Homework 5
Computational Chemistry
(CBE 60547)

Prof. William F. Schneider

Due: *<2015-03-19 Thu>*

1 Not Ar again!

As they say, there are many ways to skin a cat! You have computed the wavefunctions of Ar several different ways in homework already — using `FDA`, the descendent of Hartree’s first calculations, and using the molecular orbital `GAMESS` code — so it is only natural to do the same using `VASP`.

Create an atoms object for Ar centered in a 12 × 12 × 12 Å unit cell. Perform a `VASP` calculation using the PBE exchange correlation functional, Gaussian smearing, and a small sigma value (0.01). Answer the following questions.

(a) What are the total number of electrons in the calculation? (Hint: `calc.get_number_of_electrons()`)

(b) What is the potential energy of the Ar atom?

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

(d) Identify the orbitals. What are their occupancies and energies? Which orbitals are kept in the core? (Hint: `calc.get_eigenvalues()`, `calc.get_occupation_numbers()`)

(e) How many basis functions are in the calculation? (VASP doesn’t always print this for bigger systems, as such there is no in-built function in ase. Look in the OUTCAR file to figure this out).

1.1 Solution

```
from ase import Atoms, Atom
from ase.visualize import view
from jasp import *

Ar = Atoms([Atom(’Ar’, [0.0, 0.0, 0.0])],
           cell=(12,12,12))

Ar.center()

# This will use the default encut from the POTCAR
with jasp(’calcs/Ar’, # working directory
          xc=’PBE’, # xc functional
          ismear=0, # Gaussian smearing
          sigma=0.01, # very small smearing factor
          atoms=Ar) as calc:
    calc.calculate()
    print 'a) electrons in the calculation = {0}'.format(calc.get_number_of_electrons())
    print 'b) potential energy = {0}'.format(Ar.get_potential_energy())
    print 'c) No. of SCF iterations = {0}'.format(calc.get_number_of_iterations())
    orbital_energies = calc.get_eigenvalues()
    orbital_occupancies = calc.get_occupation_numbers()
    print '|orbital energy| orbital occ.|'
    for e, occ in zip(orbital_energies, orbital_occupancies):
        print '| {0:1.2f}|{1}|'.format(e, occ)
```

(a) electrons in the calculation = 8.0

(b) potential energy = -0.03061587

(c) No. of SCF iterations = 8

orbital energy	orbital occ.
-24.18	2.0
-10.16	2.0
-10.16	2.0
-10.16	2.0
-0.32	0.0
0.63	0.0
0.88	0.0
0.88	0.0

(d) The first orbital in the table above the 3s orbital and the three degenerate orbitals are the 3p orbitals. The 1s, 2s, 2p orbitals are kept in the core.

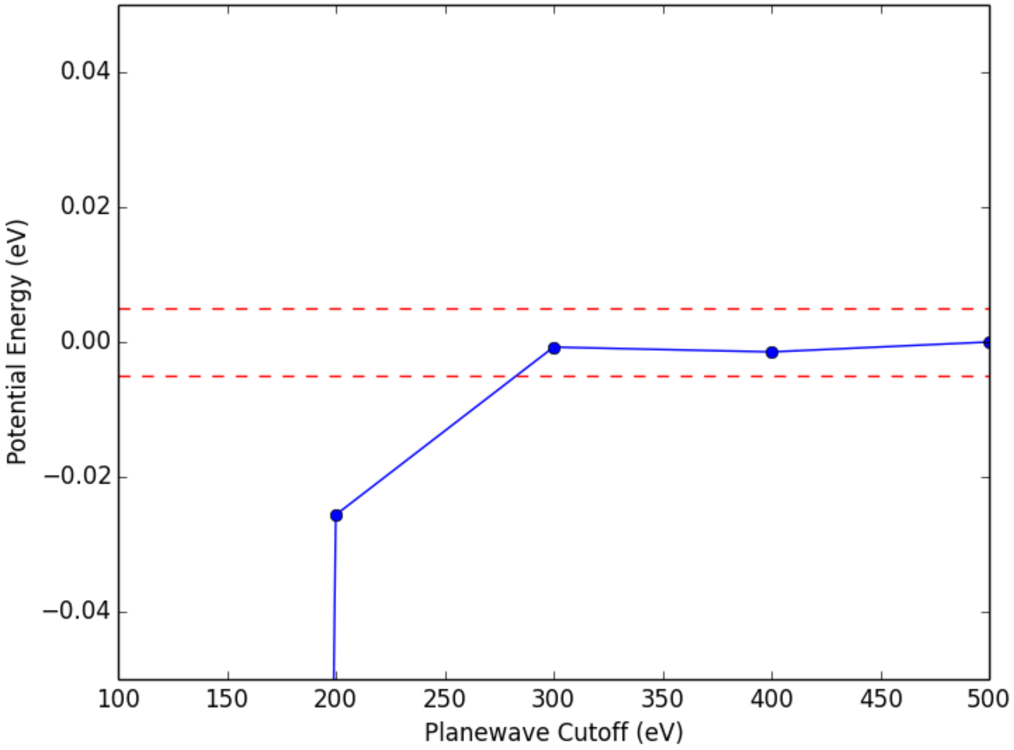
(e) The calculation uses 110592 plane-waves.

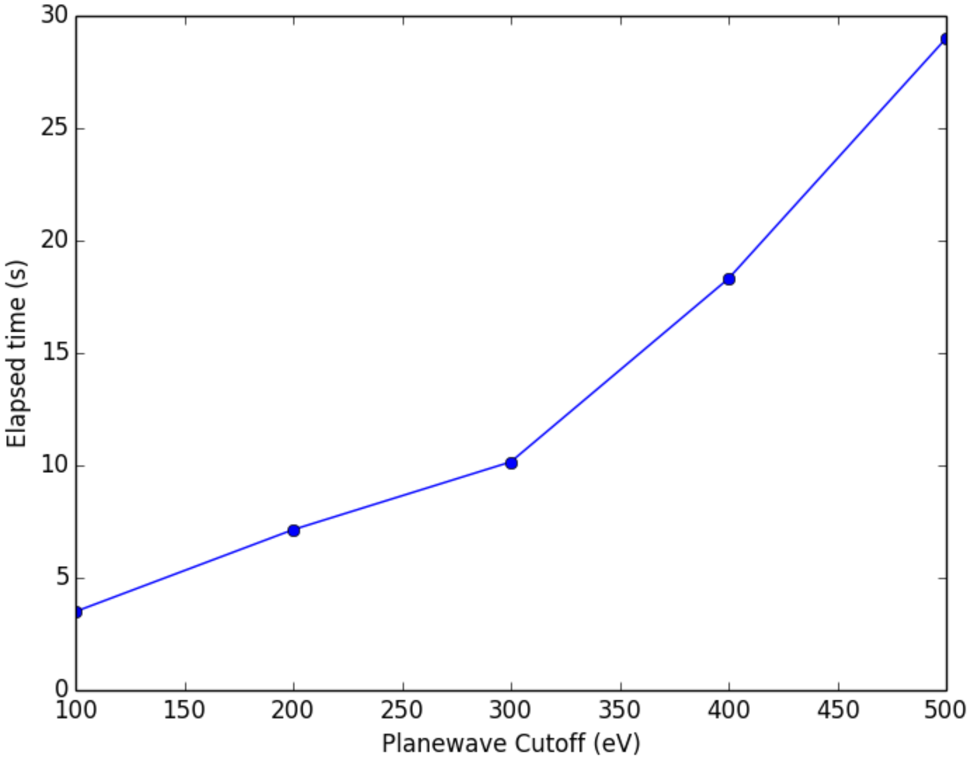
```
grep 'total plane-waves' ./calcs/Ar/OUTCAR
```

```
total plane-waves  NPLWV = 110592
```

2 Convergence of Encut

```
from ase import Atoms, Atom
from ase.visualize import view
from jasp import *
import matplotlib.pyplot as plt

Ar = Atoms([Atom('Ar', [0.0, 0.0, 0.0])],
           cell=(12,12,12))
Ar.center()

ready = True
encuts = [100, 200, 300, 400, 500] # eV
energies = [] #eV
times = []
for encut in encuts:
    with jasp('calcs/encuts/Ar-{0}'.format(encut), # working directory
              xc='PBE', # xc functional
              encut=encut, # Planewave cutoff
              ismear=0, # Gaussian smearing
              sigma=0.01, # very small smearing factor
              atoms=Ar) as calc:
        try:
            energies.append(Ar.get_potential_energy())
            times.append(calc.get_elapsed_time())
        except(VaspSubmitted, VaspQueued):
            ready = False

if not ready:
    import sys; sys.exit()

energies = np.array(energies)
energies -= energies[-1]

plt.figure()
plt.plot(encuts, energies, ’o-’)
plt.axhline(-0.005, ls =’--’, c=’r’)
plt.axhline(0.005, ls=’--’, c=’r’)
plt.ylim(-0.05, 0.05)
plt.xlabel(’Planewave Cutoff (eV)’)
plt.ylabel(’Potential Energy (eV)’)
plt.savefig(’encuts.png’)

plt.figure()
plt.plot(encuts, times, ’o-’)
plt.xlabel(’Planewave Cutoff (eV)’)
plt.ylabel(’Elapsed time (s)’)
plt.savefig(’times.png’)
plt.show()
```





The energy seems to be converged at an encut of 300 (eV). A dramatic increase in computational time is seen on increasing the energy cutoff.

3 Spin polarized Oxygen

```
from ase import Atoms, Atom
from ase.visualize import view
from jasp import *

O = Atoms([Atom(’O’, [0.0, 0.0, 0.0])],
           cell=(12,12,12))
O.center()

with jasp(’calcs/O’, # working directory
          xc=’PBE’, # xc functional
          encut=400, # planewave cutoff
          ismear=0, # Gaussian smearing
          sigma=0.01, # Very small smearing factor
          ispin=2, # Spin polarized calculation
          atoms=O) as calc:
    calc.calculate()
    print ’a) O potential energy = {0:1.3f} eV’.format(O.get_potential_energy())
    print ’b) Spin up - Spin down electrons = {0:1.2f}’.format(O.get_magnetic_moment())
    print ’c) Spin up orbitals’

    up_energies = calc.get_eigenvalues(spin=0)
    up_occupancies = calc.get_occupation_numbers(spin=0)
    print ’|orbital energy| orbital occ.|’
    for e, occ in zip(up_energies, up_occupancies):
        print '| {0:1.2f}|{1}|'.format(e, occ)

    print '\n Spin down orbitals'
    down_energies = calc.get_eigenvalues(spin=1)
    down_occupancies = calc.get_occupation_numbers(spin=1)
    print '|orbital energy| orbital occ.|'
    for e, occ in zip(down_energies, down_occupancies):
        print '| {0:1.2f}|{1}|'.format(e, occ)
```

(a) O potential energy = -1.525 eV

(b) Spin up - Spin down electrons = 2.00

(c) Spin up orbitals

orbital energy	orbital occ.
-25.19	1.0
-10.19	1.0
-10.19	1.0
-10.19	1.0
-0.28	0.0
0.51	0.0
0.88	0.0
0.88	0.0

Spin down orbitals

orbital energy	orbital occ.
-21.95	1.0
-7.16	0.33333
-7.16	0.33333
-7.16	0.33333
-0.20	0.0
0.70	0.0
0.92	0.0
0.92	0.0

The oxygen atom has two unpaired electrons, which is what we see in (b). It appears that in the spin-down orbitals the electons are smeared across the degenerate p-down orbitals. This is a feature of the symmetry of the problem. To converge to a lower symmetry solution, we need to break the symmetry of the box and probably explicitly specify the inital occupancies.

4 Geometry Optimizations

```
from ase import Atoms, Atom
from ase.visualize import view
from jasp import *
import numpy as np

O2 = Atoms([Atom('O', [0.0, 0.0, 0.0]),
            Atom('O', [1.5, 0.0, 0.0])],
           cell=(12,12,12))
O2.center()

```

```
with jasp('calcs/O2-geometry', # working directory
          xc='PBE', # xc functional
          encut=400, # planewave cutoff
          ismear=0, # Gaussian smearing
          sigma=0.01, # Very small smearing factor
          ispin=2, # Spin polarized calculation
          ibrion=1, # Quasi Newton minimization
          nsw=20, # optimization steps
          atoms=O2) as calc:

    calc.calculate()
    print 'a) Convergence = {0}'.format(calc.read_convergence())
    print 'b) relaxation steps = {0}'.format(calc.get_number_of_ionic_steps())
    print 'c) Final energy = {0:1.3f} eV'.format(O2.get_potential_energy())
    print 'd) Magneric Moment = {0:1.2f}'.format(O2.get_magnetic_moment())

    up_energies = calc.get_eigenvalues(spin=0)
    up_occupancies = calc.get_occupation_numbers(spin=0)
    print '|orbital energy| orbital occ.|'
    for e, occ in zip(up_energies, up_occupancies):
        print '| {0:1.2f}|{1}|'.format(e, occ)

    print '\n Spin down orbitals'
    down_energies = calc.get_eigenvalues(spin=1)
    down_occupancies = calc.get_occupation_numbers(spin=1)
    print '|orbital energy| orbital occ.|'
    for e, occ in zip(down_energies, down_occupancies):
        print '| {0:1.2f}|{1}|'.format(e, occ)
```

(a) Convergence = True

(b) relaxation steps = 16

(c) Final energy = -9.865 eV

(d) Magneric Moment = 2.00

orbital energy	orbital occ.
-32.52	1.0
-20.66	1.0
-13.42	1.0
-13.27	1.0
-13.27	1.0
-6.95	1.0
-6.95	1.0
-0.30	0.0
0.57	0.0
0.65	0.0
0.76	0.0
0.76	0.0
0.91	0.0
0.95	0.0
1.44	0.0
1.72	0.0

Spin down orbitals

orbital energy	orbital occ.
-31.30	1.0
-18.86	1.0
-12.49	1.0
-11.42	1.0
-11.42	1.0
-4.68	0.0
-4.68	0.0
-0.20	0.0
0.72	0.0
0.77	0.0
0.85	0.0
0.85	0.0
0.94	0.0
0.98	0.0
1.60	0.0
1.78	0.0

The magnetic moment and the occupancies make sense, because the triplet oxygen is the stable state for diatomic oxygen.

5 Vibrational Frequencies/ZPE

```
from ase import Atoms, Atom
from ase.visualize import view
from jasp import *
import numpy as np

c = 3e10 # speed of light cm/s
h = 4.135667516e-15 # eV/s

O2 = Atoms([Atom(’O’, [0.0, 0.0, 0.0]),
            Atom(’O’, [1.234, 0.0, 0.0])],
           cell=(12,12,12))
O2.center()

with jasp(’calcs/O2-frequency’, # working directory
          xc=’PBE’, # xc functional
          encut=400, # planewave cutoff
          ismear=0, # Gaussian smearing
          sigma=0.01, # Very small smearing factor
          ispin=2, # Spin polarized calculation
          ibrion=6, # finite-differences frequency
          nfree=2,
          nsw=20,
          potim=0.01, # step size Angs
          atoms=O2) as calc:
    calc.calculate()
    O2_energy = O2.get_potential_energy()
    freq = calc.get_vibrational_frequencies() #cm^(-1)

print ’Calculated frequencies = {0} cm^(-1)’.format(freq)
# only the first frequency is meaningful
O2_freq = freq[0]
print ’a) Oxygen frequency = {0:1.3f} cm^(-1)’.format(O2_freq)

ZPE = 0.5*h*c*O2_freq
```

```python
print 'b) The ZPE of O_{{2}} is {0:1.3f} eV'.format(ZPE)

# O atom energy from last calc
O_energy = -1.525 #eV

print 'c) ZPE corrected bond-dissociation energy = {0:1.3f} eV'.format(2*O_energy - O2_energy - ZPE)
```

```
Calculated frequencies = [1566.985581, 96.476627, 87.667451, 1.4e-05, (9e-06+0j), (2.7e-05+0j)]
a) Oxygen frequency = 1566.986 cm^(-1)
b) The ZPE of O_{2} is 0.097 eV
c) ZPE corrected bond-dissociation energy = 6.718 eV
```

This the ZPE corrected bond-dissociation energy predicted by DFT is significatly higher than the experimentally calculated value of 5.15 eV (http://en.wikipedia.org/wiki/Bond-dissociation_energy). This can be due to several reasons. We did not consider the low-symmetry configuration of the O atom, or if 400 eV is a converged plane-wave cutoff energy. Moreover, GGA exchange-correlation functionals are known to predict overbinding.