "Maximum Force"
longithorone_OPLS_low_4.out | more

 Maximum Force            0.037142     0.000450     NO
 RMS     Force            0.004970     0.000300     NO
 Maximum Displacement     0.948905     0.001800     NO
 RMS     Displacement     0.234625     0.001200     NO
--
 Maximum Force            0.016448     0.000450     NO
 RMS     Force            0.002426     0.000300     NO
 Maximum Displacement     1.085268     0.001800     NO
 RMS     Displacement     0.284053     0.001200     NO
...
```

Here, I have issued a command to search for the words "Maximum Force" in the file longithorone_OPLS_low_4.out. Specifically, I want all the instances of that phrase, along with the three lines after it. The "more" command tells it to give me the output page by page (quite a few optimization steps were required, so there is a lot of output).

The first column of numbers is the value of the parameter in the current iteration; the second is the desired threshold value. As you can see, the first two steps were quite far away from equilibrium.

RMS means "root mean square," which is basically an average that works on signed quantities. Because there are a lot of atoms, and each one has a particular gradient associated with it, we need the RMS gradient. Remember, gradient is the negative of force. Displacement is something that tells you how much the atoms moved between the last iteration and this one. Occasionally, the displacement will not converge, but the forces will go to zero. In that case, the optimization will automatically terminate.

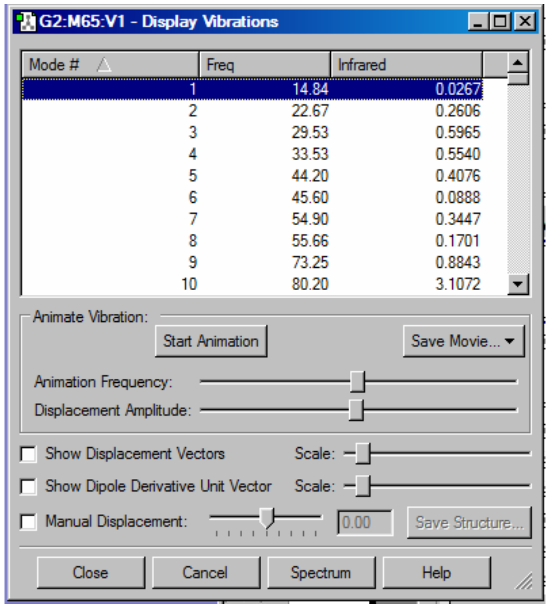
```
  [ekwan@iliadaccess03 output]$ grep -B 7 Stationary
  longithorone_OPLS_low_4.out
           Item               Value     Threshold  Converged?
   Maximum Force            0.000008     0.000450     YES
   RMS     Force            0.000001     0.000300     YES
   Maximum Displacement     0.001556     0.001800     YES
   RMS     Displacement     0.000316     0.001200     YES
   Predicted change in Energy=-5.392921D-09
   Optimization completed.
      -- Stationary point found.
```

In this case, everything converged after 64 iterations. What is the energy? At each step, the output file contains the energy:

```
[ekwan@iliadaccess03 output]$ grep "SCF Done"
longithorone_OPLS_low_4.out
 SCF Done:  E(RB3LYP) =  -1532.21514319 A.U. after   12 cycles
 SCF Done:  E(RB3LYP) =  -1532.23006873 A.U. after   11 cycles
...
 SCF Done:  E(RB3LYP) =  -1532.24125618 A.U. after    5 cycles
 SCF Done:  E(RB3LYP) =  -1532.24125618 A.U. after    1 cycles
```

Notice that the energy starts out high, and then decreases. E(B3LYP) tells you that this a density functional method was used to get at the energy--more on this in a moment. The last energy is the energy of the molecule in the geometry of this particular stationary point. Specifically, this is the **electronic energy**. Roughly speaking, this is the energy it takes to bring together all the atoms and electrons, which is why it looks like a huge negative number. Every geometry step involves an energy evaluation, which is itself an iterative process. As we get closer and closer to equilibrium, the energy evaluation process speeds up and requires fewer cycles.

Of course, we need to verify this structure is a true local minimum with a frequency analysis. (There is no way to know if it is the *global* minimum. However, there are methods I will discuss shortly that can give one some confidence the global minimum, or something near it, has in fact been found). GaussView can display all the normal modes (Results... Vibrations):


G2:M65:V1 - Display Vibrations

Mode #	Freq	Infrared
1	14.84	0.0267
2	22.67	0.2606
3	29.53	0.5965
4	33.53	0.5540
5	44.20	0.4076
6	45.60	0.0888
7	54.90	0.3447
8	55.66	0.1701
9	73.25	0.8843
10	80.20	3.1072

The fact that all the frequencies are positive numbers indicates that this is a true local minimum. This is also reflected in the thermochemical energy output:

```
[ekwan@iliadaccess03 output]$ grep -4 Gibbs
longithorone_OPLS_low_4.out

 Zero-point correction=                   0.636208 (Hartree/Particle)
 Thermal correction to Energy=                    0.672630
 Thermal correction to Enthalpy=                  0.673575
 Thermal correction to Gibbs Free Energy=         0.567792
 Sum of electronic and zero-point Energies=           -1531.605048
 Sum of electronic and thermal Energies=              -1531.568626
 Sum of electronic and thermal Enthalpies=            -1531.567682
 Sum of electronic and thermal Free Energies=         -1531.673465
```

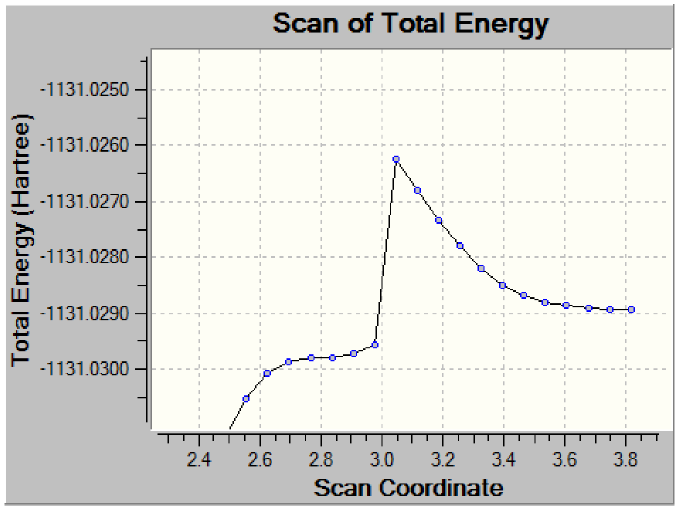
These corrections take into account the fact that, above absolute zero, various vibrational and rotational energy levels get populated, and therefore contribute to the energy. The very last line represents the Gibbs free energy of this compound in this geometry, at least as estimated by Gaussian using this particular method.

If there had been any imaginary frequencies, there would have been a message like "1 imaginary frequency ignored." This would mean a transition state.

Constrained Optimizations and Transition States

Because these are still stationary points, one might think that the same optimization methods would apply. Unfortunately, they seem to be much better at optimizing downwards towards a ground state than upwards towards a transition state. As a result, unless the starting structure is already very close to the transition state stationary point, it will optimize away from it.

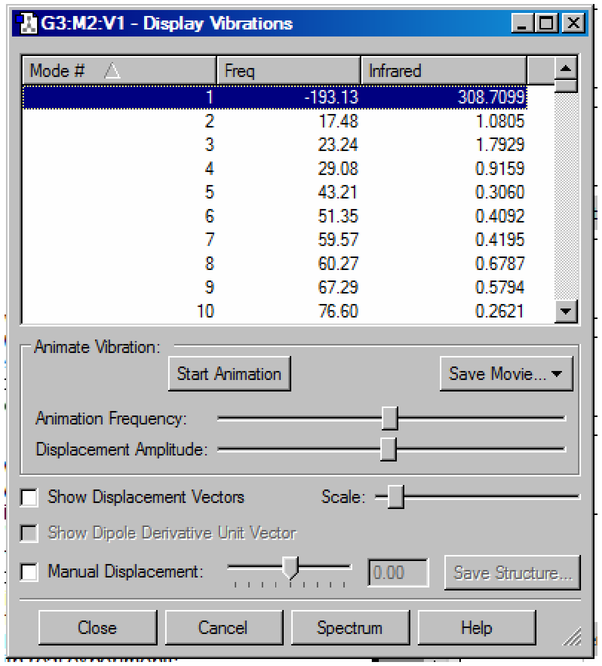
The best strategy for getting a starting structure close to a TS is to perform a "scan" in which the forming bond distance is fixed at certain values, while all the other internally redundant coordinates are allowed to relax. Here is a scan I performed to find the transition state for a Michael reaction:



Constrained Optimizations and Transition States

The scan began with the two molecules quite separated, and then the forming bond distance was fixed at around 3.8 A and the structure was optimized. The resulting pseudo-stationary point was then perturbed slightly to bring the reacting partners 0.07 A closer together (the "step size"), and then a new constrained optimization was performed. In such a procedure, one hopes that the scan will reach an energy maximum, which will turn out to be the transition state.

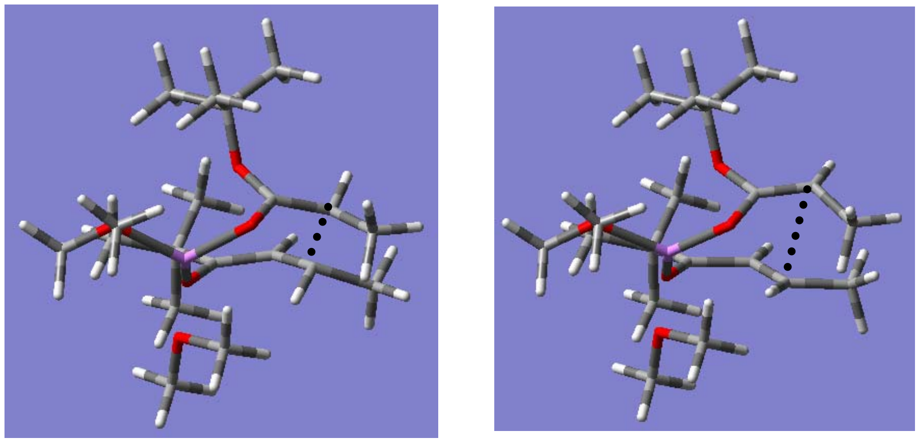
As it turned out in this case, the scan *did* reach a maximum, but then the energy dropped considerably. This is a consequence of the step size being too large--the perturbation between steps was enough to cause the structure to "kink" into a much lower energy conformation. This turned out to be of no consequence, however, as a finer step size, followed by unconstrained transition optimization eventually resulted in the transition state. This time, the frequency analysis shows one negative frequency: that of the forming bond:


G3:M2:V1 - Display Vibrations

Mode #	Freq	Infrared
1	-193.13	308.7099
2	17.48	1.0805
3	23.24	1.7929
4	29.08	0.9159
5	43.21	0.3060
6	51.35	0.4092
7	59.57	0.4195
8	60.27	0.6787
9	67.29	0.5794
10	76.60	0.2621

Animate Vibration:
Animation Frequency:
Displacement Amplitude:
Show Displacement Vectors
Scale:
Show Dipole Derivative Unit Vector
Manual Displacement: 0.00

One can animate these normal modes to verify that the negative frequency corresponds to the desired bond formation event:



A more rigorous analysis would involve an **intrinsic reaction coordinate (IRC)** scan, which would take the transition state and follow the minimum energy path in both directions to check that the observed transition state actually connects the desired starting materials and products.

Other transition state search methods are available that try to connect a starting material structure with a product structure. These can work well in some cases, but are not as reliable.

You may notice that I say "observed" as if this sort of thing were experimental in nature. While this is not strictly speaking, from a philosophical perspective, true, it is true in practice. As I often say, like anything to do with computers, computational chemistry obeys the rule "garbage in, garbage out." One has to ask a meaningful question to get a meaningful result. Even then, apparently logical computations (optimizations) will often fail to converge, or may give non-sensical results. One must always use chemical intuition to see if the results being found actually make sense, and then rigorously follow up the theoretical calculations with real experiments.

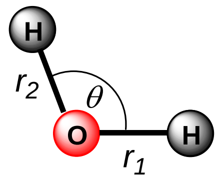
Molecular Mechanics

Q: How are geometries turned into energies?

In organic chemistry, we have the **functional group** concept, which says that a ketone in one molecule usually behaves a lot like a ketone in another molecule. When this transferability concept is applied to computational chemistry, one has **molecular mechanics** (MM).

MM is the cheapest of the computational methods and provides reasonable geometries and energies for ground states. However, it will not understand bonds that are stretched from equilibrium, like in transition states or other "exotic" behaviors like attractive dispersion forces unless they have been explicitly programed in. In this way, MM methods are a lot like the empirical NMR prediction methods used in ChemDraw.

Here is water again:



A molecular mechanic treatment of the energy of water would be something like:

$$E = k_{OH}\left(r_1 - r_{OH,eq}\right)^2 + k_{OH}\left(r_2 - r_{OH,eq}\right)^2 + k_{HOH}\left(\theta - \theta_{HOH,eq}\right)^2$$

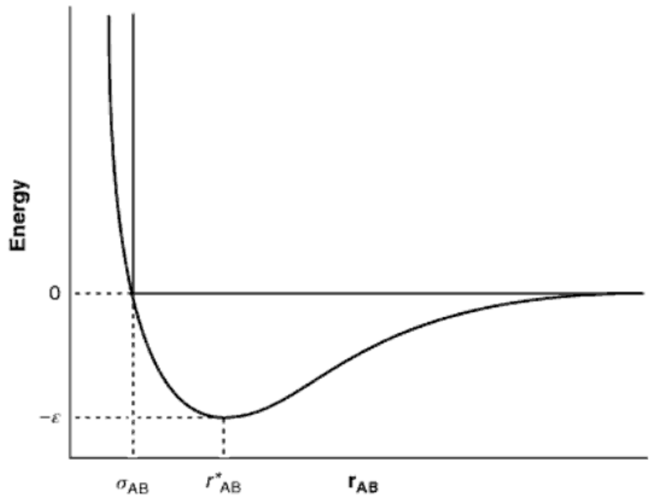
Thus, we are treating the bonds and bond angles as harmonic potentials: balls and springs. In more complicated molecules, dihedral angles will also appear with similar potentials. However, since torsional potentials are necessarily periodic, they will often contain trigonometric terms. Now, in this approximation, the energies are expanded as second-order terms. But as bonds get stretched farther from equilbrium, the validity of such potentials decreases. The answer is to use higher-order terms in the Taylor expansion. Inclusion of a cubic term would lead to an expression like:

$$E = \frac{1}{2}\left[k_{OH} + k_{AB}^{(3)}\left(r_1 - r_{OH,eq}\right)\right]\left(r_1 - r_{OH,eq}\right)^2$$

The term with the superscript (3) is the cubic force constant, or "anharmonic" constant. Unfortunately, such a function would diverge to negative infinity at large separations *r*. Therefore, in practice, one needs to include a quartic term. The standard organic force field MM3 uses such terms. One can also envision representing complex periodic behavior in torsional potentials by using higher order Fourier series.

What is missing here? So far, we have only considered the **bonded terms**. However, real molecules also have all kinds of non-covalent interactions that must be represented by additional **non-bonded terms**.

Consider the **hard sphere** and **Lennard-Jones** potentials (diagram taken from page 26 of Cramer):



In the hard-sphere potential, the two balls don't interact until some critical distance, the sum of the radii of the two balls. This is σ_{AB}. In the Lennard-Jones potential, there is a highly repulse $1/r^{12}$ term at close distances and a weakly attractive $1/r^6$ term at long distances.

Molecular Mechanics

As Cramer puts it, "one of the more profound manifestations of quantum mechanics is that [the hard sphere potential] does *not* accurately represent reality. Instead, because the 'motions' of electrons are correlated...the two atoms simultaneously develop electrical moments so as to be mutually attractive. The force associated with this interaction is referred to variously as 'dispersion,' the 'London' force, or the 'attractive van der Waals' force." In fact, the helium dimer is bound with one vibrational state with an equilibrium bond length of 55 A! The proper description of dispersive interactions is currently a major research topic in computational chemistry. In general, most higher level methods such as HF, MP2, or DFT have some problems describing things like benzene dimer, although some progress has been made (see Chem 106, Lecture 30).

Of course, there are other non-bonded terms other than steric repulsion and dispersion. In particular, one must take **electro-static interactions** into account. For an *inter*molecular complex between two molecules A and B, one can consider the classical energy of interaction to be the interaction of the multipole moments M of A (zeroth-order: charge; first-order: the x, y, and z components of the dipole moment; second-order: the nine quadrupole moments; etc.) and the electrical potentials from the moments of B. However, it's not clear how this would work for *intra*molecular interactions.

Instead, it is easier to **assign every atom a partial charge q** and calculate the electrostatic interaction energy U_{AB} as:

$$U_{AB} = \frac{q_A q_B}{\varepsilon_{AB} r_{AB}}$$

where ε is a constant that describes how well the charges interact through space. One can assign the charge simply based on the atom type, or on a more complex scheme involving the kinds of atoms near the atom in question. ε, too, can vary and is generally parameterized to neglect electrostatic interactions between atoms that are directly connected and attenuate interactions mediated by a torsional angle. Finally, **hydrogen-bonding** can also be described and is usually parametrized by a rapidly decaying "10-12" potential.

Obviously, one needs a lot of parameters to have a good force field. These parameters are generally fitted to reproduce experimental data and perform well in "normal" situations. For the standard atoms in the organic chemist's toolkit, there are good quality parameters for most functional groups.

Note that **there is no quantum mechanical justification for partitioning the energy into a sum of pairwise-additive contributions!** Nonetheless, MM is a viable way to model large systems or deal with smaller systems which have a lot of conformational flexibility because the computational resources it demands are tiny.

There are many flavors of MM **force fields**, which are sets of parameters designed for different systems. They are separated into "all atom" (aa) and "united atom" (ua) types. In the ua approximation, some groups like methyl groups are treated as a bigger "ball" to save computational time. A good summary appears in Chapter 2 of Cramer. Here is my summary of his summary:

CHARMM/m: Developed by Karplus and co-workers for biomolecules

MM2/MM3/...: Developed by Allinger and co-workers for organic molecules. MM2 has been superseded by MM3.

MMFF: Halgren and co-workers. For organics and biomolecules.

OPLS: Jorgensen and co-workers for organics and biomolecules with an emphasis on parameters that fit the experimental properties of liquids.

The choice of force field is a matter of taste and circumstance.

Molecular Mechanics

Q: How do we find the *global* minimum?

A frequency analysis can only tell us if a structure is a *local* minimum. You have a few options.

(1) Guess Yourself. This means drawing all the likely looking structures, minimizing them, and hoping you are thorough enough to find the global minimum. For complicated structures, particularly with more expensive quantum-mechanical methods, this is essentially the only option.

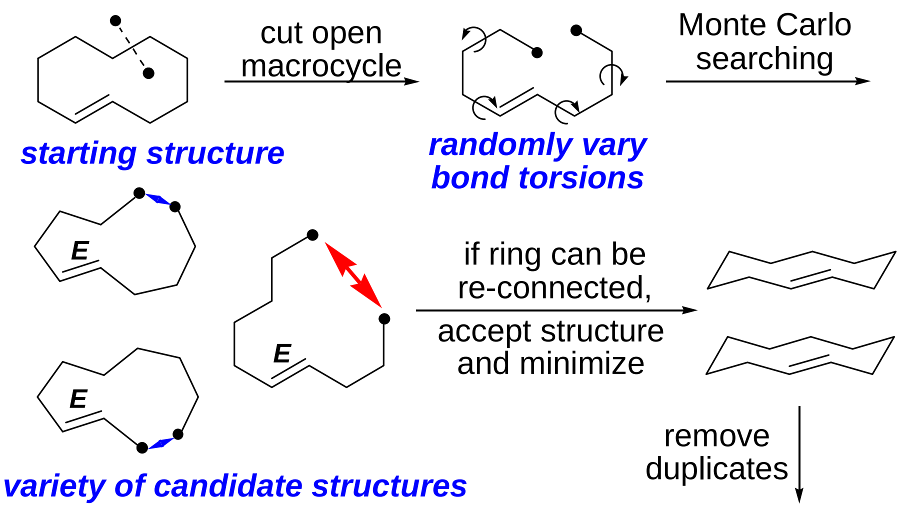
(2) Have the Computer Search Everything. If you are willing to accept a lower cost method (molecular mechanics) or perhaps wait a long time, then you can have the computer search the entire phase space of the system. It will try every combination of dihedral angles, bond angles, etc. Although this guarantees you will find the global minimum, it amounts to characterizing the entire PES, and is usually impractical.

(3) Have the Computer Guess. Here, you have the computer try and search part of the phase space at random. There are a number of methods for doing this, but a very common one is the **Monte Carlo** method.

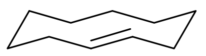
1) Start with some geometry.
2) Randomly change the bond angles, dihedrals, etc.
3) Calculate the new energy.
4) If the new energy is lower than the old energy, then "accept" this structure and use this structure in step 2.
5) If the new energy is higher, then *maybe* reject it. Reject structures that are much higher in energy more frequently than structures that are a bit higher in energy. More precisely, reject if $\exp[-(E_2-E_1)/RT]$ is less than some random number Z.

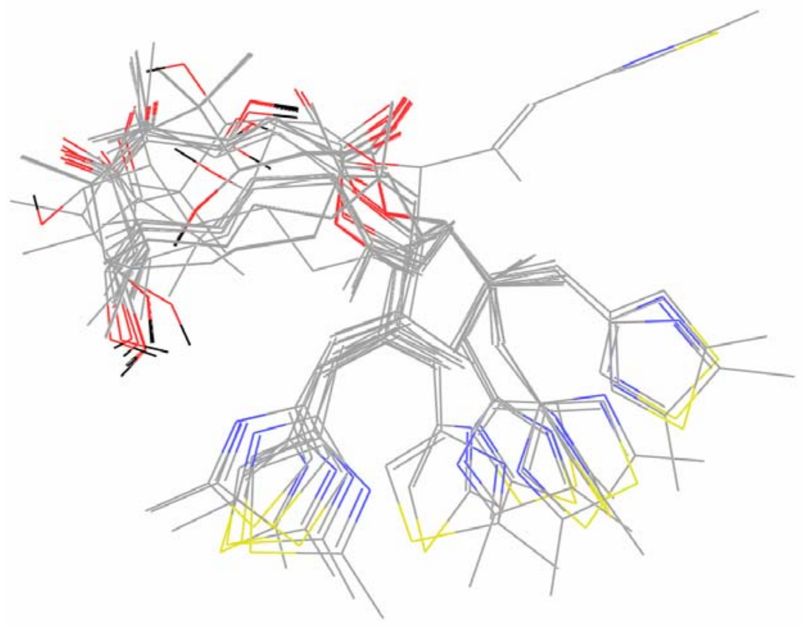
The fact that there is a chance of accepting higher energy structures means that the optimization algorithm can climb out of local minima.

Here is how you can visualize this process. My co-worker Joe and I are interested in macrocyclic conformations:


cut open macrocycle
Monte Carlo searching
starting structure
randomly vary bond torsions
E
E
E
if ring can be re-connected,
accept structure and minimize
remove duplicates
variety of candidate structures

Here are the lowest 20 structures for epothilone (as far as MM knows from this search):





Quantum Mechanical Methods

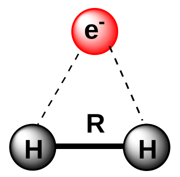
For many problems, the quality of molecular mechanics energies is simply insufficient. For better results, we turn to quantum mechanics. As you know, in non-relativistic quantum mechanics, absolutely everything is described by Schrodinger's equation, shown here in its one-dimensional, time-independent form:

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

$$\hat{H}\psi=\left(\hat{T}+\hat{V}\right)\psi=E\psi$$

The Hamiltonian operator H is composed of a kinetic energy part T and a potential part V. The potential energy operator V essential defines the nature of the question; the energy E is the answer.

Most of the time, we are interested in molecules. The simplest molecule is the hydrogen ion H_2^+:



$$E=E_{nuc}+E_{elec}$$

$$\hat{H}_{nuc}=\hat{H}_{trans}+\hat{H}_{rot}+\hat{H}_{vib}$$

If we make the Born-Oppenheimer approximation and assume that rotations and vibrations can be separated (the rigid rotor approximation), then we can come up with some equations for the translational, rotational, and vibrational energy of the nuclei. The translational energy depends on the volume of the cube V in which the molecule can move:

$$E_{trans}=\frac{h^2}{8mV^{2/3}}\left(n_x^2+n_y^2+n_z^2\right) \qquad n = 1, 2, 3, \ldots$$

h is Planck's constant while the *n* are quantum numbers. At absolute zero, we can assume the molecule is not moving, so translational energy makes *no contribution* to the total energy. The rotational energies are also quantized:

$$E_{rot}=\frac{h}{8\pi^2 cI}J\left(J+1\right) \qquad J = 0, 1, 2\ldots$$

where *I* is the moment of inertia and *J* is the rotational quantum number. The ground rotational state is exclusively occupied at absolute zero, so this, too, makes no contribution to the energy. However, the vibrational energy levels are:

$$E_{vib}=\left(\nu+\frac{1}{2}\right)h\nu$$

This means that the minimum vibrational energy, or **zero point energy (ZPE)** is nonzero ($E_{ZPE} = h\nu/2$). Thus, we find that at absolute zero, the energy of the molecule is:

$$E=E_{elec}+ZPE$$

Q: What is the electronic energy?

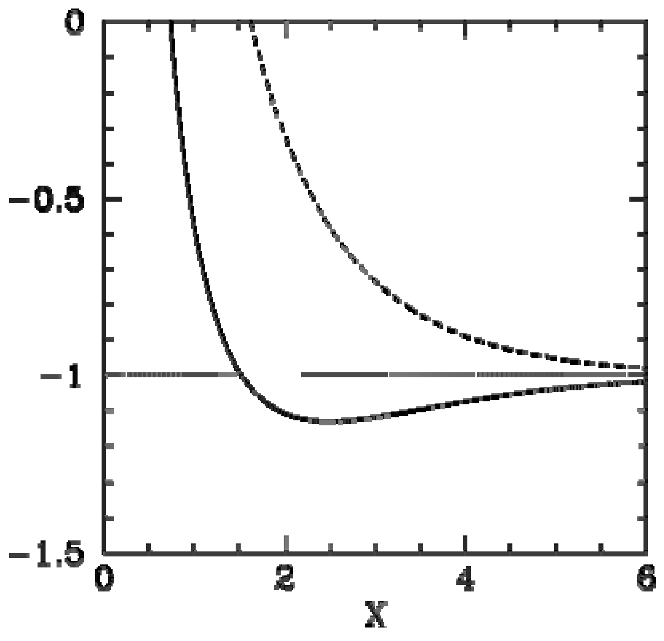
This is the quantity that is generally delivered to you by a quantum chemistry program like Gaussian. For the hydrogen ion, the Schrodinger equation is:

$$\left(-\frac{\hbar^2}{2m}\nabla^2-\frac{1}{r_a}-\frac{1}{r_b}-\frac{1}{R}\right)\psi(r;R)=E\psi(r;R)$$

r is the position of the electron (variable) while R is the internuclear separation (parameter). Because we are making the Born-Oppenheimer approximation, we solve the equation for a fixed value of R. This is what I mean when I say that we turn a geometry into an energy. (The position of the electrons is not fixed and is not part of the geometry.)

After quite a bit of math, one finds that the solution looks like...

Quantum Mechanical Methods



http://farside.ph.utexas.edu/teaching/qmech/lectures/node129.html

The solid line is the bound state; the dashed line is the unbound state.

Unfortunately, for anything more complicated, an analytical or exact solution is not possible. One has to resort to making guesses. Fortunately, our guesses are very good. Just how good are they? This is answered by the **Variational Theorem**.

The Variational Theorem

If a system has a ground state energy of E_1, and ϕ is a normalized, well-behaved function of the system's, then E_1 is no higher than the variational integral W:

$$\mathbf{W} = \int \phi^* H \phi d\tau \geq E_1$$

Proof: Levine, I. Quantum Chemistry (5th ed.) pg 208-209.

ϕ	a "guess" for the wavefunction; has to be "reasonable" (satisfy the boundary conditions of the problem being considered)
ϕ*	the complex conjugate of ϕ (ϕ need not be real)
H	the Hamiltonian operator, which you can think of as a widget which gives the energy of the guess ϕ
dτ	a notation that means "integrate over all space"
W	the ***variational energy***

(Note that ϕ has nothing to do with the spherical coordinate.)

So, for any guess ϕ, the ground state energy of the system E_1 is no higher than W. This suggests we should try to vary ϕ, and seek a form of it that minimizes W:

Translation:

$$\begin{tikzcd}
\boxed{\text{Exact solution unknown.}} \arrow[r] & \boxed{\text{Make a reasonable guess, } \phi.} \arrow[r] & {} \\
\boxed{\text{Compute the integral W.}} \arrow[r] & \boxed{\textbf{The real ground state energy is below the value of the integral.}} & \\
{} \arrow[r] & \boxed{\text{Refine } \phi \text{ and try again.}} &
\end{tikzcd}$$

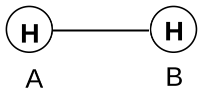
Q: How do we systematically improve the guess ϕ?

Lecture 9: Introduction to Computational Chemistry

The Linear Combination of Atomic Orbitals (LCAO)

Instead of varying the *function* ϕ, make ϕ a linear combination of some other functions, and vary the *coefficients* to minimize the variational integral.

Let's try this for the hydrogen molecule. A reasonable guess for ϕ might be 1s orbitals of the hydrogen *atom*. Thus, we form a linear combination:



$\phi = c_1\phi_1 + c_2\phi_2$
ϕ_1 = 1s orbital on hydrogen A
ϕ_2 = 1s orbital on hydrogen B

Thus, we seek values of c_1 and c_2 that minimize W. More precisely:

$$\frac{\partial W}{\partial c_1} = 0 \qquad \frac{\partial W}{\partial c_2} = 0$$

This is advantageous because it's hard to vary a *function*, but it's easy to vary the coefficients c_1 and c_2 in a linear combination.

The Secular Equations

As it turns out, if you vary c_1 and c_2 to minimize W, then c_1 and c_2 have to satisfy the *secular equations*:

$$c_1(H_{11} - S_{11}W) + c_2(H_{12} - S_{12}W) = 0$$
$$c_1(H_{21} - S_{21}W) + c_2(H_{22} - S_{22}W) = 0$$

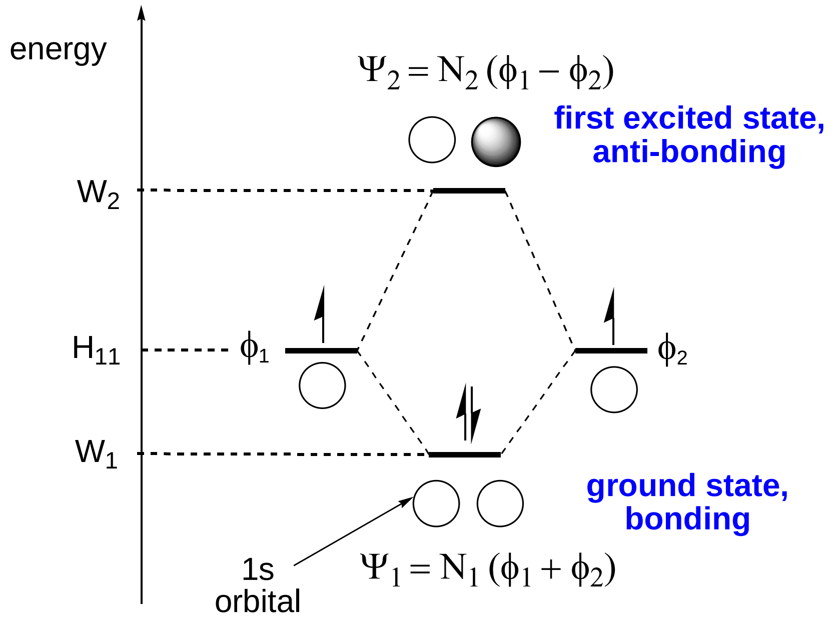
where $H_{12} = \langle \phi_1 | H | \phi_2 \rangle = \int \phi_1^* H \phi_2 \, d\tau$ Hamiltonian matrix element/ resonance integral

$S_{12} = \langle \phi_1 | \phi_2 \rangle = \int \phi_1^* \phi_2 \, d\tau$ overlap integral

Note: $S_{ij} = S_{ji}$ (overlap of i with j = overlap of j with i)

$H_{ij} = H_{ji}^*$ (the Hamiltonian is Hermitian)

For details, please see Levine, 220-223. If we look at the solutions, we find something like this:


energy
$\Psi_2 = N_2(\phi_1 - \phi_2)$
first excited state, anti-bonding
W_2
H_{11}
ϕ_1
ϕ_2
W_1
ground state, bonding
1s orbital
$\Psi_1 = N_1(\phi_1 + \phi_2)$

(The factors *N* are to *normalize* the wavefunctions. This amounts to saying that the electron has a 100% chance of being *somewhere*.)

Basis Sets

Q: What form do these orbitals take?

Clearly, we need some functions if we want to form a linear combination and variationally minimize the coefficients. A **basis set** is a collection of such functions. The shape of these functions has been optimized for all the different elements in the periodic table for various considerations: computational efficiency, quality of results, etc.

Basis Sets

The "Gaussian-type orbitals" were popularized by Pople and Boys and are particularly numerically efficient:

$$g_{ijk} = \underset{\text{normalization constant}}{N}\underbrace{x^i y^j z^k}_{\text{indicates angular momentum}}\underbrace{\exp\left(-\alpha r^2\right)}_{\text{a Gaussian}}$$

i + j + k = 0: s-type Gaussian
i + j + k = 1: p-type Gaussian
i + j + k = 2: d-type Gaussian

Each g_{ijk} is a *primitive Gaussian*. Special linear combinations of primitives are selected to look like hydrogenlike orbitals. These are called *contracted Gaussian-type orbitals* (CGTO). These orbitals are used because integrating Gaussians is very easy computationally.

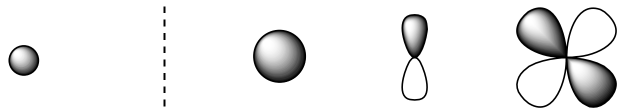
The most common basis set of this type is called 6-31G*. For a hydrogen atom:

hydrogens: 2x1s

Carbons, however, have a lot more electrons, and so need many more basis orbitals:

CORE ***VALENCE***

carbons:



2x1s | 2x2s 6x2p 6x3d

A basis set that is divided into core and valence orbitals is called "split valence." The emphasis on valence orbitals is a reflection of the fact that more mathematical flexibility is required to describe the behavior of the bonding electrons, which differs between molecules, than that of the core electrons, which is very similar between molecules.

What does the 6-31G* nomenclature mean? It is equivalent to the 6-31g(d) designation:

$$\text{6-}\underbrace{\text{31}}\text{g(d)}$$

the core is described by one CGTO composed of six primitive GTOs $\rightarrow$ 6

d-type orbitals are added to heavy atoms

this is a split-valence basis set

these are Gaussian-type orbitals (GTOs)

two CGTOs are used to describe every atomic orbital: one with three CGTOs and one with one CGTO

This generalizes to bigger basis sets:

$$\text{6-31g(3d2f,2p)}$$

heavy atoms have three sets of d-orbitals and two sets of f-orbitals added

hydrogen atoms have two p-orbitals added to them

Since *J*-couplings in NMR are transmitted through protons, the addition of p-orbitals to hydrogens is important for the quality of the results of NMR predictions.

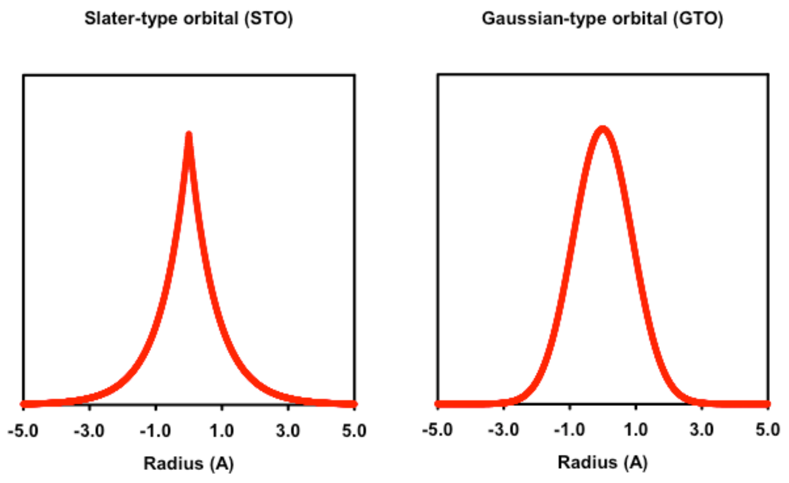
The orbitals of anions, excited electornic states, and loosely associated supramolecular complexes require **diffuse functions**, which are designated by "+" as in 6-31+g(d,p). These are an additional set of orbitals for the heavy atoms with small orbital exponents.

Q: Where do orbital exponents come from?

Basis Sets

Let me remind you that we want to construct a variational guess for the wavefunction that has coefficients that can be optimized to minimize its variational energy. The number of two-electron integrals increases as N^4 where N is the number of basis functions. Therefore, we want to minimize not only the number of basis functions but the computational cost involved in evaluating any particular integral. Ideal orbitals should also have a lot of density where there is electron density in real life.

If you recall the hydrogen molecule example, we guessed that the orbitals in the hydrogen *molecule* (which we don't know) were similar to the orbitals in the hydrogen *atom* (which we do know). The generalization of this is the **Slater-type orbitals (STOs)**:

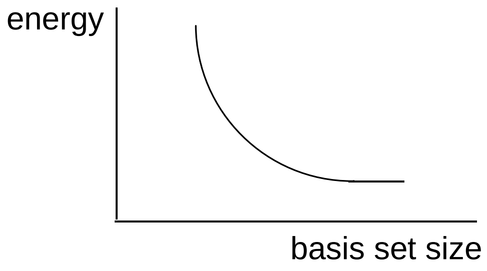


The STOs are advantageous from a chemical standpoint because "real" orbitals have cusps (whereas GTO primitives do not). However, integrals of STOs are very annoying from a computational standpoint, so they are not generally used.

GTOs are also "bad" chemically because they have no radial nodes. The solution to both these problems is to combine a number of GTO primitives into a CGTO.

However, with split-valence basis sets, one can fit one set of the basis functions to the STOs, but it is unclear what to fit the remaining basis functions to. By "fit," I mean adjust the orbital exponents of the CGTOs and the linear combination coefficients of the CGTO expansion to best fit the STO. The answer is to use the variational principle. Pople and co-workers came up with a test set of molecules, and optimized the parameters to give the best results for a given method.

There are many other basis sets as well. The most popular competitor are the **Dunning correlation-consistent** basis sets. One can imagine that as the basis set gets larger, the mathematical flexibility (the "span") gets larger and larger until it becomes infinitely flexible. Thus, the variational energy should drop as the basis set decreases in size:

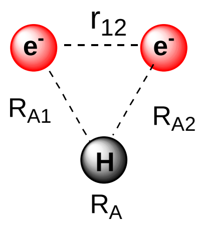


For reasons that will be obvius soon, the infinite basis set is called the "Hartree-Fock (HF) limit." The question is: how smoothly will the energy go down as the basis set size goes up? As it turns out, the quality of a calculation depends a lot on how well **electron correlation** is taken into account. The Dunning basis sets are variationally optimized to add constant increments of electron correlation energy as the basis set size increases. The nomenclature is "cc-pV*N*Z," for correlation consistent polarized valence (double (D), triplet (T), ...) zeta. Diffuse functions are designated by the "aug" prefix, as in: aug-cc-pVDZ for a double-zeta basis set.

A new trend is the use of **density fitting** or **resolution of the identity (RI)** methods. These expand the basis set with some auxiliary basis set to speed up calculations.

The Orbital Approximation and the Pauli Principle

A helium atom has two protons and two electrons:



The Hamiltonian now includes an electron-electron repulsion term:

$$\left(\frac{-\nabla_{r_1}^2}{2}-\frac{\nabla_{r_2}^2}{2}-\frac{2}{R_{A1}}-\frac{2}{R_{A2}}+\boxed{\frac{1}{r_{12}}}\right)\Psi(r_1,r_2)=E\Psi(r_1,r_2)$$

Nobody knows how to deal with this exactly, so one makes the **orbital approximation** that the overall wavefunction is the product of a number of one-electron wavefunctions, or orbitals. The composite wavefunction is called a **Hartree product**:

$$\Psi(r_1,r_2)=\phi_1(r_1)\phi_2(r_2)$$

For a lithium atom, you might think that you can just write a Hartree product like this:

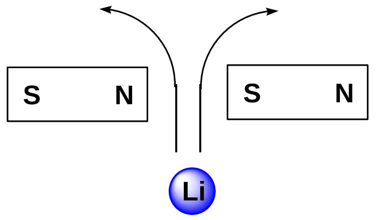
$$\Psi(r_1,r_2,r_3)=\phi_1(r_1)\phi_2(r_2)\phi_3(r_3)$$

However, you will find that if you use hydrogen-like orbitals for the ϕ, you will find that the answer violates the variational principle! That is, the variational energy of the wavefunction will actually be *lower* than the experimental wavefunction. The problem is that this wavefunction violates the **Pauli exclusion Principle**, which not only prevents two electrons from having the same quantum numbers, but also requires that **the wavefunction be antisymmetric with respect to electron interchange**. If I interchange the indices in the trial function for lithium above, I do *not* get the negative of the trial function:

$$\phi_1(r_1)\phi_2(r_2)\phi_3(r_3)\neq-\phi_1(r_2)\phi_2(r_1)\phi_3(r_3)$$

All I have done here is to make ϕ_1 a function of the coordinates of electron 2 instead of electron 1 and the reverse for ϕ_2. So what *would* satisfy the Pauli principle? The **Slater Determinant**.

As you know, if you shoot some lithium atoms through a magnetic field, some of them will curve to the left and some will curve to the right (whereas helium atoms won't curve):



This called **spin** because the lithium atoms behave like clasically spinning charged spheres (even though they're not really spheres or spinning). For every spatial wavefunction, we can put an α or β electron in it to make a **spin-orbital**. For example, putting an α electron in orbital 1 gives:

$$\psi_1(1)=\phi_1(1)\alpha(1)$$

The Slater determinant gives us a way to construct anti-symmetric wavefunctions. For helium, we need to construct one for two electrons:

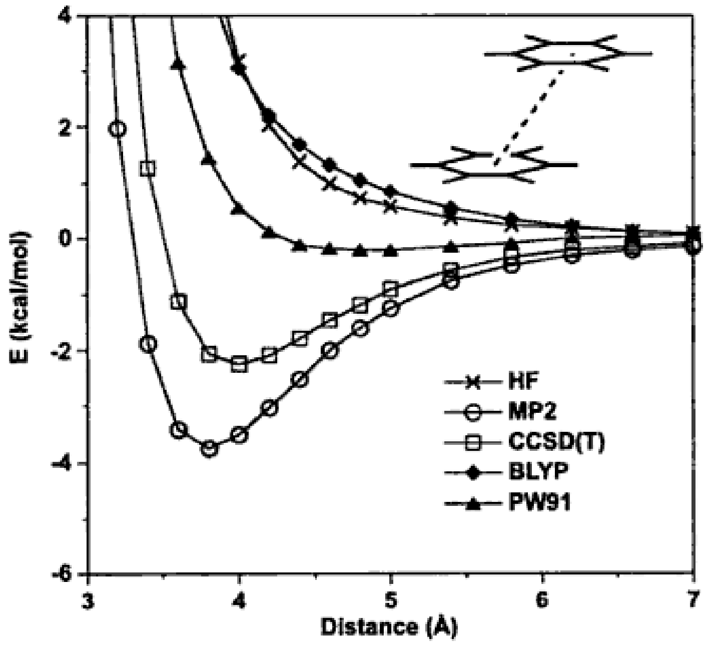
$$\Psi_{He}=\frac{1}{\sqrt{2}}\begin{vmatrix}\phi_1(1)\alpha(1) & \phi_2(1)\alpha(1)\\ \phi_1(2)\alpha(2) & \phi_2(1)\alpha(1)\end{vmatrix}$$

$$=\frac{1}{\sqrt{2}}\left[\phi_1(1)\alpha(1)\phi_2(1)\alpha(1)-\phi_2(1)\alpha(1)\phi_1(2)\alpha(2)\right]$$

Now, interchanging the electrons *does* result in an anti-symmetric wavefunction. Note that every electron appears in every spin orbital somewhere; the electrons are identical.

Electron Correlation

I already mentioned that electron correlation is essential to non-covalent interactions like π-stacking. The crosses in the graph below show the potential energy curve for benzene dimer:



Notice that HF thinks that the benzene dimer is not bound at all! Clearly, this is wrong. Interestingly, some of the other methods there *do* account for some electron correlation, but clearly don't do it quite correctly. For example, BLYP is a common kind of density functional (to be discussed shortly) and accounts for some electron correlation, but still doesn't understand the stacking interaction.

The neglect of electron correlation has other serious consequences:

- Heats of formation are very inaccurate. For a bunch of small molecules calculated at HF/aug-cc-pVQZ, a very large mean unsigned error of 62 kcal/mol was found!

- Processes which break bonds cannot be described well. For example, pericylic reactions can involve a number of stretched bonds with some diradical character. HF cannot predict the barrier heights of these correctly.

- Other measures of energy differences like isomerization energy (e.g., CO + HO radical vs. H radical + CO2) don't come out right either.

In general, **molecular geometries are more accurate than energies**, regardless of the method. HF is not exception. Note that geometry with the HF method (or any other method) requires **analytic gradients** to be fast enough for practical use. This means we must know the derivatives of the density matrix elements with respect to the coordinates of the system, which is not really obvious. Fortunately, Pulay discovered an efficient way to do this in 1969, and fast HF methods are now widely available.

The important point is that despite all this <u>computations accurately describe reality:</u>

Accuracy of HF Results (David Sherill)

bond lengths: to within 0.02 A
bond angles: to within 2 degrees
vibrational frequencies: to within 10%
dipole moments: to within 0.3 D
bond dissociation energies: 25-40 kcal/mol off

Fancier methods are better. For coupled-cluster:

bond lengths: to within 0.004 A
bond angles: to within 0.03 degrees
vibrational frequencies: to within 2%
dipole moments: to within 0.05 D
bond dissociation energies: 1-2 kcal/mol off

Post-HF Methods

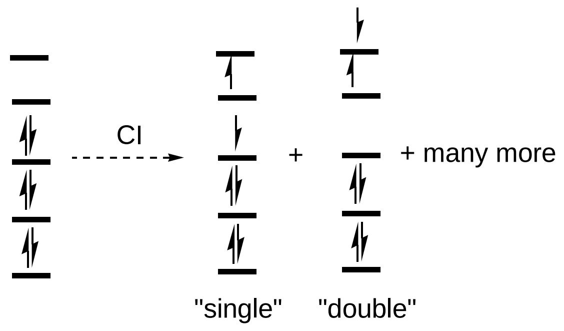
Q: How can we take electron correlation into account?

There are many methods available for this, but in general, the more accurate the method, the more costly it is. The "scaling" of the method tells you how much more time you need as you increase the number of basis functions N. One must also consider a "pre-factor" which tells you how much "up front" cost is required.

Hartree-Fock ~ N^4
DFT (density functional theory) ~ N^3 (but larger pre-factor than HF)
MP2 (Moller-Plesset perturbation theory) ~ N^5
CISD (configuration interaction singles and doubles) ~ N^6
CCSD(T) (coupled cluster singles and doubles with perturbative estimates of triples) ~ N^7

The last one, CCSD(T), is the "gold standard" computational method, but is staggeringly expensive in CPU and memory requirements. Just calculating the benzene dimer with CCSD(T) with it was a research-grade project worthy of a JACS article just five years ago!

The idea behind the MP2, CISD, and CCSD(T) methods is relatively simple to understand. The Hartree-Fock forms its guess from a single Slater determinant. If we want to improve the guess, we can include determinants involving "excited states":



Hartree-Fock limit: HF with an infinite basis set

If we variationally minimize the Slater determinants resulting from all these excitations *and* have an infinite basis set, then we have reached an exact numerical solution for the non-relativistic Schrdoinger equation--the **configuration interaction limit**.

Of course, all these excitations are very costly. In CISD, we just make single and double excitations. In CCSD, we are trying to do the same thing, but in a different way. We apply an operator exp[T] to the HF wavefunction:

$$\Psi = e^{\hat{T}}\Psi_{HF}$$
$$\hat{T} = \hat{T}_1 + \hat{T}_2 + \hat{T}_3 + \cdots$$

where the indices indicate the level of excitation the operator represent. This has certain advantages, among them **size consistency**. If you calculate the energy of A separately and B separately, the sum should be about the same as calculating the energy of A and B together, but separated by quite a large distance. If that's true, the method is called "size consistent."

In MP2, we apply perturbation theory to correct the energy of the Hartree-Fock wavefunction. Thus, the orbitals are the same, but the energies are different. (This means the optimized geometries are different, too.) As organic chemists, we work with molecules containing dozens of heavy atoms, if not hundreds. As such, we can only afford HF and MP2, and even then MP2 is problematic for larger atoms.

A huge breakthrough in quantum chemistry arrive in the 1990s, when **density functional theory** (DFT) became widely available. Although it has a larger pre-factor than HF, its scaling with *N* is considerably less than the other post-HF methods, making it an attractive, practical method.

Density Functional Theory (DFT)

Q: How can we take electron correlation into account in a *practical* way?

(This discussion is taken from Cramer, Chapter 8.)

Ab initio methods like Hartree-Fock or MP2 are ways to arrive at the wavefunction, which is itself a proxy for electron density (or other observables via the action of operators). In DFT, we try to compute the electron density ρ *directly*. The total number of electrons *N* is conserved over all space:

$$N = \int \rho(r)\,dr$$

Is the electron density enough information to reconstruct the information we want--the energies and wave functions? Yes! The Hamiltonian requires the positions and atomic numbers of the nuclei and the total number of electrons. Clearly, having the density can give us the total number of electrons.

What about the nuclei? They are point positive charges, so the electron density reaches local maxima at the nuclear positions. So we can reconstruct the positions of the nuclei from the maxima in electron density. Their atomic numbers can be extracted from:

$$\left.\frac{\partial\bar{\rho}(r_A)}{\partial r_A}\right|_{r_A=0} = -2Z_A\rho(r_A)$$

where r_A is the position of an electron density maximum, ρ bar is the spherically averaged density, and *Z* is the atomic number of nucleus A.

So we have shown that *if* we had the electron density, we would (at least in principle) be able to write down the Schrodinger equation, solve it, and get the energies and wavefunctions. But how do we actually go about doing that? Specifically, **how does one turn the density into energy**?

In **Thomas-Fermi DFT** (1927), one tries to calculate the energy in a classical way. The attraction between the density and the nuclei is:

$$V_{ne}\left[\rho(r)\right] = \sum_k^{nuclei} \int \frac{Z_k}{\left|r - r_k\right|}\rho(r)\,dr$$

Electron-electron repulsions are given by:

$$V_{ee} = \frac{1}{2}\iint \frac{\rho(r_1)\rho(r_2)}{\left|r_1 - r_2\right|}dr_1 dr_2$$

What about the potential energy? In this crude treatment, one envisions an infinite number of electrons moving through an infinite volume of space containing a uniformly distributed positive charge. This is the "**uniform electron gas**." Thomas and Fermi showed that:

$$T_{ueg} = \frac{3}{10}\left(3\pi^2\right)^{2/3}\int \rho^{5/3}(r)\,dr$$

No two electrons can be in the same place, so one introduces a "hole function" to account for this:

$$\left\langle \Psi \left| \sum_{i<j}^{electrons} \frac{1}{r_{ij}} \right| \Psi \right\rangle = \frac{1}{2}\iint \frac{\rho(r_1)\rho(r_2)}{\left|r_1 - r_2\right|}dr_1 dr_2 + \frac{1}{2}\iint \frac{\rho(r_1)h(r_1;r_2)}{\left|r_1 - r_2\right|}dr_1 dr_2$$

LHS: exact quantum-mechanical interelectronic repulsion
RHS, first term: classical interelectronic repulsion
RHS, second term: corrects the density with a hole function *h*

The hole function for a multielectron system is not obvious and must often be approximated. Not having correct hole functions has serious consequences--electrons can interact with themselves! In HF, there is *no* **self-interaction error**, but all DFT methods suffer from this to some extent. This is why many DFT methods are combined with some degree of HF exchange energy.

Density Functional Theory (DFT)

The fact that electrons are indistinguishable and must be be antisymmetric with respect to interchange leads to a purely quantum mechanical effect known as **exchange** and has no classical analog. The presence of exchange keeps electrons farther apart than they would be predicted to be classically and is responsible for steric repulsion.

As it turns out, the contribution of exchange to the classical interaction energy is much larger (1-2 orders of magnitude) than the electron correlation energy. Slater proposed that the "exchange hole" can be approximated by a sphere of constant potential with a radius depending on the magnitude of the density at that position (Cramer, pg 252):

$$E_x = -\frac{9}{8}\left(\frac{3}{\pi}\right)^{1/3} \int \rho^{4/3}(r)\,dr$$

This is "**Slater exchange**." Unfortunately, Thomas-Dirac DFT is entirely *in*adequate to describe anything of chemical interest. In fact, it erroneously predicts that all molecules are unbound states. Additionally, there was no clear analog to the variational principle, so the theory made little impact on chemistry.

In 1964, Hohenberg and Kohn proved two theorems that proved crucial to establishing DFT as a useful tool in quantum chemistry. This led to the awarding of the Nobel Prize to Kohn in 1998 (along with Pople for developing other computational methods).

Hohenberg-Kohn Existence Theorem

There is a one-to-one correspondence between the ground state wavefunction and the ground state electron density.

Hohenberg-Kohn Variational Theorem

The electron density that minimizes the total energy is the exact ground state density.

So far, it looks like we have to guess a density, convert it into a candidate Hamiltonian and wavefunction, evaluate the energy by solving the Schrodinger equation, and keep guessing until the variational energy is minimized. But this is obviously rather unsatisfactory! How do we go about finding better densities? How do we convert these densities into energies without solving the Schrodinger equation? Remember, the whole point of using the electron density is to *avoid* solving the Schrodinger equation in the first place!

The answer is the **Kohn-Sham self-consistent field** method, which is analogous to the SCF method used in Hartree-Fock. The idea is to consider a fictitious system of *non*-interacting electrons that have the same electron density as the real system where the electrons *do* interact. Then, we divide the energy *functional* into different components:

$$E[\rho(r)] = T_{ni}[\rho(r)] + V_{ne}[\rho(r)] + V_{ee}[\rho(r)] + \Delta T[\rho(r)] + \Delta V_{ee}[\rho(r)]$$

where T_{ni} is the kinetic energy of the non-interacting electrons, V_{ne} is the classical nuclear-electron attraction, V_{ee} is the classical electron-electron repulsion, ΔT is the quantum-mechanical correction to the kinetic energy, and ΔV is the quantum-mechanical correction to the electron-electron repulsion. (A *functional* is a function that takes functions, rather than numbers, as arguments.)

What is the nature of ΔT and ΔV? This is called the "**exchange correlation energy (E_{xc})**" and accounts for self-interaction energy and the difference in kinetic energy between the fictitious and real systems, in addition to exchange and correlation. **Nobody knows what the exact functional is.** (If you could figure this out, you'd win a Nobel Prize.) In DFT methods, one makes a guess for what the functional is. That's where all the different flavors of DFT come from (B3LYP, PW91, M05-2X, etc.)--they are different approximations to E_{xc}. Note that while *exact* DFT is variationally correct, *approximate* DFT is *not* variationally correct. However, both are size-consistent.

Density Functional Theory (DFT)

So **Hartree-Fock is an exact solution to an approximate theory, while DFT is an approximate solution to an exact theory.** It just so turns out that DFT gives much better results than HF. Purists will complain that since approximate DFT is not variationally correct, there is no systematic way to improve DFT results, and therefore, wavefunction methods are preferable. They are correct about the first part, but wrong about the second part--it is now well established that DFT gives chemically meaningful information about a wide variety of interesting systems.

Q: What are the various flavors of DFT?

There are a lot of choices here, so I will just summarize the key points.

The **local density approximation (LDA)** assumes that the exchange-correlation functional only depends on the electron density at the point where the functional is being evaluated. Systems involving unpaired electrons use the **local spin density approximation (LSDA)**. The functionals developed by Vosko, Wilk, and Nusair (VWN) are an example of this. The details are quite abtruse--it's not clear where all the terms in the expression come from, physically speaking. They were obtained by empirically fitting parameters to some known results. In this sense, DFT methods can be considered truly semi-empirical.

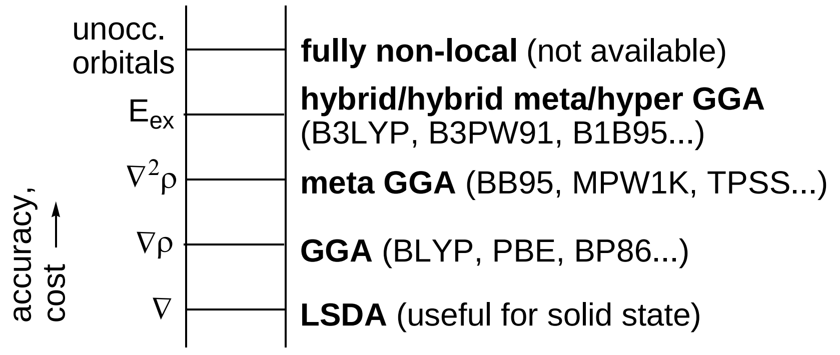
Of course, in a real system, the electron density is *not* uniform. The first-order correction to this is to account for not only the electron density at a particular point, but also the rate at which the density is changing at that point--the gradient. This leads to the **generalized gradient approximation (GGA).** One popular kind of GGA DFT is called PW91.

Going futher, we can have **meta-GGA** methods which include higher-order terms like the Laplacian ∇^2.

In **adiabatic correction methods**, one computes some of the exchange correction exactly with HF, and then includes some terms from the other expressions. For example, in the popular B3LYP functional, one uses the three-parameter scheme:

$$E_{XC}^{B3LYP} = (1-a)E_X^{LSD} + aE_{XC}^{HF} + bE_X^{B88} + cE_C^{LYP} + (1-c)E_C^{LSD}$$

This is a "**hyper GGA**" functional, since it includes a GGA part and a Hartree-Fock exchange part. This gives rise to a ladder of functionals:


unocc. orbitals
E_{ex}
$\nabla^2\rho$
$\nabla\rho$
∇
accuracy, cost →
fully non-local (not available)
hybrid/hybrid meta/hyper GGA (B3LYP, B3PW91, B1B95...)
meta GGA (BB95, MPW1K, TPSS...)
GGA (BLYP, PBE, BP86...)
LSDA (useful for solid state)

There are numerous benchmarks which show that the accuracy goes up as you go up the ladder (with a lot of nuances). For this course, we will just use B3LYP, which is known to work well for many systems. However, it does *not* describe medium- to long-range electron correlation very well like that involved in π-stacking. Functionals like DFT-D (Grimme) and M06-2X (Truhlar) have been specifically parametrized to work well for these cases.

DFT scales as N^3, whereas HF scales as N^4, where *N is* the numebr of basis functions, so there is a clear advantage for larger systems. However, note that the pre-factor for DFT is larger, particularly if HF exchange is needed.

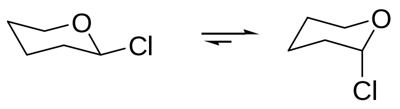
Further Information: Nick Mosey has prepared a detailed set of course notes on DFT which are available on the web here: www.chem.queensu.ca/people/faculty/Mosey/chem938.htm.

Jumping the Gun

Clearly, computational chemistry is immensely complicated, and you are not going to learn it all from me in an hour. I don't really even understand a lot of it. But don't feel bad--there is absolutely no need to understand everything before getting started, just as you don't need to know exactly how a car's engine works before driving. Professor Jacobsen reminds me on occasion that as organic chemists, our talent is for making very simple models of very complicated systems, and getting out some results that make a lot of sense. My philosophy is that knowing a little about how things work will enable us to choose computational methods appropriately, design computational studies that answer real chemical questions, and troubleshoot calculations when they invariably crash. However, it is easy to get bogged down in the details of, say, how to compute the Hartree-Fock density matrix. Although learning these details takes a copious amount of time, they don't really help us get any of the answers we need in organic chemists. So I suggest we all try to get a general sense for all of the different methods, and open up dialogs with physical chemists to collaborate when we need to do something a bit more complicated.

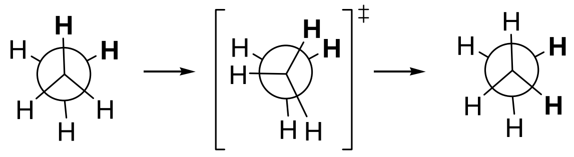
In this tutorial, which I strongly suggest you go through at home, we will do two things:

(1) Compute the energy difference between axial and equatorial 2-chlorotetrahydropyran.



(As you know, the stability difference is anomeric in nature.)

(2) Compute the rotational barrier in ethane.



To get started, you will need several things:

(a) GaussView 5.0 installed on your computer (freely available for PC and Mac from the chemistry library)
(b) a secure FTP client (Mac: Fetch; PC: WinSCP)
(c) a secure telnet client that supports keychain authorization (PC: PuTTY; Mac: terminal)
(d) an account on the Odyssey Computing Cluster (http://rc.fas.harvard.edu; rchelp@fas.harvard.edu).

Computations themselves do not cost any money, but buying computers to which you have priority access on the cluster does. They're not cheap (starting price: \$10 000). But the cluster has computers everyone can use for free (but you have to wait).

The Odyssey Cluster

old days: buy one really fast, expensive computer that can do one calculation at a time

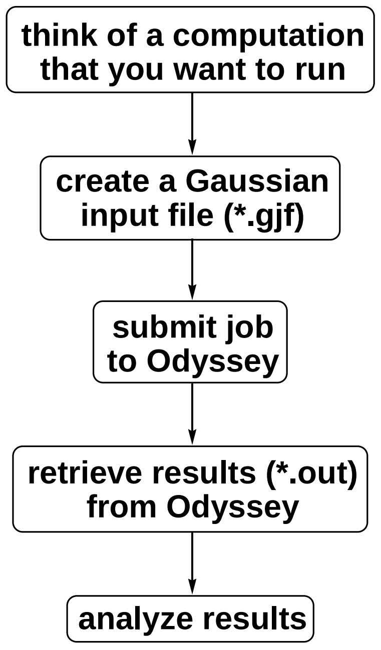
these days: buy a lot of decent computers, and then use a lot of them at once

The Odyssey Cluster lets us do the latter. The magic of computer science means that computations can be done in **parallel**, which means that every calculation is broken up into many pieces, and different computers work on different parts of the calculation simultaneously. This results in drastic speedups in computational efficiency, without the enormous cost of supercomputers.

Anyone with an FAS affiliation can have an account on Odyssey and run calculations. If you belong to a particular research group, you may have access to certain computer resources that other people don't. For example, the Evans group has 8 "nodes." Every node contains 8 CPUs, or cores. At the moment, "cross-node parallelism" is theoretically possible, but unnecessary for our purposes. Thus, we are generally limited to 8 cores/job. (The Jacobsen group has recently purchased some 12-core nodes.)

Computational Workflow

No matter what the project you're working on is, everyone follows the same workflow. We'll consider each step in turn:



think of a computation that you want to run
create a Gaussian input file (*.gjf)
submit job to Odyssey
retrieve results (*.out) from Odyssey
analyze results

This is the hardest part, but unfortunately, there's no room to talk much about this.

You have to tell Gaussian, a quantum chemistry software package what you want it to do.

Computing resources are precious, and we all share them using the LSF queueing system.

Output comes back in a text file, but you can use GaussView to look at the results graphically.

This is where you have to use your chemical intuition.

Idea: Not everyone is using their computers all the time, so let's share the unused resources in a fair way.

Implementation: Different clusters use different programs, but here at Harvard, we use the LSF program. Here is some terminology:

job: a computation you want to run on the cluster
core: an individual CPU
node: a bunch of CPUs put together into one computer

queue: a group of jobs waiting to run on the cluster; different queues target different nodes with differing priorities depending on the user who submitted the jobs

Q: How do I access the cluster?

The first step is to login to the cluster via SSH (odyssey.fas. harvard.edu). The first time you login, you may be asked whether to accept a "fingerprint;" say yes. For security reasons, you are asked to type in your login name, password, and RSA security token passcode. The idea is that nobody can login as you, unless they have *both* your password and passcode. The passcode changes with time in a pre-defined way; you read it off a small plastic key fob ("hard token") or a readout on your screen ("soft token"). Most accounts are now setup with soft tokens.

When I login, it looks like this:

```
login as: ekwan
Using keyboard-interactive authentication.
Password:
Using keyboard-interactive authentication.
Enter PASSCODE:
Last login: Fri Feb  4 09:35:40 2011 from c-66-30-9-
8.hsd1.ma.comcast.net

        Java Runtime Environment 1.6 module
        ****************************************************

          This module sets up the following environment
          variables for Java Runtime Environment 1.6:
              PATH /n/sw/jdk1.6.0_12/bin

        ****************************************************

Loading module devel/intel-11.1.046.
Loading module hpc/gaussian-09_ekwan.
[ekwan@iliadaccess01 ~]$
```

When I typed my password and passcode, no text was mirrored to the screen (so it's harder to steal my password if you happen to be looking over my shoulder). The other information tells me when I logged in last and from where, as well as what *modules* have been loaded. The cluster is not just for Gaussian; many other scientists are using other programs. Loading modules prepares the cluster to use Gaussian by setting certain environment variables, aliases, etc.

Now, we're ready to take a look around.

The Queue System

Q: What computing resources are available?

```
[ekwan@iliadaccess01 ~]$ bqueues

QUEUE_NAME       PRIO STATUS        NJOBS  PEND   RUN  SUSP
dae              200  Open:Active     64      0    64     0
enj              200  Open:Active     84      0    84     0
karplus          196  Open:Active     24      0    24     0
lsdiv             55  Open:Active    143      0   143     0
betley            45  Open:Active      4      0     4     0
short_parallel    30  Open:Active      2      0     2     0
normal_parallel   25  Open:Active   2561   2284   265     0
long_parallel     20  Open:Active   2969   1304  1665     0
long_serial       15  Open:Active    150      0   150     0
short_serial      10  Open:Active   8472   7907   532    33
normal_serial      5  Open:Active   3296   1906  1150   240
unrestricted_pa    3  Open:Active    159      0   159     0
unrestricted_se    1  Open:Active    161     97    64     0
```

(This is a truncated listing.) *number of cores pending, running, or suspended*

The LSF system tries to allocate the available resources fairly. Jobs are submitted to queues, each of which targets certain nodes. However, unlike lines in real life, these are *not* first-in, first-out. Priority within a queue is assigned based on how much computing power you have been using within the last few days. For example, suppose my co-worker Joe submits 100 eight-core jobs to dae. dae only has 8 eight-core nodes, so supposing they're all free at the moment, all 8 start running immediately. Then, suppose I decide to submit a job, and I haven't been doing any computations lately. My job will run before any of Joe's remaining 92 jobs, but not before one of the 8 jobs currently on dae finish.

Not everyone has access to all the queues. For example, only people in the Evans group have access to the dae queue (since Professor Evans paid for the computers). However, you can still run jobs on our nodes if you submit your jobs to the common queues, which end with "serial." These target a mixture of FAS-owned computers and computers owned by individual research groups. However, these jobs are subject to "suspension," which means that if you are running a job on a dae node, and I submit a job, then your job will be be paused immediately. My job will run, and then your job will resume. The "parallel" queues are also open access, but are not subject to suspension.

Both the parallel and serial queues are subject to **time limits**. Your job will terminate automatically if it exceeds its allotted time, which is calculated from when it starts (not when it's submitted).

short: one hour limit
normal: one day limit
long: one week limit
unrestricted: no limit

The longer the time limit, the longer it will take for your jobs to start running. When the cluster is busy (about three quarters of the time), you will find that jobs submitted to normal_parallel will run within a day. (You will get a sense of how long your jobs will take as you go along. The molecules in this tutorial will take less than an hour.)

Both the parallel and serial queues target eight-core nodes. One consideration is that parallel queues run jobs "exclusively," which means that only one job can run on them at a time. Therefore, you should always submit 8-core jobs to the parallel nodes. The serial nodes do not have this restriction, so the fewer nodes you request, the higher the chances are that there will be a node somewhere that can accomodate your request.

What is the total activity on the cluster?

```
[ekwan@iliadaccess01 ~]$ busers allusers

USER/GROUP        NJOBS   PEND    RUN  SSUSP  USUSP    RSV
allusers          23416  14599   8398    249      0    170
jwzorek            1072    768    304      0      0      0
```

There are about 15 000 cores worth of jobs pending right now, which means the cluster is busy (but not super busy). Joe and I are working on a big project right now, so we have quite a few jobs waiting.

Managing Your Jobs

Q: How do I see what jobs I'm running right now?

```
[ekwan@iliadaccess01 ~]$ bjobs -u jwzorek -w -a

JOBID    USER    STAT  QUEUE           FROM_HOST     EXEC_HOST   JOB_NAME                                    SUBMIT_TIME
24467474 jwzorek RUN   dae             iliadaccess01 8*dae022    monorhizopodin_OPLS_low_39                  Jan 28 14:59
24467480 jwzorek RUN   dae             iliadaccess01 8*dae012    monorhizopodin_OPLS_low_4                   Jan 28 14:59
24467481 jwzorek RUN   dae             iliadaccess01 8*dae013    monorhizopodin_OPLS_low_4_unsolv            Jan 28 14:59
24467486 jwzorek RUN   long_parallel   iliadaccess01 8*hero0408  monorhizopodin_OPLS_low_7                   Jan 28 14:59
24467487 jwzorek RUN   long_parallel   iliadaccess01 8*hero2409  monorhizopodin_OPLS_low_7_unsolv            Jan 28 14:59
25611512 jwzorek PEND  normal_parallel iliadaccess02    -        still_38_OPLS_1_prod                        Feb  7 20:04
25611515 jwzorek PEND  normal_parallel iliadaccess02    -        still_38_OPLS_low_11_prod                   Feb  7 20:04
25611516 jwzorek EXIT  normal_parallel iliadaccess02 8*hero1916  still_39_OPLS_low_1_prod                    Feb  7 20:04
24467468 jwzorek DONE  long_parallel   iliadaccess01 8*hero1405  monorhizopodin_OPLS_low_36                  Jan 28 14:59
```

(This is also a partial listing.)

(1) I needed the "-u jwzorek" flag because I'm not running any jobs at the moment. That tells it to display only the jobs being run by jwzorek. The -w flag requests a "wide" listing, which helps me see the long names of the jobs. Long, descriptive names, preferably connected with a painfully detailed and rigorous filing system are a very good idea, particularly if you are going to run a lot of jobs at once. The -a flag will show "all" the jobs, including the ones that have recently crashed or completed.

(2) The status column tells you if the jobs are running (RUN), waiting in a queue (PEND), have recently completed successfully (DONE), or have recently crashed (EXIT). Every time a job terminates, you get an email from LSF.

(3) EXEC_HOST tells you where the job is running right now. For example, the first job is running on eight cores on dae022.

(4) If you are impatient and want to see why your job is still pending:

```
[ekwan@iliadaccess01 ~]$ bjobs -l 25611512

Job <25611512>, Job Name <still_38_OPLS_1_prod>, User <jwzorek>, Project <defau
                     lt>, Status <PEND>, Queue <normal_parallel>, Command <#!/b
                     in/bash; #BSUB -q normal_parallel...
Mon Feb  7 20:04:44: Submitted from host <iliadaccess02>, CWD <$HOME/jobs1/stil
                     l_38_OPLS_1_prod>, Exclusive Execution, 8 Processors Reque
                     sted, Requested Resources <span[ptile=8]>;
 PENDING REASONS:
 Not specified in job submission: 902 hosts;
 Job's requirement for exclusive execution not satisfied: 422 hosts;
 Not enough hosts to meet the job's spanning requirement: 7 hosts;
 Job slot limit reached: 7 hosts;
 Closed by LSF administrator: 16 hosts;
 Unable to reach slave batch server: 2 hosts;
 Load information unavailable: 11 hosts;
```

Among a lot of other omitted output, this tells us that the job isn't running mostly because there aren't enough free nodes right now.

Managing Your Jobs

Occasionally, you will want to shuffle your jobs around. Here are some useful commands, each of which requires a jobid to tell it which job to work on. If you put in "0," it tries to apply the command to *all* your jobs.

bkill [jobid] - abort a certain job

Sometimes, a job will fail to terminate, even if it is done or you issue the bkill command manually. In that case, use the "-r" flag to kill the zombie job.

bstop [jobid] - pause a certain job; status changes to USUSP
bresume [jobid] - resume said job; status reverts to PEND or RUN, depending on what's going on

btop [jobid] - move the job to the top of the queue (at least, the top of the jobs from *you* in that queue)

You cannot alter anyone else's jobs.

To monitor the activity on a certain node that's running your job:

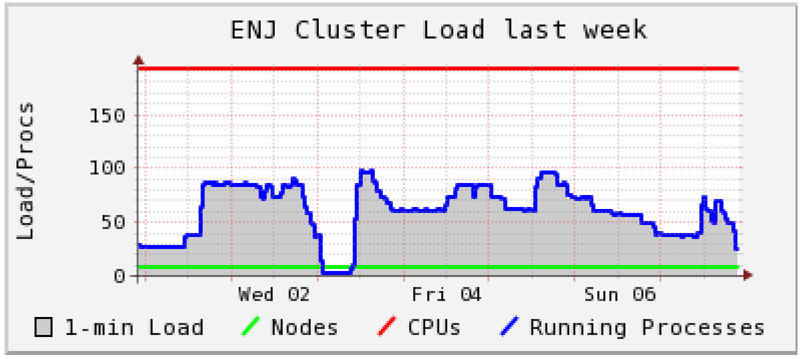
```
[ekwan@iliadaccess01 ~]$ ssh dae022

Last login: Mon Nov 15 01:05:53 2010 from iliadaccess02.rc.fas
harvard.edu

...

[ekwan@dae022 ~]$ top

...

  PID USER     VIRT  RES  SHR  S  %CPU %MEM TIME+        COMMAND
 1546 jwzorek 4468m 420m 3124 R 798.1  1.2 30642:03     l701.exe
13838 ekwan   10988 1136  680 R   3.9  0.0     0:00.03 top
24309 root    21760 5792 1524 S   1.9  0.0   132:22.94 lim
```

(There's not enough room here to see everything. The other columns are about other resource usage. For a full explanation, try "man top" to pull up the manual page for top.)

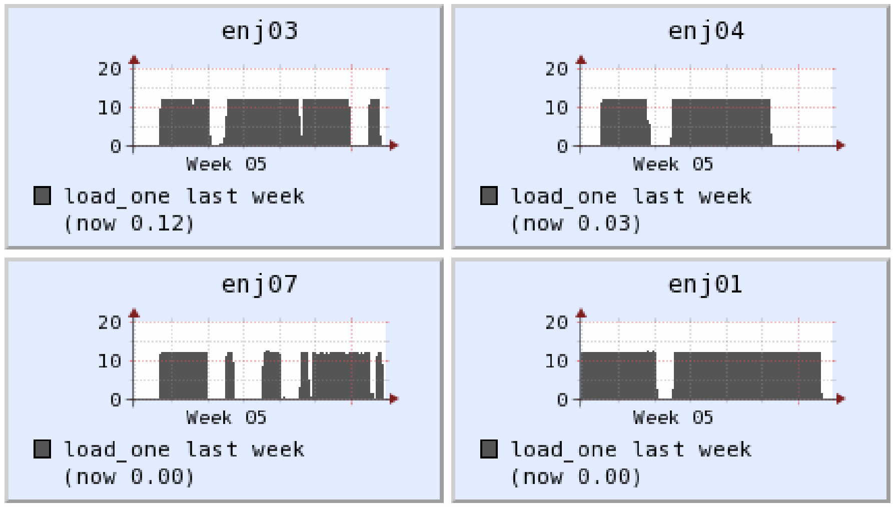
This confirms that a Gaussian subroutine, called a "link," is currently active and using all eight cores. Every link is associated with a particular part of Gaussian. In this case, link 701 is "one electon integrals first or second derivatives." A full listing can be found here:

http://www.gaussian.com/g_tech/g_ur/m_linklist.htm

Finally, you can look at what's going on with "Ganglia," which is available at http://software.rc.fas.harvard.edu/ganglia/:

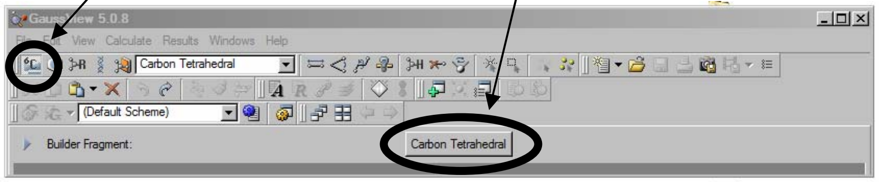


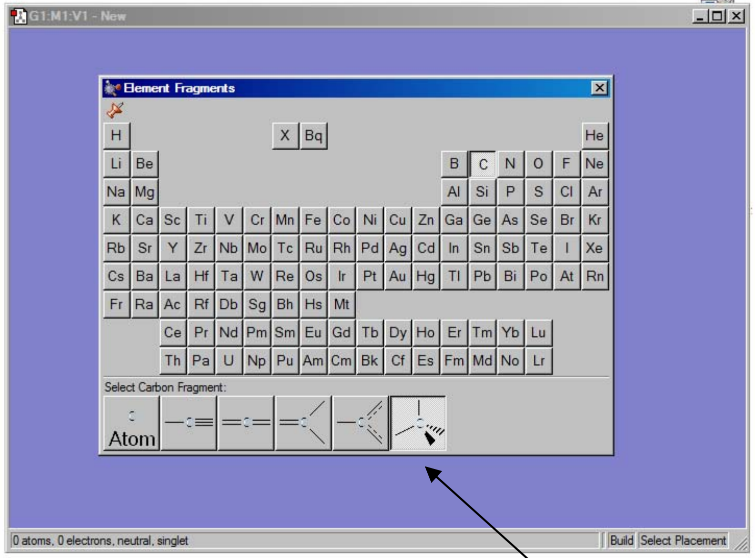
The above shows aggregate CPU usage; the below shows CPU usage per node. There are a lot of other options.



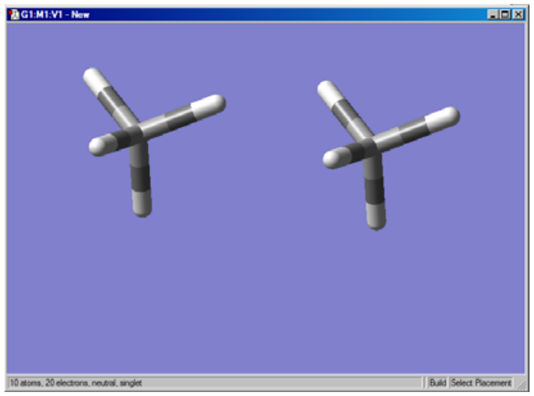
Test Case #1

I am now ready to take you through submitting your first job. Our first task is to draw axial 2-chlorotetrahydropyran in Gaussview. Click on the leftmost button in the toolbox that looks like "^{6}C" and then on "carbon tetrahedral."




G1:M1:V1 - New
Element Fragments
H X Bq He
Li Be B C N O F Ne
Na Mg Al Si P S Cl Ar
K Ca Sc Ti V Cr Mn Fe Co Ni Cu Zn Ga Ge As Se Br Kr
Rb Sr Y Zr Nb Mo Tc Ru Rh Pd Ag Cd In Sn Sb Te I Xe
Cs Ba La Hf Ta W Re Os Ir Pt Au Hg Tl Pb Bi Po At Rn
Fr Ra Ac Rf Db Sg Bh Hs Mt
Ce Pr Nd Pm Sm Eu Gd Tb Dy Ho Er Tm Yb Lu
Th Pa U Np Pu Am Cm Bk Cf Es Fm Md No Lr
Select Carbon Fragment:
Atom
0 atoms, 0 electrons, neutral, singlet
Build Select Placement

A periodic table comes up. You can select different valences for the various elements. Click somewhere on the blank window behind the periodic table. A methane molecule appears. Clicking somewhere else produces a new methane molecule:



You have various options for manipulating the whole screen. If you are *not* clicking on any part of a bond or molecule:

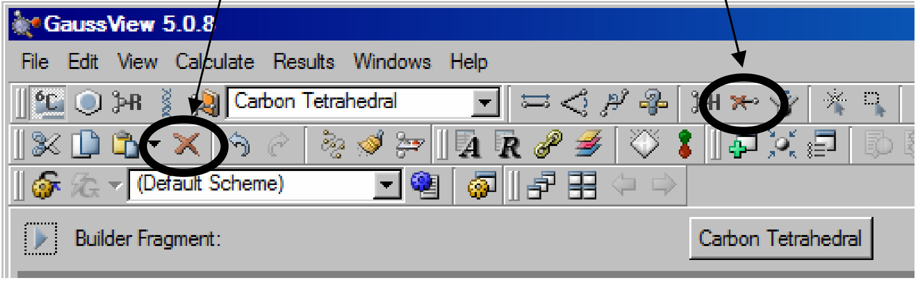
roll mouse wheel: zoom in or out
click and drag: rotate entire view
control-click and drag: rotate entire view in plane
shift-click and drag: move entire view

If you *are* clicking on part of the molecule:

alt-drag: rotate that molecular fragment
shift-alt-drag: move that molecular fragment

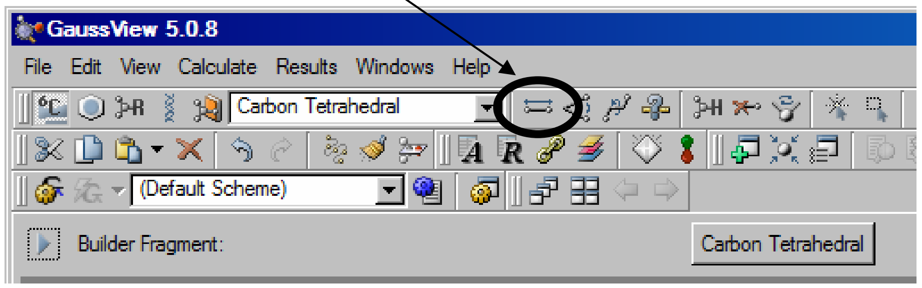
To delete atoms, click on the small X tool button (not the big X button, which deletes the entire molecule!). Click on individual atoms to delete them.

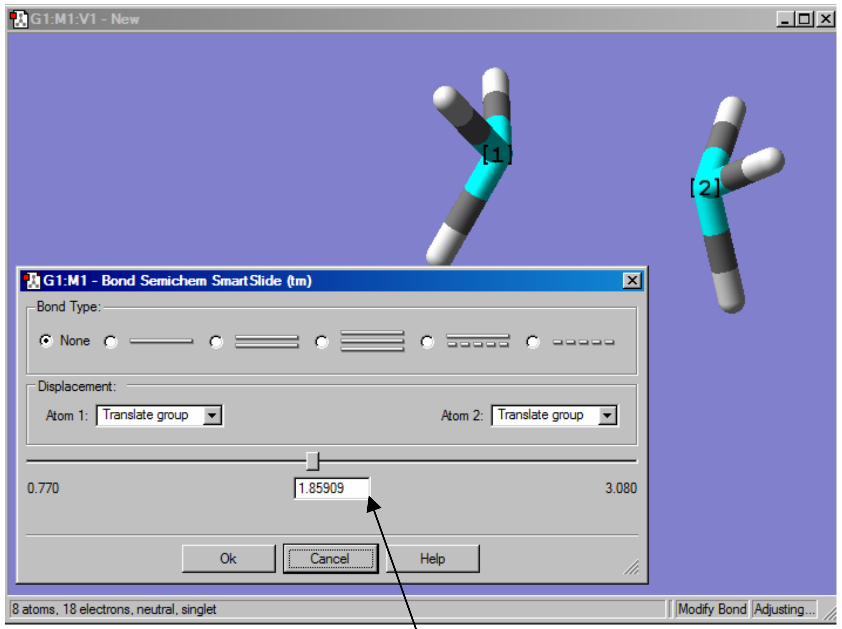
delete whole molecule *use this*



You can connect two atoms with a bond using the single bond tool. However, keep in mind that the bonds you draw are a convenient tool for visualizing the connectivity in the molecule, and *not* relevant to the energy calculation (unless you're doing molecular mechanics). Click on two atoms, and then on the tool, which pulls up this dialog box:

single bond tool





Bond Type: (✓ None)

Displacement: Atom 1: Translate group Atom 2: Translate group

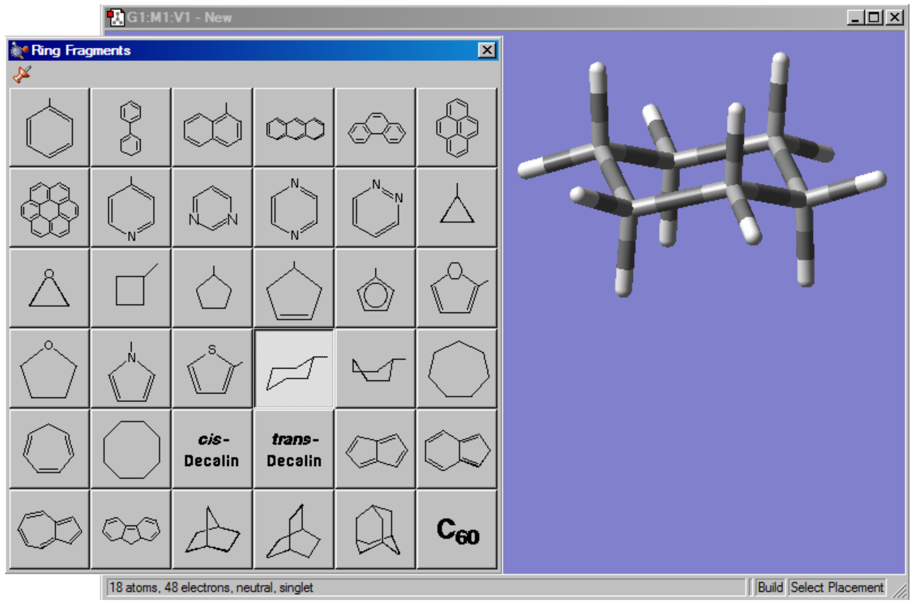
0.770 1.85909 3.080

8 atoms, 18 electrons, neutral, singlet | Modify Bond | Adjusting...

adjust bond distance here (angstroms by default)

The two adjacent tools adjust the bond angle (select three atoms) and dihedral angle (select four atoms). The tool with a question mark in it will show you the relevant bond distance, angle, or dihedral in the status bar if you select the relevant atoms.

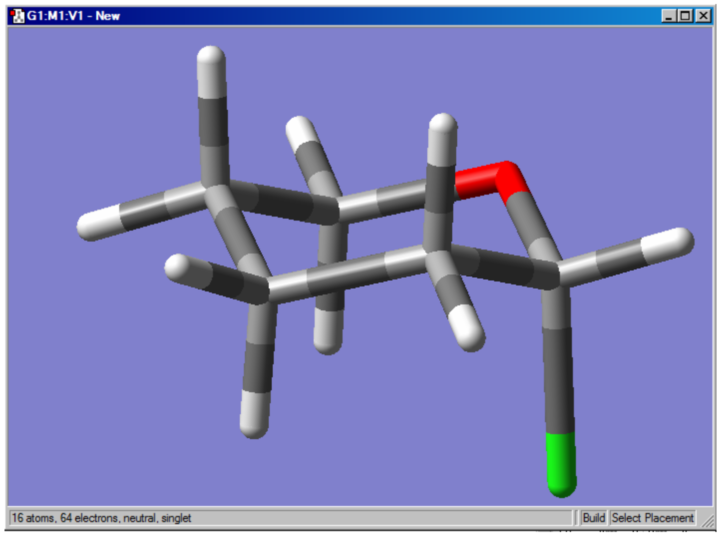
Clearly, this is a cumbersome way to go about your business. For the more common fragments, a template tool is available by clicking on the benzene ring next to the "^{6}C" button:



18 atoms, 48 electrons, neutral, singlet | Build | Select Placement

Clicking in the blue window will paste in a cyclohexane ring, or whatever other template you desire. Use these tools to draw axial 2-chlorotetrahydropyran. (Clicking on an existing atom with a different atom type will convert it to the new atom type. Use this to construct the ether.)

My result looks like this:



Next, we need to save the file. Save it as a Gaussian input file (*.gjf) and check "Write Cartesians." This generates a text file that you can open in Wordpad or some other text editor:

```
%chk=C:\Users\Eugene Kwan\Desktop\test.chk
# hf/3-21g geom=connectivity

Title Card Required

0 1
 C                 -0.04198489    0.54090527    0.00000000
 C                  1.47312111    0.54090527    0.00000000
 C                  2.02505211    1.95198327    0.00000000
 C                 -0.03973589    2.75718127    1.16017200
 C                 -0.59253589    1.34656027    1.15887600
 H                  3.14365111    1.91799227    0.06271400
 H                  1.84567011   -0.00472873    0.90656200
 H                  1.84841511   -0.00895373   -0.90191000
 H                 -0.41467289    0.97393427   -0.96538500
 H                 -0.41758189   -0.51331473    0.06350200
 H                 -0.41505589    3.30660927    2.06228600
 H                 -0.41159089    3.30378227    0.25384900
 H                 -0.32653189    0.84083827    2.12419100
 H                 -1.71108389    1.38162127    1.09393800
 O                  1.47538911    2.75652027    1.16066100
 Cl                 1.60357089    2.74535830   -1.51344298

 1 2 1.0 5 1.0 10 1.0 9 1.0
 2 3 1.0 7 1.0 8 1.0
 3 6 1.0 15 1.0 16 1.0
 4 5 1.0 11 1.0 12 1.0 15 1.0
 5 14 1.0 13 1.0
 6
 7
 8
 9
 10
 11
 12
 13
 14
 15
 16
```

This is the "connectivity table." It tells Gaussian about where all the bonds are. This can be useful for molecular mechanics or in some rare cases for troubleshooting a problem with the coordinate system, but we don't need it here, so delete it.

The coordinates will be different for the molecule you draw, due to translation and rotation of the whole molecule. However, Gaussian will rotate the entire specification to the "standard basis" before running the calculation, so this is of no consequence.

The **header** tells Gaussian what to do. The default header is not what we want, so change it to this:

```
%chk=checkpoint.chk
%mem=3GB
%nprocshared=8
#t opt freq=noraman b3lyp/6-31g(d) pop=nbo

title

0 1
```

```
%chk=checkpoint.chk
%mem=3GB
%nprocshared=8
#t opt freq=noraman b3lyp/6-31g(d) pop=nbo

title

0 1
```

(1) **Checkpoints:** Gaussian stores results every so often in a checkpoint file so that the calculation can be restarted if it crashes. The first line tells it to store the checkpoint for this job in checkpoint.chk.

(2) **Memory:** This is the amount of RAM that the job will use. This is not to be confused with the amount of swap space, which will be dynamically allocated. Volatile RAM is 10-100 times faster than more permanent hard disk space. Most nodes have at least 24 GB available, but 3 GB is perfectly sufficient for most jobs. Allocating too much memory can slow a calculation down. In general, the number of integrals a calculation will require is far greater than the amount that can be stored in RAM. Thus, a “direct” algorithm is used in which integrals are calculated, and then forgotten, on the fly. This turns out to be faster than swapping them to disk.

(3) **Shared Processors:** This tells Gaussian that you want it to use up to 8 processors. Occasionally, you will want to cut this number. Because there are some overhead costs associated with parallel computing, an eight-core job is not eight times faster than a one-core job. However, the parallel code in Gaussian is very good and the extra speedup is worth it, particularly for calculations where you want a result quickly.

(4) **Route Card:** This is the line that starts with a pound (#) sign. This is the crucial line that tells Gaussian what to do with the geometry you have given it. If you need to write more, you can continue it onto the next line.

#: Indicates start of route card
t: Requests “terse” output. Nothing gives a standard amount of output while **p** gives a verbose level of output.
opt: Requests an optimization to a ground state
freq: Requests a frequency job. The sub-keyword “noraman” tells it not to compute the Raman transitions.
b3lyp/6-31g(d): method and basis set
pop=nbo: ask for a standard orbital analysis

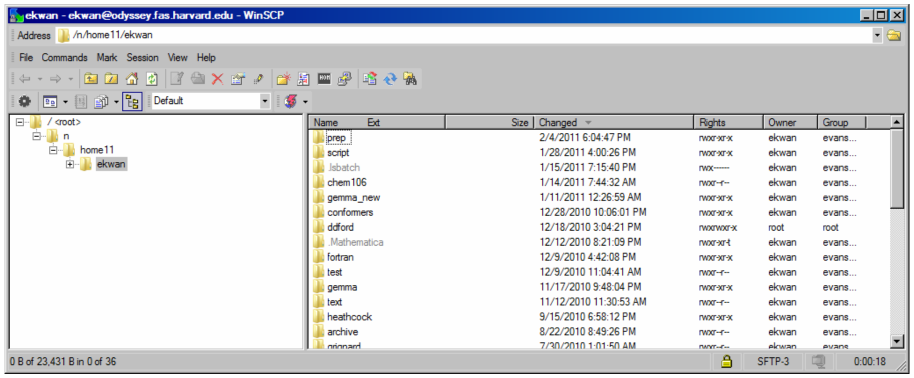
I don’t have room to explain all the Gaussian keywords here. However, a detailed list is available online here:

www.gaussian.com/g_tech/g_ur/l_keywords09.htm

(5) **Title Card:** This is the title of the job. You don’t have to change this if you don’t want to. I keep track of jobs by their filename instead.

(6) **Charge and Multiplicity:** This tells Gaussian how many electrons to use. 1 1 would indicate a positively charged singlet. -1 2 would indicate a negatively charged doublet. Calculations involving open shell species are significantly more complicated, and should only be undertaken if you know what you’re doing. It’s not simply a question of plugging in the geometry and getting an optimized ground state result.

Now, we are ready to submit the job. You will need to use your secure FTP client to do this. Use odyssey.fas.harvard.edu as the domain name and enter in the same password. You will need a new passcode, however. Note that the same passcode cannot be used for two logins; if you use one for logging into the terminal, then you need to wait for a new one to start an SFTP session. On my screen (different if you use a different client):


ekwan - ekwan@odyssey.fas.harvard.edu - WinSCP
Address
File Commands Mark Session View Help
Default
/ <root>
n
home11
ekwan
SFTP-3
0:00:18

Address: /n/home11/ekwan

Name	Ext	Size	Changed	Rights	Owner	Group
prep			2/4/2011 6:04:47 PM	rwxr-xr-x	ekwan	evans...
script			1/28/2011 4:00:26 PM	rwxr-xr-x	ekwan	evans...
.lsbatch			1/15/2011 7:15:40 PM	rwx------	ekwan	evans...
chem106			1/14/2011 7:44:32 AM	rwxr--r--	ekwan	evans...
gemma_new			1/11/2011 12:26:59 AM	rwxr-xr-x	ekwan	evans...
conformers			12/28/2010 10:06:01 PM	rwxr-xr-x	ekwan	evans...
ddford			12/18/2010 3:04:21 PM	rwxrwxr-x	root	root
.Mathematica			12/12/2010 8:21:09 PM	rwxr-xr-t	ekwan	evans...
fortran			12/9/2010 4:42:08 PM	rwxr-xr-x	ekwan	evans...
test			12/9/2010 11:04:41 AM	rwxr--r--	ekwan	evans...
gemma			11/17/2010 9:48:04 PM	rwxr-xr-x	ekwan	evans...
text			11/12/2010 11:30:53 AM	rwxr--r--	ekwan	evans...
heathcock			9/15/2010 6:58:12 PM	rwxr-xr-x	ekwan	evans...
archive			8/22/2010 8:49:26 PM	rwxr--r--	ekwan	evans...
grignard			7/30/2010 1:01:50 AM	rwxr--r--	ekwan	evans

0 B of 23,431 B in 0 of 36

The left-hand panel shows the directory structure of the remote session. The right-hand panel shows the contents of the active directory. The top bar shows the current location. Note that the current location in your SFTP session is *not* the same as in your current SSH session.

Copy the input file (.gjf extension) into a directory of your choice. Be sure to transfer it as **text not binary** or you will encounter some weird errors. Typically, you will not want to copy it into your root. Here are some useful file management commands you can use in your SSH session:

pwd – ask for the current directory
mkdir [dirname] – create directory dirname
rm –rf [dirname] – delete directory dirname and anything in it
rm [filename] – delete filename

Troubleshooting Note: Occasionally, there is still a problem with the format of the carriage returns and line breaks when you transfer a file from your computer to the cluster. In that case, try the dos2unix command (use "man dos2unix" for more details).

Now that the job is copied onto the cluster, we can take a look at it:

```
[ekwan@iliadaccess01 chem106]$ ls -l
total 160
-rw-r--r-- 1 ekwan evans_lab 1044 Oct 19 11:00 2-
chloroTHP_ax.gjf
-rwxr--r-- 1 ekwan evans_lab  757 Feb 25  2010 analyze.sh
-rwxr--r-- 1 ekwan evans_lab 1436 Feb 25  2010 eek.sh
drwxr-xr-x 2 ekwan evans_lab 2048 Nov 24 18:02 jobs
drwxr-xr-x 2 ekwan evans_lab 7168 Jan 15 19:15 output
-rwxr--r-- 1 ekwan evans_lab  347 Jan 16  2010 submit.sh
-rwxr--r-- 1 ekwan evans_lab  521 Jan 13 07:50 template.sh
```

As you can see, the .gjf file is now in the directory. The filenames appear in the rightmost column. Their *permissions* appear in the leftmost column. The dates the files were last modified are also shown, along with their sizes in bytes.

To submit the job, you can invoke the submission script I have written. (A script is a small program written, in this case, in bash, which is a high-level, interpreted language.) It performs a number of repetitive housekeeping duties that you will not want to be bothered with. In particular, it runs your job in a separate directory with the same name as the input file, tells LSF how many cores you want and which queue to submit the job to, creates a scratch directory on the local node, etc.

To use the scripts, you will need to transfer them onto the cluster as well. They are available on the course website. You will need to execute the command "chmod u+rwx *.sh" before you can run the scripts. This will give the computer permission to execute the programs (something of a safety feature).

To submit the job, do this:

```
[ekwan@iliadaccess01 chem106]$ ./submit.sh 8 short_parallel

Beginning batch submission process...
Each job is being submitted to 8 processors in the queue
short_parallel.

Submitting job file 2-chloroTHP_ax.gjf...
Job <13724321> is submitted to queue <short_parallel>

Job submission complete.
```

This will submit every .gjf file in the directory into the queue short_parallel, requesting 8 cores for every job. (*Caution!* Double check your job file before you submit the job. Gaussian will be very unhappy if you make any syntax errors, and your job will crash or do something you don't expect it to.)

```
[ekwan@iliadaccess01 chem106]$ bjobs -w

13724321   ekwan RUN   short_parallel        iliadaccess
8*hero1807    2-chloroTHP_ax   Jan 28 14:59
```

As you can see, the job is now running in short_parallel. You will see its status as PEND if it's waiting. If you suddenly realize you made a mistake, use "bkill [jobid]" to cancel the job.

Eventually, your job will run and complete and you will get a long email from LSF full of gibberish. I like these notifications, but filter them away from my inbox. At that point, an output file (in this case, 2-chloroTHP_ax.out) will be copied to the output directory. Note that you will run into a problem if there is no output directory present; use "mkdir output" to make such a directory if it does not already exist. You only have to do this once per directory containing the various scripts.

In this case, my output directory is ~/chem106/output. ~ represents *your* home directory or root. (The *actual* root, /, is not a good place to put things.) Let's go there now:

```
[ekwan@iliadaccess01 chem106]$ pwd
/n/home11/ekwan/chem106
[ekwan@iliadaccess01 chem106]$ cd output
[ekwan@iliadaccess01 output]$ ls
output.txt                  2-chloroTHP_ax.out
```

As you can see, the output file is there. The script I wrote automatically extracts the energies. To save time, I ran the jobs for both the axial and equatorial:

```
[ekwan@iliadaccess01 chem106]$ more output.txt
*************************************
Job 2-chloroTHP_ax started at...
Tue Oct 19 12:01:27 EDT 2010

Job finished at:
Tue Oct 19 12:04:11 EDT 2010

Job terminated normally...

Final energy:
SCF Done: E(RB3LYP) = -731.374863805 A.U. after 1 cycles

No imaginary frequencies found.
 Sum of electronic and thermal Free Energies=     -731.267453
*************************************
```

electronic energy

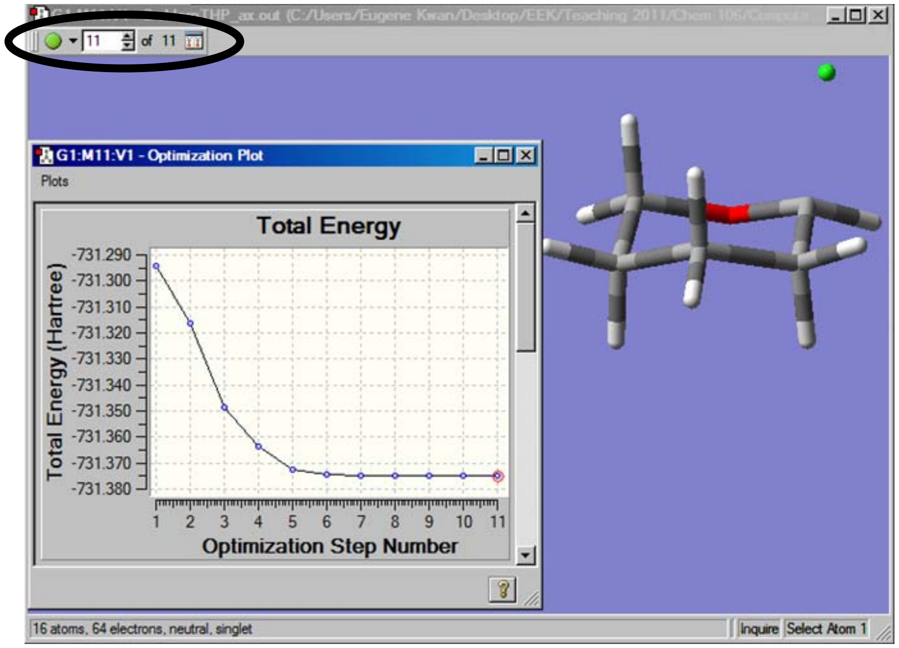
free energy

this is a true local minimum

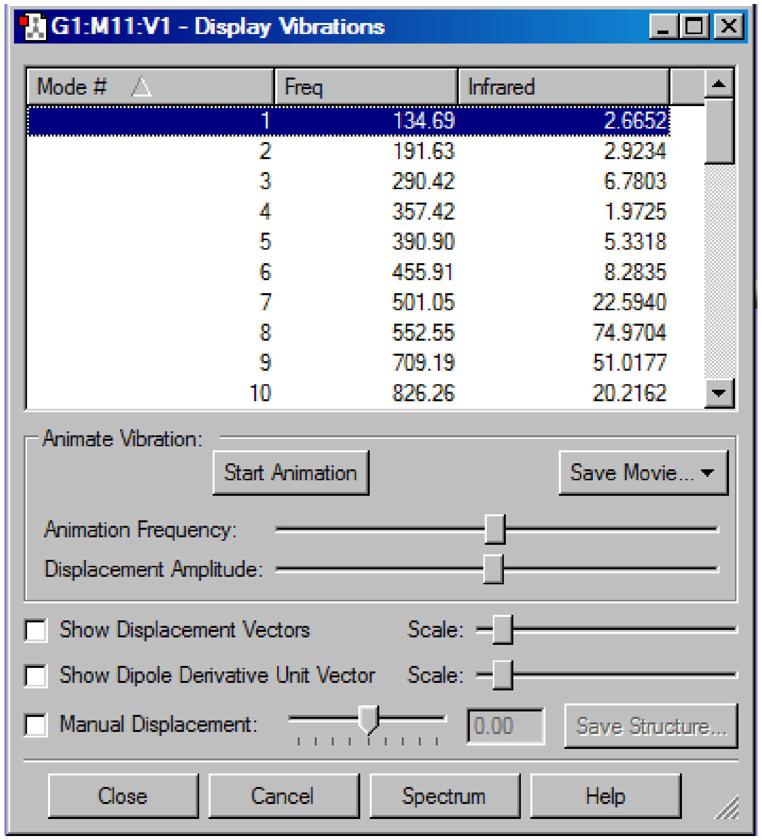
```
*************************************
Job 2-chloroTHP_eq started at...
Tue Oct 19 12:04:38 EDT 2010

Job finished at:
Tue Oct 19 12:06:40 EDT 2010

Job terminated normally...

Final energy:
SCF Done: E(RB3LYP) = -731.368902477 A.U. after 1 cycles

No imaginary frequencies found.
 Sum of electronic and thermal Free Energies=     -731.262071
*************************************
```

For reasons that are not clear to me, all the energies are reported in hartree. To find the energy difference in kcal/mol, you will need to subtract the two numbers and multiply by 627.509469. In this case, one finds that the axial conformer is more stable by 3.4 kcal/mol.

You can also have a look at the optimized geometries in GaussView. To open the file, make sure you select “Gaussian Output Files (*.out *.log)” and “Read Intermediate Geometries (Gaussian Optimizations Only).”



To see all the intermediate geometries, you can click through the circled box. As you can see, this optimization went smoothly, converging nicely to a stationary point. You can also check to make sure that no imaginary frequencies were found by going Results…Vibrations:

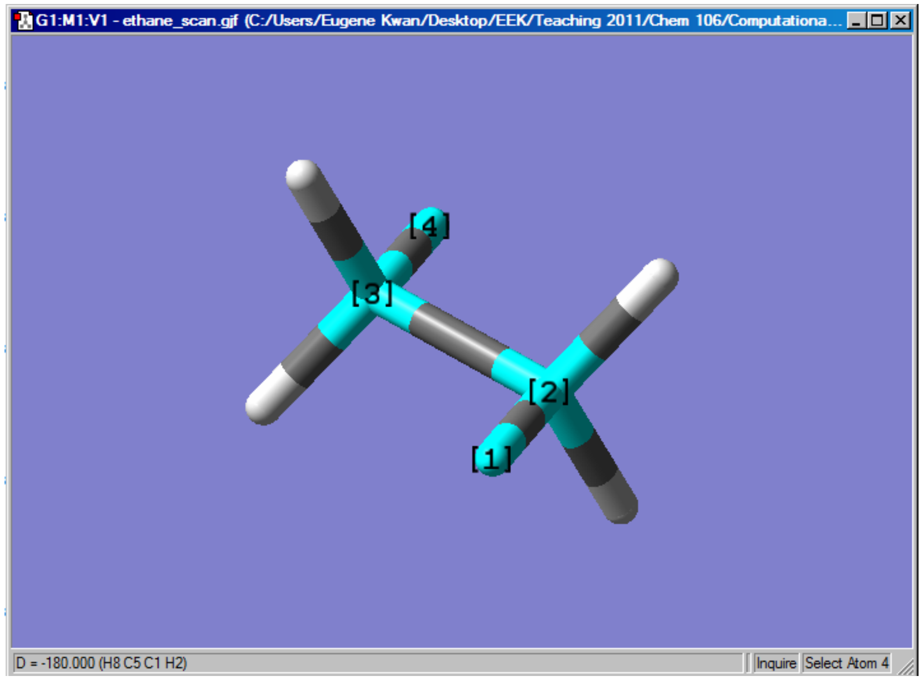


Mode #	Freq	Infrared
1	134.69	2.6652
2	191.63	2.9234
3	290.42	6.7803
4	357.42	1.9725
5	390.90	5.3318
6	455.91	8.2835
7	501.05	22.5940
8	552.55	74.9704
9	709.19	51.0177
10	826.26	20.2162

As before, there aren’t any negative numbers in the frequency column, so this is a real local minimum. This information is also captured by the output.txt file when it says “No imaginary frequencies found.”

Notice that there’s no bond drawn between the carbon and the chlorine. This is because Gaussian doesn’t know anything about where the bonds are. GaussView simply draws in bonds when the distances between atoms are within certain thresholds. The C-Cl bond is abnormally long due to the anomeric effect, so it is not recognized as a bond. However, the lack of a line connecting the C and Cl has no significance.

Finally, we want to find a transition state for the rotational barrier in ethane. As I mentioned already, the most robust way to find transition states is to perform a series of constrained optimization where the reacting coordinate is frozen in increments. For many reactions, this is a forming bond distance. In this case, it is a bond torsion. Draw ethane and use the question mark tool to identify the atom numbers of a H-C-C-H torsion:



G1:M1:V1 - ethane_scan.gif (C:/Users/Eugene Kwan/Desktop/EEK/Teaching 2011/Chem 106/Computationa...

D = -180.000 (H8 C5 C1 H2)

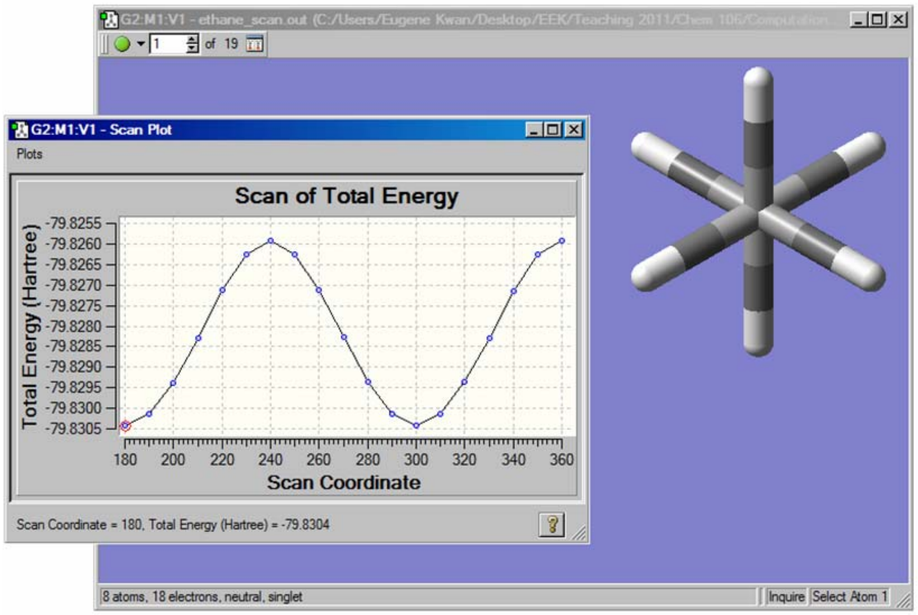
Now, we need to tell Gaussian to perform a constrained optimization. In the world of Gaussian, this is called a "modredundant optimization." The notation

D 8 5 1 2 S 18 10.000000

means to scan the dihedral (**D**; use B for bond and A for angle) angle between atoms 8-5-1-2 eighteen times in steps of 10 degrees. If you scan too coarsely, you can have "kinking" problems where the energy profile you get is not smooth. Here is my input file:

```
%chk=checkpoint.chk
%mem=3GB
%nprocshared=8
#t opt=modredundant b3lyp/6-31g(d) pop=none

title

0 1
 C                 -1.27672721    1.36139405    0.37637410
 H                 -0.91105109    1.81418310   -0.52149173
 H                 -2.00301536    0.61545024    0.12944624
 H                 -1.72899504    2.10885135    0.99416159
 C                 -0.10678799    0.70753803    1.13486901
 H                  0.34547984   -0.03991928    0.51708152
 H                  0.61950015    1.45348183    1.38179688
 H                 -0.47246412    0.25474897    2.03273484

D 8 5 1 2 S 18 10.000000
```

This results in a nice energy profile (Results…Scan):



G2:M1:V1 - ethane_scan.out (C:/Users/Eugene Kwan/Desktop/EEK/Teaching 2011/Chem 106/Computation...

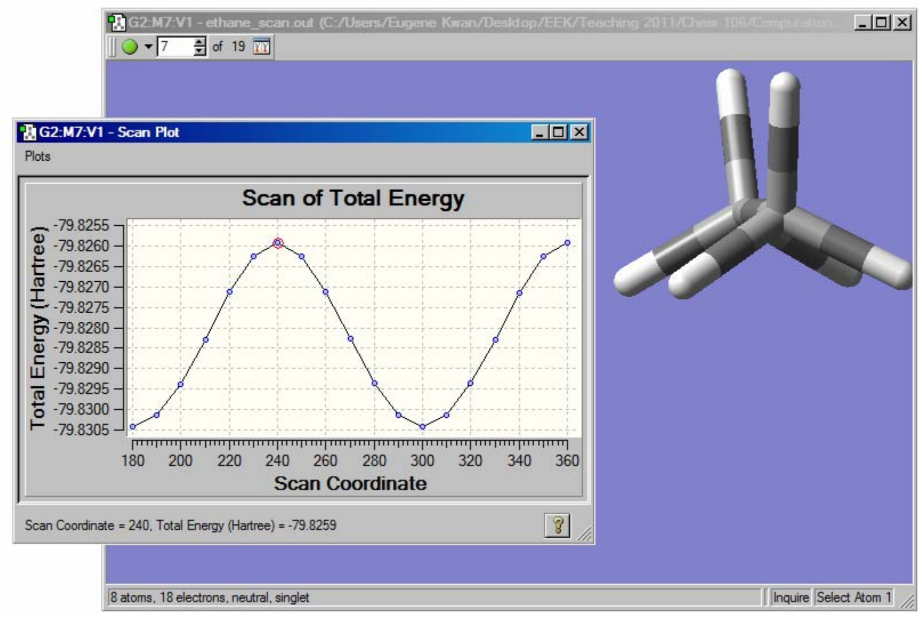
1 of 19

G2:M1:V1 - Scan Plot

Scan Coordinate = 180, Total Energy (Hartree) = -79.8304

8 atoms, 18 electrons, neutral, singlet

However, we are not done yet. Just because we have found a maximum in the energy profile is no guarantee it is a true transition state. For that, we need the gradient to be zero (or at least below the convergence threshold) and there to be exactly one imaginary frequency. To do this, save the geometry corresponding to an energy maximum as a new .gjf file:



Notice that the energy maximum, is, in fact, an eclipsed conformation. This is a good sign. We will need a new header to request an **unconstrained** transition state search:

```
%chk=checkpoint.chk
%mem=3GB
%nprocshared=8
#t opt=(calcfc,ts,noeigentest) freq=noraman b3lyp/6-31g(d)
pop=nbo

title

0 1
```

Take note of the sub-keywords under the main **opt** keyword.

ts: Request optimization to a first-order saddle point (i.e., a transition state)

calcfc: Perform a frequency job at the beginning of the calculation (i.e., calculate the Hessian). An accurate Hessian is **required** for any TS search. Occasionally, you can read the frequencies in from the checkpoint file; in that case, you should use the **readfc** sub-keyword. So you must either have calcfc or readfc for any transition state search.

noeigentest: Suppress curvature testing. You should turn this on to avoid crashes. (The manual recommends curvature testing in most cases, but this seems to be an anachronism.)

We also need **freq=noraman**, which will perform a second frequency calculation at the end of the job to verify that there is, in fact, only one negative frequency.

Let me re-iterate that:

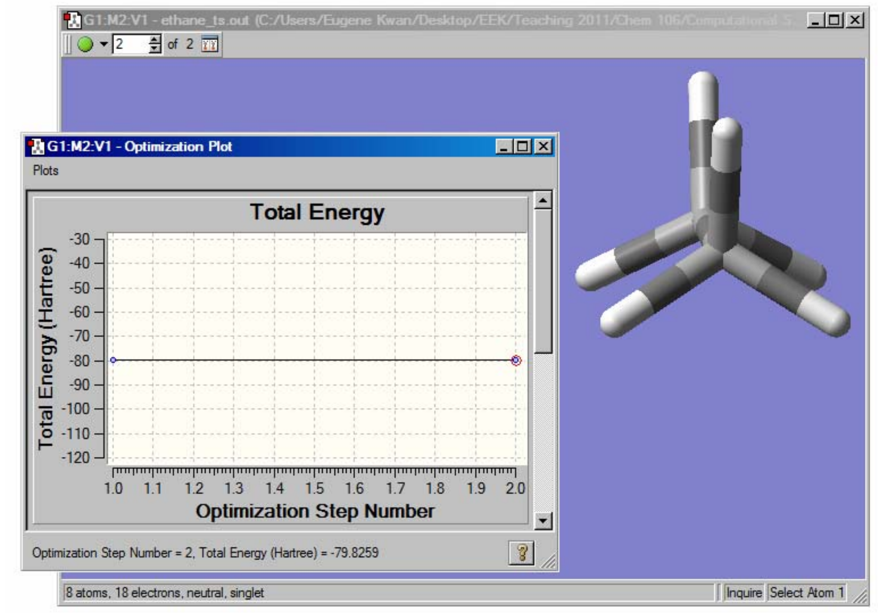
(1) You will only find the transition state if you are very, very close to it already.

(2) Even if you're very close to the correct geometry, transition state optimizations often fail.

(3) Check your syntax! Check your syntax!

Gaussian also has the ability to try and connect a starting material and product with a transition state directly; see the QST2 and QST3 methods under the "opt" keyword. I have never used these, so I can't offer any advice on them.

In my run, the transition state optimization converged nicely:


G1:M2:V1 - ethane_ts.out (C:/Users/Eugene Kwan/Desktop/EEK/Teaching 2011/Chem 106/Computational 5
2 of 2
G1:M2:V1 - Optimization Plot
Plots
Total Energy
Total Energy (Hartree)
Optimization Step Number
8 atoms, 18 electrons, neutral, singlet
Inquire
Select Atom 1

Optimization Step Number = 2, Total Energy (Hartree) = -79.8259

You should check on your optimizations as they progress to make sure they're not going "off the reservation." That just wastes valuable computer time. (You can just download the output file from the directory the job is running in, and ignore the error GaussView gives you when it opens the file.)

Good Signs:

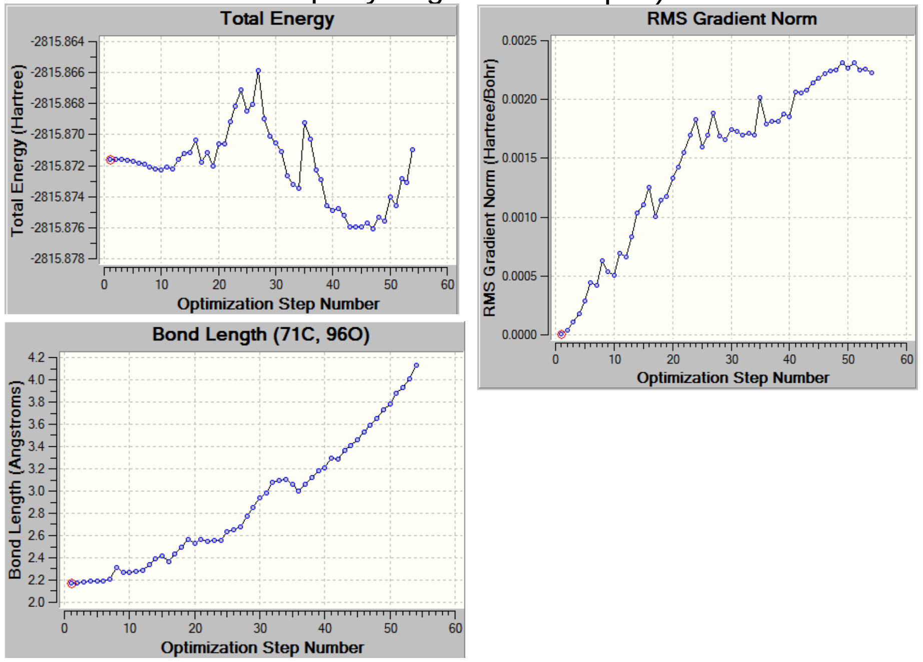
In an ideal world, every TS optimization will look like this one: the geometry starts off correct, and is minimized to a stationary point of correct curvature in one step. This gives rise to the flat energy optimization curve shown above. In other cases, the energy will go *up* and settle down into a transition state, while the RMS gradient will drop *down* and go to zero as the stationary point is reached. That is good news—the optimization algorithm is approaching the transition state along the reaction coordinate. In general, if your optimization does not converge to a transition state within 30 steps or so, you are in trouble.

Ambiguous:

Sometimes, the transition state energy will go down, but so will the gradient. This *may* result in a successful optimization. However, this often later results in…

Bad Signs:

The transition states "collapses" to starting material or product. In the case below, it opened up to starting material (use Plots…Molecular Property to get the third plot):



This concludes the tutorial. You should work through these examples yourself. Let me close by saying: **computations are hard work!** Don't think that it's easy just because you're sitting at your desk…