

Lab 1. Electronic Structure Calculations: Basics

In this lab you will:

- Perform basic *ab initio* calculations in Psi4.
- Learn how to create an input for evaluation of basic properties.
- Compute the energy differences between different conformations of the same molecules.
- Calculate the interaction energies between two molecules.

Authors: Lyudmila Slipchenko (lslipchenko@purdue.edu (<mailto:lslipchenko@purdue.edu>); ORCID: 0000-0002-0445-2990) and Victor H. Chavez (gonza445@purdue.edu (<mailto:gonza445@purdue.edu>); ORCID: 0000-0003-3765-2961)

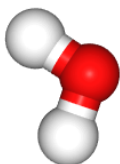
```
In [1]: ▶ 1 #You will need to import the following modules to run calculations.
2 #Scientific Python tools
3 import numpy as np
4 import pandas as pd
5
6 #Computational Chemistry Tools
7 import psi4
8
9 #Visualization tools
10 import matplotlib.pyplot as plt
11 import qcelemental as qc
```

Part A: Computing energies of different conformations

```
In [2]: ▶ 1 #Let us create a water molecule in psi4. We require a string with three main components:
2 #Charge and spin multiplicity
3 #Geometry (Z-matrix, xyz)
4 #Symmetry
5
6 bent_geometry = psi4.geometry("""
7 0 1
8
9 O
10 H 1 0.96
11 H 1 0.96 2 104.5
12
13 symmetry c1
14 """)
15
16 #Perform an energy calculation for water using:
17 e_bent = psi4.energy("HF/6-31G*", molecule=bent_geometry)
```

```
In [3]: ▶ 1 #The Psi4 geometry class gives the geometrical information to QCElemental for plotting.
2 print("Bent water molecule")
3
4 bent = qc.models.Molecule.from_data(bent_geometry.save_string_xyz())
5 bent.show()
```

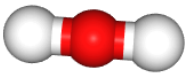
Bent water molecule



```
In [4]: 1 #Based on the example of the bent water, define a geometry and make an energy calculation for a linear molecule
2
3 #Input as cartesian coordinates for linear water molecule
4 linear_geometry = psi4.geometry("""
5 0 1
6
7 0
8 H 1 0.96
9 H 1 0.96 2 180.0
10 """)
11
12 #Energy calculation
13 e_linear = psi4.energy("HF/6-31G*", molecule=linear_geometry)
```

```
In [5]: 1 #Confirm that your geometry is correct by showing the structure
2
3 print("Linear water molecule")
4
5 linear = qc.models.Molecule.from_data(linear_geometry.save_string_xyz())
6 linear.show()
```

Linear water molecule



```
In [6]: 1 #Pandas is a library designed to work nicely with databases.
2 #If you followed the naming patterns for the variables, you should see a nice table with your results.
3
4 #Print results
5 water = pd.DataFrame(data = {'Linear':[e_linear], 'Bent':[e_bent]})
6 water.index=['Energy']
7 water
```

Out[6]:

	Linear	Bent
Energy	-75.948235	-76.010376

Questions:

a) Compare the total energy of bent and linear water. What are their relative energies? Express the relative energies in terms of kJ/mol, kcal/mol and in hartrees. Keeping in mind that the Boltzmann energy at room temperature $K_B T$, is the linear water conformation easily accessible at room temperature?

```
In [7]: 1 #Response
```

b) **Extra:** You can repeat the same steps with planar and pyramidal ammonia NH_3 to see if the "umbrella inversion" process is feasible at room temperature. Is tunnelling a factor in this inversion process?

```
In [8]: 1 #Responnse
```

Part B: Computing interaction energies

1. Compute the energies of two water clusters using the following given geometry at **HF/6-31G** level of theory:

```
O -0.089523 0.063946 0.08686
H 0.864783 0.058339 0.103755
H -0.329829 0.979459 0.078369
O 2.632273 -0.313504 -0.750376
H 3.268182 -0.937310 -0.431464
H 2.184198 -0.753305 -1.469059
```

2. Show the geometry of the cluster.
3. Make a table showing the energy of the monomers, cluster and the interaction energy

Hint: Compute the energy of the first monomer A (first three atoms), then compute the energy of the second monomer B (last three atoms), followed by the energy of the dimer AB all six atoms.

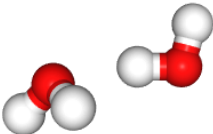
The total interaction energy:

$$E_{int} = E(AB) - (E(A) + E(B))$$

In [9]:

▶

```
1 #Response
2 #Each of the monomers in the cluster is a water molecue, we can then recycle the energy of the previous
3
4 #Calculate cluster energy
5 cluster_geometry = psi4.geometry("""
6 0 1
7     O    -0.089523    0.063946    0.086866
8     H     0.864783    0.058339    0.103755
9     H    -0.329829    0.979459    0.078369
10    O     2.632273   -0.313504   -0.750376
11    H     3.268182   -0.937310   -0.431464
12    H     2.184198   -0.753305   -1.469059
13 """)
14
15
16 cluster = qc.models.Molecule.from_data(cluster_geometry.save_string_xyz())
17 cluster.show()
```



In [10]:

▶

```
1 e_cluster = psi4.energy('HF/6-31G*', molecule=cluster_geometry)
```

In [11]:

▶

```
1 interaction = pd.DataFrame(data = {'Monomer':[2*e_bent], 'Cluster':[e_cluster]})
2 interaction.index=['Energy']
3 interaction['$E_{int}$ (Hartrees)'] = interaction['Monomer'] - interaction['Cluster']
4 interaction
```

Out[11]:

	Monomer	Cluster	E_{int} (Hartrees)
Energy	-152.020753	-152.029933	0.00918

Questions

a) Show the three different energies (monomer, cluster, interaction) in hartrees, kJ/mol and kcal/mol. (Pandas has the ability of multiplying a whole column/row by a constant)

In [12]:

▶

```
1 #Response
```

Part C: Fun with formaldehyde.

1. With the following geomtry build a formaldehyde molecule:

H	-0.0000000	0.9275885	1.1766889
C	-0.0000000	0.0000000	0.6019825
H	-0.0000000	-0.9275885	1.1766889
O	0.0000000	-0.0000000	-0.6001772

2. Perform energy calculation of the molecule at the HF/6-31G* level of theory.

3. Perform vibrational frequency calculation of formaldehyde at this geometry at the same level.

In this part you will visualize fundamental volumetric information of a molecule, being the electron density and the molecular orbitals. In order to do this, Psi4 needs to know in advance that you are interested in these quantities. The keyword is as follows:

```
In [13]: ▶ 1 psi4.set_options({'cubeprop_tasks': ['orbitals', 'density'],
2                  'cubeprop_orbitals': [12],
3                  })

In [14]: ▶ 1 #You can proceed and define the geometry of a formaldehyde molecule.

In [15]: ▶ 1 #Response
2
3
4 formal = psi4.geometry("""
5 0 1
6  H  -0.0000000  0.9275885  1.1766889
7  C  -0.0000000  0.0000000  0.6019825
8  H  -0.0000000 -0.9275885  1.1766889
9  O   0.0000000 -0.0000000 -0.6001772
10
11 symmetry c1
12 """)
```

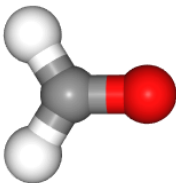
The information about the densities and orbitals are contained within Psi4 in a "wavefunction" object. This can be obtained from your usual energy calculation simply by modifying slightly your syntax:

```
e_formal, wfn_formal = psi4.energy('HF/sto-3g', return_wfn=True, molecule = formal)
```


The "wfn_formal" contains much of the information required to perform the calculation, it can give you basic information about the geometry, but also contains important scf quantities like the one particle density matrix.

```
In [16]: ▶ 1 #Use the previous syntax to get the energy and wavefunction object for formaldehyde.
2 e_formal, wfn_formal = psi4.energy('HF/sto-3g', return_wfn=True, molecule = formal)

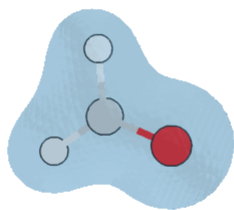
In [17]: ▶ 1 formal = qc.models.Molecule.from_data(formal.save_string_xyz())
2 formal.show()
```




```
In [18]: ▶ 1 #Let's visualize the previous quantities, we need the library "blobs".
2 #The syntax is the following, you need to use the wavefunction object you just created.
3 import blobs
4 cube = blobs.Cube(wfn_formal)
5
6 #By running this cell, blobs should generate a series of files containing the information:
7 #Density = Alpha density (Da), Beta density (Db), Total density(Dt), Density difference(Ds)
8 #Orbitals = {Psi_a_n_n-A.cube}_n with n being the number of orbital
```

In [19]: 

1	<code>#Plot of Density</code>
2	<code>cube.plot("Dt.cube")</code>



In [20]: 

1	<code>#Plots of orbitals</code>
2	<code>cube.plot("Psi_a_12_12-A.cube", cube_type="orbital")</code>



1. Use the previous syntax to visualize the molecular orbitals and save as an image the 8 occupied and 4 virtual (unoccupied) orbitals. Provide their orbital energies (in Hartrees), and assign the character (i.e., bonding/antibonding/lone pair, sigma or pi, etc.). The character may be ambiguous, but do your best. Based on this analysis, write down electronic configuration of CH_2O .

Hint: Orbital energies can be accessed from the wavefunction as in: `wfn_formal.epsilon_a().np`

In [21]: 

1	<code>#Response</code>
---	------------------------

2. Present a geometry in such a way that chemists (not computer), could understand it: write down bbond lenghts and valence angles, not xyz coordinates. Write down the nuclear repulsion and electronic energies in Hartrees (check the output in the terminal)

In [22]: 

1	<code>#Response</code>
---	------------------------

Frequency calculation

Perform vibrational frequency calculation of formaldehyde at equilibrium geometry at same level of theory (HF/6-31G*). Instead of using psi4.energy, replace and use psi4.frequencies. It is important that you obtain the wavefunction object just like you did for the cube files.

In [23]:

▶

```
1 #Tell psi4 that we want to store the information about the normal modes:
2 psi4.set_options({"NORMAL_MODES_WRITE" : True})
3
4 #Frequencies calculation
5 scf_e, scf_wfn = psi4.frequencies('HF/6-31G*', return_wfn=True)
```

Warning: used thermodynamics relations inappropriate for low-frequency modes: ['136.6732' '202.6502' '209.7802']

The information about the frequencies can be extracted from the wavefunction. Blobs is also use to examine this results.

In [24]:

▶

```
1 vib = blobs.Freq(scf_wfn)
```

In [25]:

▶

```
1 #Obtain the frequencies with:
2 vib.frequencies
```

Out[25]:

array([136.67319007+0.j, 202.65024466+0.j, 209.78018409+0.j,
 1332.712701 +0.j, 1374.87218355+0.j, 1660.66579307+0.j,
 1926.04823327+0.j, 3163.77229459+0.j, 3246.37254817+0.j])

In [26]:

▶

```
1 #Visualize the vibrations with:
```

In [27]:

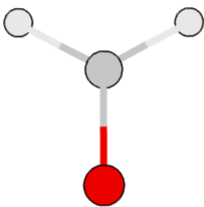
▶

```
1 vib.plot(vib=6)
```

Frequency: 1926.05 1/cm

Play

Stop



Inspect vibrational modes of the molecule, practice them over the weekend.

Provide a sketch of each mode, write down its frequency and assignment (symmetric or assymetric stretch, torsion, scissors, etc).

Comparison with to experiment. Compare your finding with available experimental data (geometry and frequencies). Are the calculated results accurate? How large are computational errors?

In [28]:

▶

```
1 #Response
```

In []:

▶

```
1
```