Homework 3
Lectures 4: Electronic Structure Models
(CBE 60553)

Prateek Mehta, Prof. William F. Schneider

Due: 02/12/2015

Here is an example input deck for a HFS/6-31G calculation on NH_3. This is a good starting template for the calculations below. You can also construct an input deck in Avogadro. Refer to the GAMESS manual for more information.

```
!   File created by the GAMESS Input Deck Generator Plugin for Avogadro
 $BASIS GBASIS=N31 NGAUSS=6 $END
 $CONTRL RUNTYP=ENERGY DFTTYP=SLATER $END

 $DATA
Title
C1
N     7.0    -1.03363     0.80618     0.00000
H     1.0    -0.01363     0.80618     0.00000
H     1.0    -1.37362     1.64340    -0.47314
H     1.0    -1.37363     0.79732     0.96162
 $END
```

1 GAMESS vs. FDA

Using `GAMESS`, perform a DFT/Hartree-Fock-Slater (`DFTTYP=SLATER`) calculation on an Ar atom using the 6-31G basis set.

(a) How many primitive Gaussians are included in this calculation? How many total basis functions? How do they divide between s, p, and d?

(b) How many SCF iterations does the calculation take to converge?

(c) What is the final calculated HFS/6-31G energy of the atom?

(d) What are the identities (1s, 2p, etc.) and energies of the occupied atomic orbitals?

(e) Compare your computed total energy and atomic orbital energies with those you got from Homework 2 using the fda code for Ar.

1.1 Solution

1.1.1 a)

This is an example input file for Ar. The resulting output files can be seen in ./Ar

```
 $BASIS GBASIS=N31 NGAUSS=6 $END
 $CONTRL RUNTYP=ENERGY DFTTYP=SLATER $END

 $DATA
Title
C1
Ar    18.0    -3.86612     1.03789     0.00000
 $END
```

By looking at the log file it seems we have 13 primitive Gaussians. Below is the shell command to produce this result.

```
1 grep "GAUSSIAN BASIS FUNCTIONS" Ar/Ar.log
```

```
NUMBER OF CARTESIAN GAUSSIAN BASIS FUNCTIONS =   13
```

We used the 6-31G basis set. The distribution of basis functions are listed below.

Orbital	Basis Sets	Primitive gauss functions	Total gauss functions
1s	1	6	6
2s	1	6	6
2p	3	6	18
3s	1	3	3
3p	3	3	9
3s+	1	1	1
3p+	3	1	3
Total	13		46

1.1.2 b)

12 SCF iterations are required to reach convergence.

```
1 grep "ITER" Ar/Ar.dat
```

```
E(R-SLATER)=     -524.4520526614, E(NUC)=    0.0000000000,   12 ITERS
```

1.1.3 c)

The final HFS/6-31G energy is -524.452 hartrees.

Table 1: `GAMESS` and `FDA` orbital energies

Orbital	E_{GAMESS} (hartree)	E_{FDA} (hartree)
1s	-113.6768	-116.9366
2s	-10.7172	-11.6037
2p	-8.3677	-9.2721
3s	-0.8218	-1.1022
3p	-0.3222	-0.5735
3s+	0.4316	
3p+	0.5206	

1.1.4 d)

1.1.5 e)

The `FDA` total energy is -526.8275 hartree, which means its a slightly better method for predicting the energy of Ar. The orbital energies are listed in table 1.

2 The Generalized Gradient Approximation

The generalized gradient approximation (GGA) is an improvement on Hartree-Fock-Slater that gives a nice balance between accuracy and computational expense. Using `GAMESS`, perform a single point calculation (`RUNTYP=ENERGY`) on the bent triatomic SO_2 using the GGA (`DFTTYP=PBE`) and PC1 basis set (`GBASIS=PC1`, `ISPHER=1`; no `NGAUSS` flag needed). Guess appropriate bond lengths and angle. Be sure to report your input file for your calculation.

(a) What is the spin multiplicity of SO_2? (Recall, the spin multiplicity is 2S +1, where S = 1/2 for one unpaired electron, S = 1 for two unpaired electrons, and so on).

(b) How many basis functions are in this calculation?

(c) How many SCF cycles does it take to converge?

(d) What SCF algorithm does the code use?

(e) What is the final total energy of the molecule?

(f) How many occupied orbitals does the molecule have? What are the energies of the HOMO and LUMO?

(g) What is the final dipole moment?

(h) What are the Mulliken gross charges on the S and O atoms?

(i) Plot out the electrostatic potential of SO_2. Which end of the molecule is electrophilic and which is nucleophilic?

2.1 Solution

The relevant files can be found at the github repository for the course.

INPUT File: ./SO2/SO2.inp

LOG File: ./SO2/SO2.log

DAT File: ./SO2/SO2.dat

2.1.1 a)

The spin multiplicity for SO_2 is 2 * 0 + 1 = 1.

2.1.2 b)

The number of basis functions in the calculation is 49.

```
grep "GAUSSIAN BASIS FUNCTIONS" ./SO2/SO2.log
```

```
NUMBER OF CARTESIAN GAUSSIAN BASIS FUNCTIONS =   49
```

2.1.3 c)

It takes 23 SCF cycles to converge.

```
grep "ITER" ./SO2/SO2.dat
```

```
E(R-PBE)=     -548.2499367382, E(NUC)=  109.8468077125,   23 ITERS
```

2.1.4 d)

The code uses the DIIS algorithm.

2.1.5 e)

The final total energy is -548.2499 hartrees

2.1.6 f)

There are 16 occupied orbitals.

```
grep "OCCUPIED" ./SO2/SO2.log
```

```
NUMBER OF OCCUPIED ORBITALS (ALPHA)          =   16
NUMBER OF OCCUPIED ORBITALS (BETA )          =   16
   16 ORBITALS ARE OCCUPIED (    7 CORE ORBITALS).
```

2.1.7 g)

The final dipole moment is 1.55 debyes

2.1.8 h)

The Mulliken charges are tabulated below.

ATOM	CHARGE
S	0.768910
O	-0.393816
O	-0.375094

2.1.9 i)

3 Geometry Optimization of SO_2

(a) Do a series of calculations in which you vary the S–O distances and O–S–O angle over a regular grid of values. Approximate the combination of values that give the lowest energy.

(b) A geometry optimization (`RUNTYP=OPTIMIZE`) is a faster way to find the optimal geometry of a molecule. Perform a geometry optimization on SO_2 using the same computational model as above. What are the optimal S–O distances and O–S–O angle?

3.1 Solution

3.1.1 a)

1. Creating input files and running `GAMESS`

   Here is some python code that creates and runs `GAMESS` calculations over a grid of distances and angles. Note that there are more efficient ways of doing these kind of calculations by using the CRC queueing system, which we will learn about in future classes.

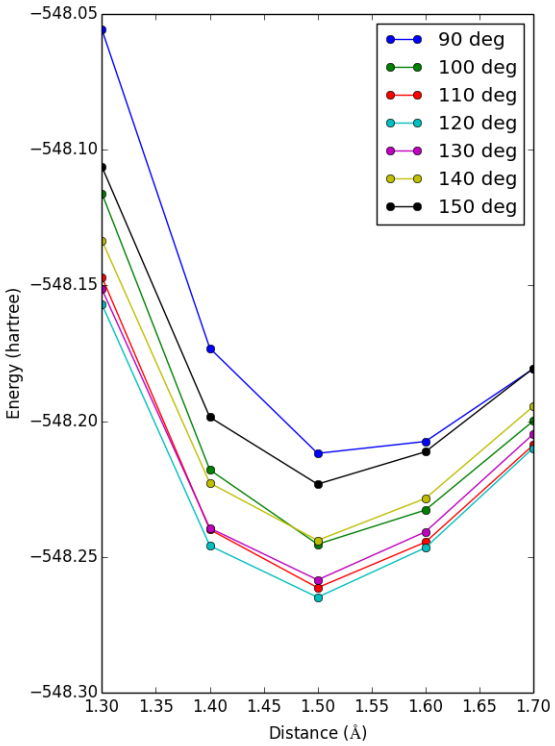
```
import os
import numpy as np

# Input file template
gamess_script=''' $BASIS GBASIS=PC1 $END
 $CONTRL COORD=ZMT ISPHER=1 RUNTYP=ENERGY DFTTYP=PBE $END

 $DATA
Title
Cnv 2

S
O 1 {0:1.2f}
O 1 {0:1.2f} 2 {1}
 $END'''

distances = np.linspace(1.3, 1.7, 5)
angles = np.linspace(90., 150., 7)
```

```
19
20  # Path to current directory
21  cwd = os.getcwd()
22
23  for distance in distances:
24      for angle in angles:
25
26          # Create calculation directory
27          wd = './SO2/optimize/{0:1.2f}-A/{1:1.2f}-deg/'.format(distance, angle)
28          if not os.path.exists(wd):
29              os.makedirs(wd)
30
31          # Change into calculation directory
32          os.chdir(wd)
33          # Create an input file
34          with open('gamess.inp', 'w') as f:
35              f.write(gamess_script.format(distance, angle))
36          # Run GAMESS
37          os.system('rungms gamess.inp > gamess.log')
38          # Change back to current working directory
39          os.chdir(cwd)
```

2. Finding the optimum distance and angle

   Now we will read the .dat files for the total energies and plot them. This is similar to what we did in Lab1.

```
1   import numpy as np
2   import matplotlib.pyplot as plt
3
4   distances = np.linspace(1.3, 1.7, 5)
5   angles = np.linspace(90., 150., 7)
6
7   plt.figure(figsize = (6,8))
8   for angle in angles:
9
10      energies = []
11      for distance in distances:
12
13          # Create calculation directory
14          filename = './SO2/optimize/{0:1.2f}-A/{1:1.2f}-deg/gamess.dat'.format(distance, angle)
15
16          with open(filename, 'r') as f:
17              for line in f.readlines():
18                  # We want lines like
19                  # E(R-PBE)=     -547.9256306493, E(NUC)=  130.1179768731,   20 ITERS
20                  if 'E(R-PBE)' in line:
21                      # Strip out the commas
22                      line = line.replace(',', "")
23                      # Now extract the energy
24                      energy = float(line.split()[1])
25                      energies.append(energy)
26                      # Break out of the loop as we got the energy
27                      break
28      plt.plot(distances, energies, 'o-', label = '{0:1.0f} deg'.format(angle))
29  plt.legend(loc='best')
30  plt.xlabel('Distance ($\AA$)')
31  plt.ylabel('Energy (hartree)')
32  plt.ticklabel_format(useOffset=False) # make the y-axis without offset
33  plt.tight_layout()
34  plt.savefig('SO2-optimization.png')
35  plt.show()
```



From the plot it looks like the lowest energy is roughly around a O-S-O angle of 120° and O-S distance of 1.5 Å. Note that we could have fit a curve to our data for a better approximation.

3.1.2 b)

Using `RUNTYP=OPTIMIZE` we have the S-O distance as 1.49 Å and the O-S-O angle as 118.35°. Looks like we came pretty close using our rough scan!

LOG File: `SO2/SO2-opt.log`

4 Other Molecules

Oxygen makes bonds with lots of things. Fill out the table below by doing an appropriate set of calculations:

4.1 Solution

AO_2	A-O (Å)	O-A-O (°)	Spin Multiplicity	Dipole Moment (eÅ)	Mulliken Charge
CO_2	1.176	180	1	0	C: 0.54, O: -0.27
NO_2	1.211	133.74	2	0.27	N: 0.38, O: -0.19
SiO_2	1.538	180	1	0	Si: 0.99, O: -0.50
SO_2	1.490	118.35	1	1.88	S: 0.82, O: -0.41