# Computational Chemistry Laboratory III (CBE 60547)

Prateek Mehta, Jerry Crum, William F. Schneider

10-29-2019 Thu

## 1 A review of what we know

So far we have learned how to:

- navigate the linux terminal
- create create and edit files using Emacs
- numerical analysis and plotting with Python
- different concepts in molecular simulations, e.g. potential energy surfaces, geometry optimizations, etc.

It might make sense to go back and read the lecture notes and notes from Lab 1 and Lab 2 if you feel the need to re-familiarize yourself with these things. In this lab, we will combine some of the things we learned and to perform DFT calculations with a powerful software package, `VASP`.

## 2 Required Software

### 2.1 Python

Acquired through Anaconda.

### 2.2 ASE

Provides packages for handling atoms and VASP calculations as objects in Python.

### 2.3 John Kitchin's vasp wrapper

Adds extra features to ASE, enhancing the usability of VASP in python.

### 2.4 bashrc, vasprc

Required run command files to assign default settings.

### 2.5 Guide

An in depth guide to installing all the required software on the CRC can be found here.

## 3 Simple SCF calculations

We will now perform a simple calculation on our CO molecule. This is done by creating a `vasp` calculator, which is an extension of the default Vasp calculator in ase (`ase.calculators.vasp`). The two properties that we will calculate in this example are the energy and the forces on the atoms.

The first time we run this code, a calculation will be submitted to the Notre Dame queue system. So when you try to print the potential energy of you will get an exception saying VaspSubmitted. You can check the status of the job by going back to the terminal and typing `qstat -u netid`. Once the job has finished running you can rerun the code, and if all went well, it should give you the energies and the forces.

```
from ase import Atoms, Atom
from ase.io import write
from ase.visualize import view
from vasp import Vasp
from vasp.vasprc import VASPRC

# These lines specify which queue to submit to, how many cores to request, and your parallel environment.
# They are not necessary if you want to use the default values.

VASPRC['queue.q'] = 'long'
VASPRC['queue.nprocs']= 8
VASPRC['queue.pe']= 'smp'

co = Atoms([Atom('C',[0,0,0]),
               Atom('O',[1.1,0,0])],
              cell=(10,10,10))

calc = Vasp('molecules/simple-co',
            xc = 'PBE',    # the exchange-correlation functional
            nbands = 8,    # number of bands
            encut = 350,   # planewave cutoff
            ismear = 1,    # methfessel-paxton smearing
            sigma = 0.01,  # very small smearing factor for a molecule
            atoms = co)

print('energy = {0} eV'.format(calc.get_potential_energy()))
print('Forces (eV/Ang.):')
print(co.get_forces())
print(calc) # Prints a summary of the calculation
            # Note: Some properties are attributes of the atoms object, and some of the calc.
```

```
energy = -14.69232797 eV
Forces (eV/Ang.):
[[-5.777  0.     0.   ]
 [ 5.777  0.     0.   ]]
SCF iterations = 16
: ------------------------------
  VASP calculation from /afs/crc.nd.edu/user/p/pmehta1/computational-chemistry/Lab3/molecules/s
  converged: True
```

```
  Energy = -14.692328 eV

  Unit cell vectors (angstroms)
        x       y     z      length
  a0 [ 10.000  0.000  0.000] 10.000
  a1 [ 0.000  10.000  0.000] 10.000
  a2 [ 0.000  0.000  10.000] 10.000
  a,b,c,alpha,beta,gamma (deg):10.000 10.000 10.000 90.0 90.0 90.0
  Unit cell volume = 1000.000 Ang^3
  Stress (GPa):xx,   yy,    zz,    yz,    xz,    xy
            -0.004  0.002  0.002-0.000 -0.000 -0.000
 Atom#  sym       position [x,y,z]tag  rmsForce constraints
   0    C   [0.000      0.000      0.000]  0   5.78      T T T
   1    O   [1.100      0.000      0.000]  0   5.78      T T T
-----------------------------------------------

INCAR Parameters:
-----------------
        nbands: 8
        ismear: 1
         encut: 350.0
         sigma: 0.01
        magmom: None
          kpts: [1, 1, 1]
    reciprocal: False
            xc: PBE
           txt: -
         gamma: False

Pseudopotentials used:
----------------------
C: potpaw_PBE/C/POTCAR (git-hash: ee4d8576584f8e9f32e90853a0cbf9d4a9297330)
O: potpaw_PBE/O/POTCAR (git-hash: 592f34096943a6f30db8749d13efca516d75ec55)
```

We can also look at the files created by `VASP` to see if everything went ok.

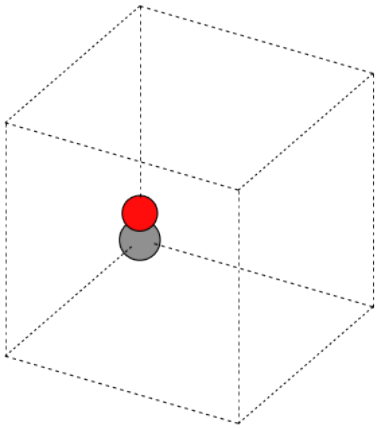
## 4 Geometry Optimizations

Now let us try to do a geometry optimization. For this `VASP` needs two additional keywords (at least) - `IBRION` and `NSW`. `IBRION` controls the relaxation algorithm and `NSW` specifies the total number of steps.

```
calc = Vasp('molecules/geometry-co',  # output dir relative to current dir
            xc='PBE',     # the exchange-correlation functional
            nbands=8,     # number of bands
            encut=350,    # planewave cutoff
            ismear=1,     # Methfessel-Paxton smearing
            sigma=0.01,   # very small smearing factor for a molecule
            nsw=20,       # Number of ionic steps
```

```
            ibrion=2,    # Conjugate gradient alogrithm
            atoms=co)

print('energy = {0} eV'.format(calc.get_potential_energy()))
print('number of geometry steps = {0}'.format(calc.get_number_of_ionic_steps()))
print('Forces (eV/Ang.):')
print(calc.get_forces())
print('Equilibrium Positions (Angs.):')
for atom in co:
    print(atom.symbol, atom.position)

# Save an image. Note that this is done outside the with statement
write('images/CO-relaxed.png', co, show_unit_cell=2, rotation='60x,-30y,90z')
```

```
energy = -14.81175954 eV
Forces (eV/Ang.):
[[ 0.003  0.     0.   ]
 [-0.003  0.     0.   ]]
Equilibrium Positions (Angs.):
C [-0.022  0.     0.   ]
O [ 1.122  0.     0.   ]
```



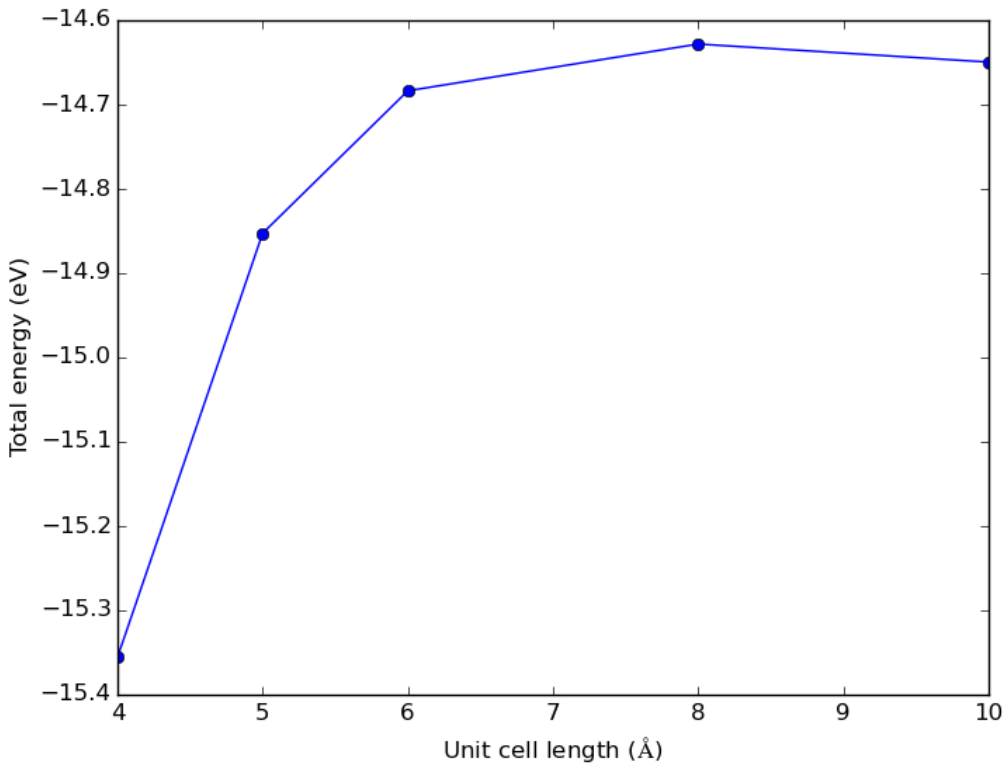
We might also want to visualize the relaxation trajectory. Using the terminal, change into the directory where you performed the calculation, and type in `jaspsum -t`.

## 5 Effect of Unit Cell Size

Let us consider a more complicated example. Here we will vary the size of the unit cell, to see how interactions between periodic images affect the energy.

```
from vasp import *
import numpy as np

L = [4,5,6,8,10]

energies = []

atoms = Atoms([Atom('C',[0,0,0]),
               Atom('O',[1.2,0,0])])


```

```
for a in L:
    atoms.set_cell([a,a,a], scale_atoms=False)
    atoms.center()

    calc = Vasp('molecules/co-L-{0}'.format(a),
                encut = 350,
                xc = 'PBE',
                atoms = atoms)
    energies.append(calc.get_potential_energy())

import matplotlib.pyplot as plt

plt.plot(L,energies, 'bo-')
plt.xlabel('Unit Cell Length ($\AA$)')
plt.ylabel('Total energy (eV)')
plt.savefig('images/co-e-v.png')
plt.ylim([-15.4,-14.6])
plt.show()
```




We can see that at small box sizes, there are attractive interactions between CO molecules that lower the total energy. At larger box sizes the energy starts to converge to a fixed value as the interactions are minimized. Now let's check the effect on the computational cost.

```
L = [4,5,6,8,10]

atoms = Atoms([Atom('C',[0,0,0]),
               Atom('O',[1.2,0,0])])

traj = []
for a in L:
    atoms.set_cell([a,a,a], scale_atoms=False)
```

```
    atoms.center()
    traj += [atoms]
    calc = Vasp('molecules/co-L-{0}'.format(a))
    print('{0} {1} seconds'.format(a,calc.get_elapsed_time()))

view(traj)
```

```
4 2.616 seconds
5 3.907 seconds
6 5.891 seconds
8 16.588 seconds
10 30.543 seconds
```

We can see the computational cost went up by a factor of 15! Perhaps you can now appreciate the computational cost involved in simulating 100s of atoms in large boxes!

# 6 Miscellaneous

## 6.1 Building pdfs from org files

Using the software you loaded at the beginning of lab, you should be able to build a pdf from your .org files. Let us try that, click on the Org menu and click Export/Publish. Then press 'l' and 'o'. This let's you build a pdf and open it.

Alternately, you can type, `C-c C-e l o`

## 6.2 Viewing latex equations in org documents

Click on `org-toggle-latex-overlays`. You should be able to see the Schrodinger equation below.

- $H\psi = E\psi$