Molecular Simulation HOMEWORK 4

Zhongnan Xu

10/24/12 - Wednesday

Contents

1 Convergence study of tantalum

1.1 Planewave cutoff convergence

Determine the planewave cutoff energy required to achieve a total energy convergence of 10 meV for bcc tantalum at its experimental lattice constant. Use a k-point grid of $10 \times 10 \times 10$, and the PBE exchange correlation functional.

```python
from ase import Atom, Atoms
from ase.visualize import view
import numpy as np
import matplotlib.pyplot as plt
from jasp import *
ready=True
a = 3.31 # Taken from Lange's Handbook of Chemistry
a1 = np.array((-a/2, a/2, a/2))
a2 = np.array((a/2, -a/2, a/2))
a3 = np.array((a/2, a/2, -a/2))
```

```
Ta = Atoms([Atom('Ta', (0, 0, 0), magmom=6)],
           cell=(a1, a2, a3))
encuts = (150, 200, 250, 300, 350, 400, 450, 500)
energies = []
for cut in encuts:
    with jasp('1.1/e{0:d}'.format(cut),
              xc='PBE', lreal=False, istart=0,
              encut=cut, prec='Accurate',
              kpts=(10, 10, 10), ismear=1, sigma=0.05,
              ispin=2, lorbit=11,
              atoms=Ta) as calc:
        try:
            energies.append(Ta.get_potential_energy())
        except (VaspSubmitted, VaspQueued):
            energies.append(None)
            ready = False
            pass
assert len(encuts) == len(energies)
if not ready:
    import sys; sys.exit()

import matplotlib.pyplot as plt
from matplotlib.ticker import ScalarFormatter
# First offset the energies by the last value and find the absolute value
energies = np.array(energies)
energies -= energies[-1]*np.ones(len(energies))
energies = np.absolute(energies)
energies = energies*1000
fig = plt.figure(1, (5.5, 4.5))
ax = fig.add_subplot(111)
ax.plot(encuts, energies, label='Ta', marker='o')
ax.axhline(10, ls='--', label='10 meV Convergence')
ax.set_xlabel('Kinetic Energy Cutoff (eV)', size='large')
ax.set_ylabel('Total Energy (meV/atom)', size='large')
ax.legend(loc=0, prop={'size':'large'})
ax.yaxis.set_major_formatter(ScalarFormatter(useOffset=False))
fig.tight_layout()
plt.savefig('1-1.png')
plt.show()
```

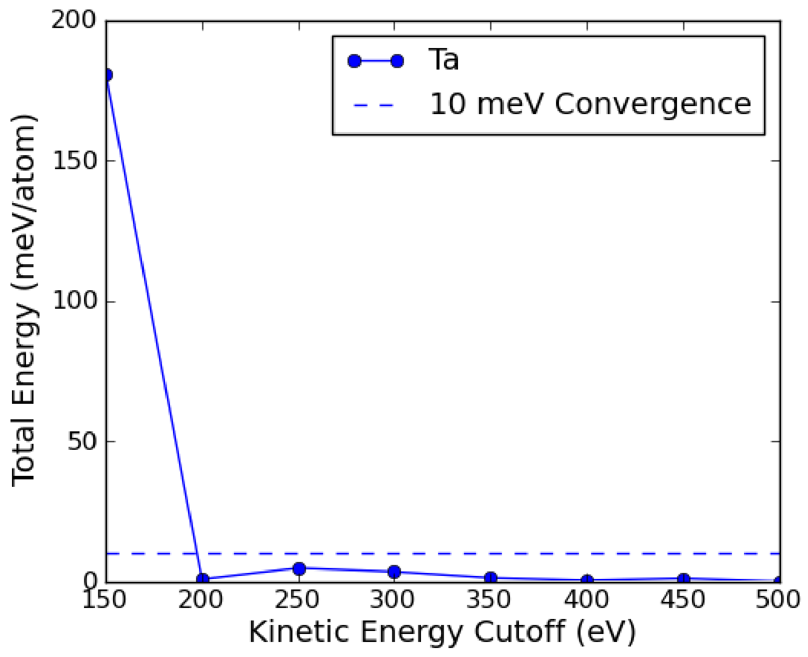

Figure 1: Convergence of the total energy of BCC Ta at the experimental lattice cutoff

It is clear that the total energy of BCC Ta has already been converged at an energy cutoff of 200 eV. This should not be surprising considering the VASP site recommends a minimum cutoff of 220 eV.

1.2 k-point grid convergence

Determine the Monkhorst-Pack k-point grid required to achieve a total energy convergence of 50 meV for bcc tantalum at its experimental lattice constant. Use a planewave cutoff of 350 eV for this study.

```python
from ase import Atom, Atoms
from ase.visualize import view
from ase.lattice.cubic import BodyCenteredCubic
import numpy as np
import matplotlib.pyplot as plt
from jasp import *

# We will rely on the ase.lattice.cubic.BodyCenteredCubic module to give us
# the correct experimental lattice constant. Note we use the primitive cell
ready=True
a = 3.31
a1 = np.array((-a/2, a/2, a/2))
a2 = np.array((a/2, -a/2, a/2))
a3 = np.array((a/2, a/2, -a/2))
Ta = Atoms([Atom('Ta', (0, 0, 0), magmom=6)],
           cell=(a1, a2, a3))

ks = (2, 4, 6, 8, 10, 12, 14, 16)
kpoints = []
for k in ks:
    kpoints.append((k, k, k))
```

```python
energies = []
for kpoint in kpoints:
    with jasp('1.2/k{0:d}'.format(kpoint[0]),
              xc='PBE', lreal=False, istart=0,
              encut=350, prec='Accurate',
              kpts=kpoint, ismear=1, sigma=0.05,
              ispin=2, lorbit=11,
              atoms=Ta) as calc:
        try:
            energies.append(Ta.get_potential_energy())
        except (VaspSubmitted, VaspQueued):
            energies.append(None)
            ready = False
            pass
assert len(ks) == len(energies)
if not ready:
    import sys; sys.exit()

import matplotlib.pyplot as plt
from matplotlib.ticker import ScalarFormatter
# First offset the energies by the last value and find the absolute value
energies = np.array(energies)
energies -= energies[-1]*np.ones(len(energies))
energies = np.absolute(energies)
energies = energies*1000
fig = plt.figure(1, (5.5, 4.5))
ax = fig.add_subplot(111)
ax.plot(ks, energies, label='Ta', marker='o')
ax.axhline(50, ls='--', label='50 meV convergence')
ax.set_ylim((0, 100))
ax.set_xlabel(r'Kpoint Grid (N$\times$ N$\times$ N)', size='large')
ax.set_ylabel('Total Energy (meV/atom)', size='large')
ax.legend(loc=0, prop={'size':'large'})
ax.yaxis.set_major_formatter(ScalarFormatter(useOffset=False))
fig.tight_layout()
plt.savefig('1-2.png')
plt.show()
```

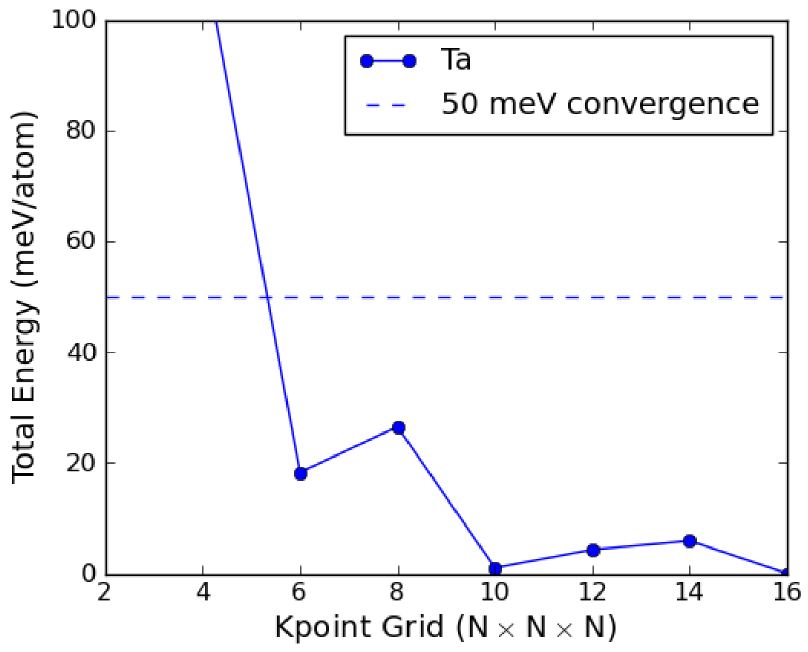

Figure 2: Convergence of BCC Ta with respect to increasing k-point grid

BCC Ta is converged within 50 meV by a k-point grid size of 6 × 6 × 6.

2 Convergence study of graphite

2.1 Planewave convergence

Determine the planewave cutoff energy required to get convergence of 10 meV or better for graphite at its equilibrium lattice constant. Use a k-point grid of (6,6,6) for this study.

```python
from jasp import *
from ase.lattice.hexagonal import Graphite
from ase.visualize import view

ready = True

atoms = Graphite('C', latticeconstant={'a':2.4612, 'c':6.7079})
encuts = (250, 300, 350, 400, 450, 500)
energies = []
for cut in encuts:
    with jasp('2.1/e{0:d}'.format(cut),
              xc='PBE', lreal=False, istart=0,
              encut=cut, prec='Accurate',
              kpts=(6, 6, 6), ismear=1, sigma=0.05, gamma=True,
              ispin=2, lorbit=11,
              atoms=atoms) as calc:
        try:
            energies.append(atoms.get_potential_energy())
        except (VaspSubmitted, VaspQueued):
            energies.append(None)
            ready = False
            pass
```

```python
assert len(encuts) == len(energies)
if not ready:
    import sys; sys.exit()

import matplotlib.pyplot as plt
from matplotlib.ticker import ScalarFormatter
# First offset the energies by the last value and find the absolute value
energies = np.array(energies)
energies -= energies[-1]*np.ones(len(energies))
energies = np.absolute(energies)
energies = energies*1000
fig = plt.figure(1, (5.5, 4.5))
ax = fig.add_subplot(111)
ax.plot(encuts, energies, label='Graphite', marker='o')
ax.axhline(10, ls='--', label='10 meV Convergence')
ax.set_xlabel('Kinetic Energy Cutoff (eV)', size='large')
ax.set_ylabel('Total Energy (meV/atom)', size='large')
ax.legend(loc=0, prop={'size':'large'})
ax.yaxis.set_major_formatter(ScalarFormatter(useOffset=False))
fig.tight_layout()
plt.savefig('2-1.png')
plt.show()
```

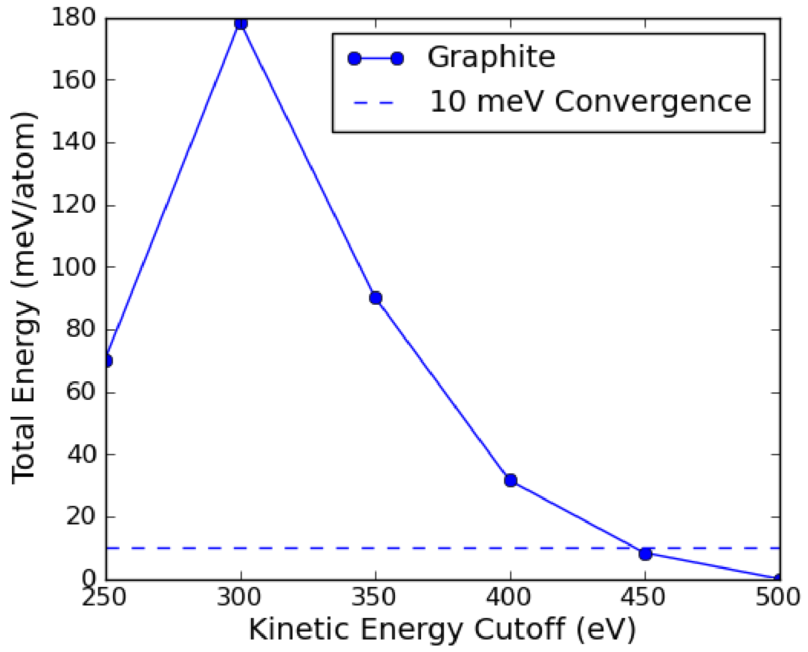

Figure 3: Convergence of graphite with respect to increasing kinetic energy cutoff

Graphite converges within 450 eV plane wave cutoff energy.

2.2 k-point convergence

Determine the k-point grid required to get convergence of 50 meV or better for graphite at its equilibrium lattice constant. Use a planewave cutoff of 350 eV for this study.

Note, I used a gamma point grid spacing because we are using a hexagonal cell. The VASP site recommends the use of gamma point spacing for hexagonal unit cells.

```python
from jasp import *
from ase.lattice.hexagonal import Graphite

# We will rely on the ase.lattice.cubic.BodyCenteredCubic module to give us
# the correct experimental lattice constant. Note we use the primitive cell
ready=True

atoms = Graphite('C', latticeconstant={'a':2.4612, 'c':6.7079})
ks = (2, 4, 6, 8, 10, 12, 14, 16)
kpoints = []
for k in ks:
    kpoints.append((k, k, k))
energies = []
for kpoint in kpoints:
    with jasp('2.2/k{0:d}'.format(kpoint[0]),
              xc='PBE', lreal=False,
              encut=350, prec='Accurate',
              kpts=kpoint, ismear=1, sigma=0.05, gamma=True,
              ispin=2, lorbit=11,
              atoms=atoms) as calc:
        try:
            energies.append(atoms.get_potential_energy())
        except (VaspSubmitted, VaspQueued):
            energies.append(None)
            ready = False
            pass
assert len(ks) == len(energies)
if not ready:
    import sys; sys.exit()

import matplotlib.pyplot as plt
from matplotlib.ticker import ScalarFormatter
# First offset the energies by the last value and find the absolute value
energies = np.array(energies)
energies -= energies[-1]*np.ones(len(energies))
energies = np.absolute(energies)
energies = energies*1000
fig = plt.figure(1, (5.5, 4.5))
ax = fig.add_subplot(111)
ax.plot(ks, energies, label='Graphite', marker='o')
ax.axhline(50, ls='--', label='50 meV convergence')
ax.set_ylim((0, 100))
ax.set_xlabel(r'Kpoint Grid (N$\times$ N$\times$ N)', size='large')
ax.set_ylabel('Total Energy (meV/atom)', size='large')
ax.legend(loc=0, prop={'size':'large'})
ax.yaxis.set_major_formatter(ScalarFormatter(useOffset=False))
fig.tight_layout()
plt.savefig('2-2.png')
plt.show()
```

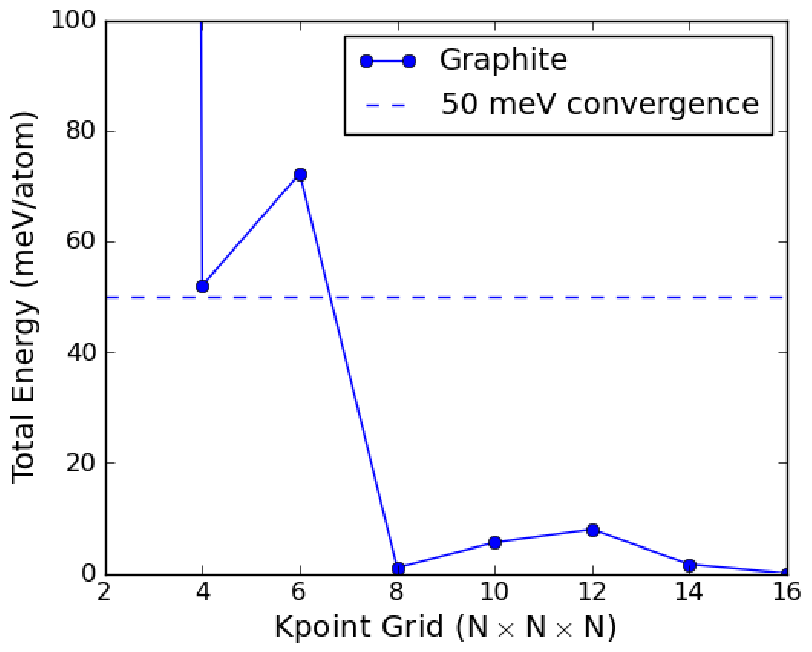

Figure 4: Convergence of Graphite with respect to increasing k-point grid

Graphite converges within an 8 × 8 × 8 kpoint gamma point grid spacing.

3 Determine the DFT lattice constant of bcc tantalum

Use the parameters you estimated in the previous problem. Construct an equation of state and determine the lattice constant and bulk modulus of tantalum. Compare your answers to literature values, and cite the source of your comparison.

```python
from jasp import *
from ase.lattice.cubic import BodyCenteredCubic
import numpy as np
import matplotlib.pyplot as plt

ready=True
lat = 3.31 # Taken from Lange's Handbook of Chemistry
space = 0.02
lats = (lat - 2*space,
        lat - space,
        lat,
        lat + space,
        lat + 2*space)
energies, volumes  = [], []
for a in lats:
    a1 = np.array((-a/2, a/2, a/2))
    a2 = np.array((a/2, -a/2, a/2))
    a3 = np.array((a/2, a/2, -a/2))
    Ta = Atoms([Atom('Ta', (0, 0, 0), magmom=6)],
               cell=(a1, a2, a3))
    with jasp('3.1/a-{0:1.2f}'.format(a),
              xc='PBE', lreal=False, istart=0,
              encut=200, prec='Accurate',
              kpts=(6, 6, 6), ismear=1, sigma=0.05,
              ispin=2, lorbit=11,
              atoms=Ta) as calc:
        try:
            energies.append(Ta.get_potential_energy())
            volumes.append(Ta.get_volume())
        except (VaspSubmitted, VaspQueued):
            energies.append(None)
            ready = False
            pass
if not ready:
    import sys; sys.exit()

from ase.utils.eos import EquationOfState
eos = EquationOfState(volumes, energies)
v0, e0, B = eos.fit()
eqlat = (2*v0) ** (1./3.)
print 'The equilibrium lattice constant is {0:1.3f} Angstroms'.format(eqlat)
print 'The bulk modulus is {0:1.3f} eV/Angstroms^3'.format(B)
eos.plot('3-1.png')
```

```
The equilibrium lattice constant is 3.305 Angstroms
The bulk modulus is 1.323 eV/Angstroms^3
```

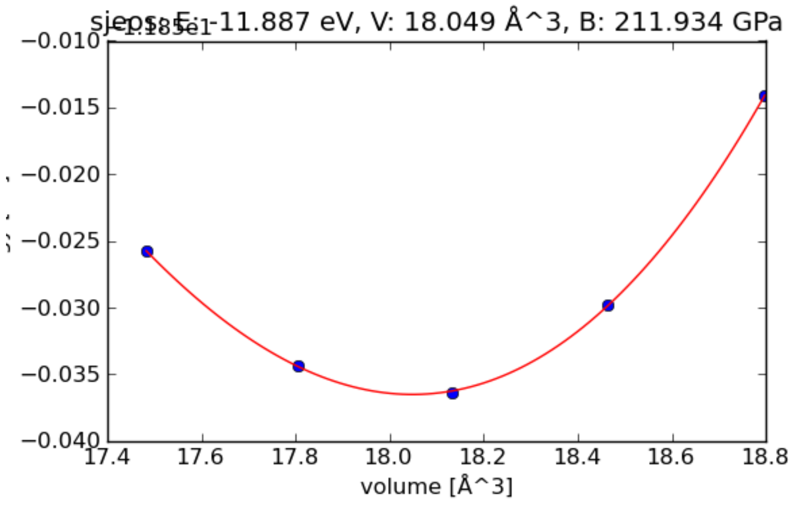

Figure 5: Convergence of Graphite with respect to increasing k-point grid

4 Determine the DFT lattice constant of fcc tantalum

Construct an equation of state to determine the lattice constant of fcc tantalum. You can assume the same parameters that were good for the bcc structure are good for the fcc structure. Which structure is more stable, fcc or bcc?

First we have to calculate the lattice constant of an FCC structure that corresponds to around 3.31 Å lattice constant of the BCC structure

The volume of the FCC primitive cell in terms of the lattice constant is $V_{\text{fcc}} = \frac{a_{\text{fcc}}^3}{4}$ while that of the BCC primitive cell is $V_{\text{bcc}} = \frac{a_{\text{bcc}}^3}{2}$. Setting these two equal, we see that $a_{\text{fcc}} = (2a_{\text{bcc}}^3)^{1/3}$.

```python
a_fcc = (2*3.31**3)**(1./3.)
print 'A good guess for the FCC lattice constant is {0:1.3f}'.format(a_fcc)
```

```
A good guess for the FCC lattice constant is 4.170
```

Now we can compute the EOS for FCC Ta.

```python
from jasp import *
import numpy as np
import matplotlib.pyplot as plt

ready=True
lat = 4.170
space = 0.02
lats = (lat - 2*space,
        lat - space,
        lat,
        lat + space,
        lat + 2*space,
        lat + 3*space,
        lat + 4*space,
        lat + 5*space)
energies, volumes  = [], []
for a in lats:
    a1 = np.array((0, a/2, a/2))
    a2 = np.array((a/2, 0, a/2))
    a3 = np.array((a/2, a/2, 0))
    Ta = Atoms([Atom('Ta', (0, 0, 0), magmom=6)],
               cell=(a1, a2, a3))
    with jasp('4.1/a-{0:1.2f}'.format(a),
              xc='PBE', lreal=False, istart=0,
              encut=200, prec='Accurate',
              kpts=(6, 6, 6), ismear=1, sigma=0.05,
              ispin=2, lorbit=11,
              atoms=Ta) as calc:
        try:
            energies.append(Ta.get_potential_energy())
            volumes.append(Ta.get_volume())
        except (VaspSubmitted, VaspQueued):
            energies.append(None)
            ready = False
            pass
if not ready:
    import sys; sys.exit()

from ase.utils.eos import EquationOfState
eos = EquationOfState(volumes, energies)
v0, e0, B = eos.fit()
eqlat = (4*v0) ** (1./3.)
print 'The equilibrium lattice constant is {0:1.3f} Angstroms'.format(eqlat)
print 'The bulk modulus is {0:1.3f} eV/Angstroms^3'.format(B)
eos.plot('4-1.png')
```

```
The equilibrium lattice constant is 4.214 Angstroms
The bulk modulus is 1.222 eV/Angstroms^3
```

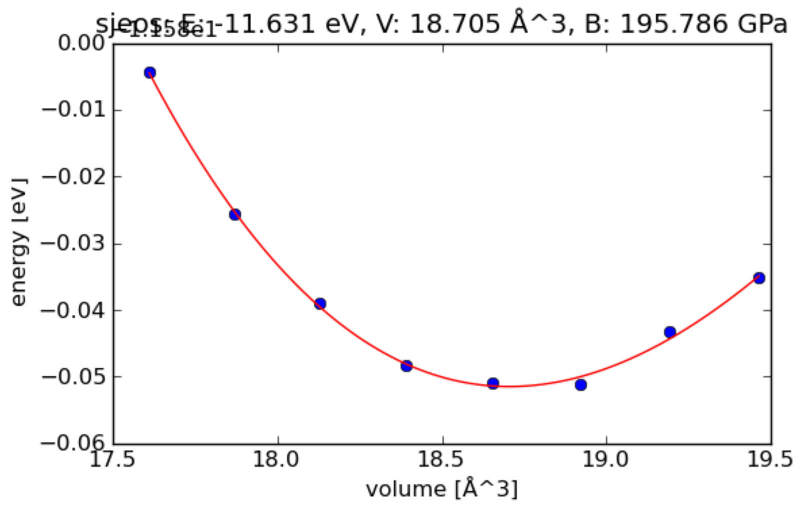

Figure 6: Equation of state for FCC tantalum

According the equation of state, we have a minimum at 4.214 Å. However, the equation of state clearly shows a volume with a lower energy than the minimum. Furthermore, the points are not smoothly fitted to a line, which suggests that the energies are not converged. From my experience, increasing the k-point spacing or kinetic energy cutoff will give us a smoother EOS. Lets do both.

```python
from jasp import *
import numpy as np
import matplotlib.pyplot as plt

ready=True
lat = 4.170
space = 0.02
lats = (lat - 2*space,
        lat - space,
        lat,
        lat + space,
        lat + 2*space,
        lat + 3*space,
        lat + 4*space,
        lat + 5*space)
energies, volumes  = [], []
for a in lats:
    a1 = np.array((0, a/2, a/2))
    a2 = np.array((a/2, 0, a/2))
    a3 = np.array((a/2, a/2, 0))
    Ta = Atoms([Atom('Ta', (0, 0, 0), magmom=6)],
               cell=(a1, a2, a3))
    with jasp('4.2/a-{0:1.2f}'.format(a),
              xc='PBE', lreal=False, istart=0,
              encut=400, prec='Accurate',
              kpts=(8, 8, 8), ismear=1, sigma=0.05,
              ispin=2, lorbit=11,
              atoms=Ta) as calc:
        try:
            energies.append(Ta.get_potential_energy())
            volumes.append(Ta.get_volume())
        except (VaspSubmitted, VaspQueued):
            energies.append(None)
            ready = False
            pass
if not ready:
    import sys; sys.exit()

from ase.utils.eos import EquationOfState
eos = EquationOfState(volumes, energies)
v0, e0, B = eos.fit()
eqlat = (4*v0) ** (1./3.)
print 'The equilibrium lattice constant is {0:1.3f} Angstroms'.format(eqlat)
print 'The bulk modulus is {0:1.3f} eV/Angstroms^3'.format(B)
eos.plot('4-2.png')
```

```
The equilibrium lattice constant is 4.208 Angstroms
The bulk modulus is 1.254 eV/Angstroms^3
```

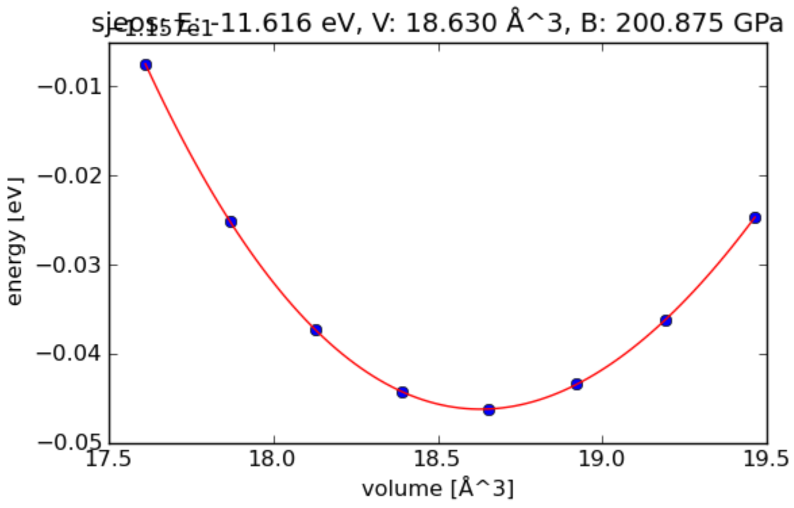

Figure 7: Convergence of Graphite with respect to increasing k-point grid

We see that the higher parameters smoothed out the EOS and gives us an equilibriuim lattice constant of 4.208 Å for Ta FCC. Now lets plot both graphs and look at their relative energetics.

```
from jasp import *
import numpy as np
import matplotlib.pyplot as plt

lat = 3.31
space = 0.02
lats = (lat - 2*space,
        lat - space,
        lat,
        lat + space,
        lat + 2*space)
bcc_energies, bcc_volumes  = [], []
for a in lats:
    a1 = np.array((-a/2, a/2, a/2))
    a2 = np.array((a/2, -a/2, a/2))
    a3 = np.array((a/2, a/2, -a/2))
    Ta = Atoms([Atom('Ta', (0, 0, 0), magmom=6)],
               cell=(a1, a2, a3))
    with jasp('3.1/a-{0:1.2f}'.format(a), atoms=Ta) as calc:
        bcc_energies.append(Ta.get_potential_energy())
        bcc_volumes.append(Ta.get_volume())

lat = 4.170
space = 0.02
lats = (lat - 2*space,
        lat - space,
        lat,
        lat + space,
        lat + 2*space,
        lat + 3*space,
        lat + 4*space,
        lat + 5*space)
fcc_energies, fcc_volumes  = [], []
for a in lats:
    a1 = np.array((0, a/2, a/2))
    a2 = np.array((a/2, 0, a/2))
    a3 = np.array((a/2, a/2, 0))
    Ta = Atoms([Atom('Ta', (0, 0, 0), magmom=6)],
               cell=(a1, a2, a3))
    with jasp('4.2/a-{0:1.2f}'.format(a), atoms=Ta) as calc:
        fcc_energies.append(Ta.get_potential_energy())
        fcc_volumes.append(Ta.get_volume())

fig = plt.figure(1, (5.5, 4.5))
ax = fig.add_subplot(111)
ax.plot(bcc_volumes, bcc_energies, label='Ta-BCC')
ax.plot(fcc_volumes, fcc_energies, label='Ta-FCC')
ax.legend(loc=0, prop={'size':'large'})
ax.set_xlabel('Volume per Ta atom', size='large')
ax.set_ylabel('Total energy per Ta atom (eV/atom)', size='large')
ax.ticklabel_format(useOffset=False)
fig.tight_layout()
plt.savefig('4-3.png')
plt.show()
```

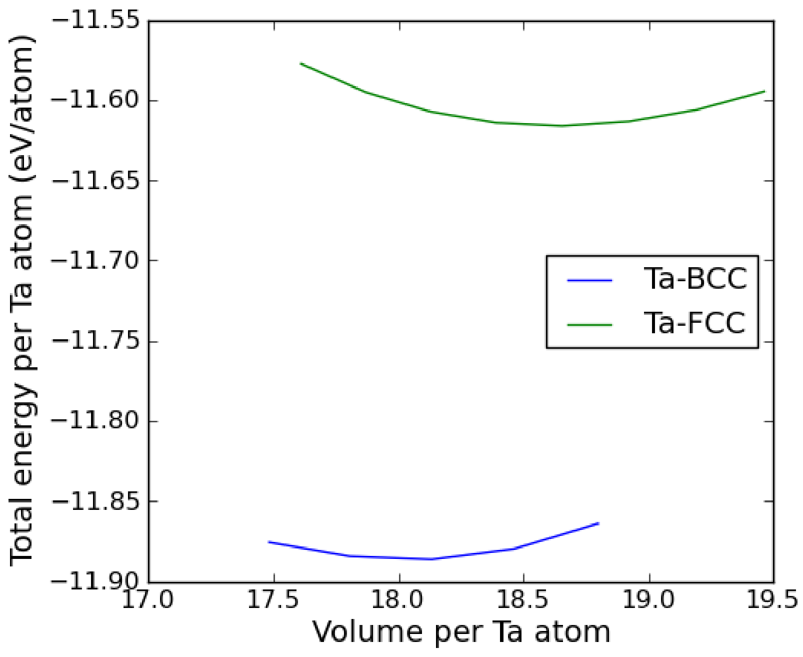

Figure 8: Comparison between the FCC and BCC phases of Ta.

It is clear that the BCC phase of Ta is more stable than the FCC phase of Ta.

5 Determine graphite lattice parameters

The crystal structure of graphite can be found at http://cst-www.nrl.navy.mil/lattice/struk/a9.html (or you can use the builtin `ase` functions).

Compute the geometry optimized total energy for graphite. Use parameters determined from the convergence study to ensure the total energy is converged to better than 50 meV. Compare your results to experimental data. Cite your source.

```
from jasp import *
from ase.lattice.hexagonal import Graphite
from ase.visualize import view

ready = True

graphite = Graphite('C', latticeconstant={'a':2.4612, 'c':6.7079})
with jasp('5.1/trial1', atoms=graphite,
          xc='PBE', lreal=False, istart=0,
          encut=450, prec='Accurate',
          kpts=(8, 8, 8), ismear=1, sigma=0.05, gamma=True,
          ibrion=1, isif=3, nsw=50, ediffg=-0.05,
          ispin=2, lorbit=11) as calc:
    try:
        calc.calculate()
        new_cell = calc.get_atoms().cell
    except (VaspSubmitted, VaspQueued):
        ready = False
        pass
    print 'New Lattice Vectors'
    print '-------------------'
    print 'a1 = {0}'.format(new_cell[0])
    print 'a2 = {0}'.format(new_cell[1])
    print 'a3 = {0}'.format(new_cell[2])
```

```
New Lattice Vectors
-------------------
a1 = [ 2.466  0.     0.   ]
a2 = [-1.233  2.136  0.   ]
a3 = [ 0.     0.     7.075]
```

From these results, we see the relaxed lattice parameters are a=2.466 and c=7.075. The experimental lattice parameter was shown to be a=2.4612 and c=6.7079 from the dft book and a review on graphite [1]. This is not surprising because DFT has been known to not be able to accurately describe Van Der Waals forces, which accounts for the bonding between the carbon layers in graphite.

6 Tantalum carbide lattice parameters

Tantalum carbide is a hard material. Use DFT to compute the lattice constant of cubic tantalum carbide in the rock salt structure (NaCl or B1), and the bulk modulus. The crystal structure of tantalum carbide can be found at http://cst-www.nrl.navy.mil/lattice/struk/b1.html or in the `ase.lattice` module. Compare the bulk modulus of the TaC to that of bcc tantalum. Which is harder?

The experimental lattice constant is 4.455 Å, and the experimental bulk modulus is 3.45 Mbar. How do your results compare to this?

```python
import sys
from jasp import *
from ase.visualize import view
import numpy as np
import matplotlib.pyplot as plt

ready=True
lat = 4.455
space = 0.02
lats = (lat - 2*space,
        lat - space,
        lat,
        lat + space,
        lat + 2*space)
energies, volumes  = [], []
for a in lats:
    a1 = np.array((0, a/2, a/2))
    a2 = np.array((a/2, 0, a/2))
    a3 = np.array((a/2, a/2, 0))
    TaC = Atoms([Atom('Ta', (0, 0, 0), magmom=6),
                  Atom('C', a1/2 + a2/2 + a3/2, magmom=2)],
                cell=(a1, a2, a3))
    with jasp('6.1/a-{0:1.2f}'.format(a),
              xc='PBE', lreal=False, istart=0,
              encut=450, prec='Accurate',
              kpts=(8, 8, 8), ismear=1, sigma=0.05,
```

[1] D. Chung, Journal of Material Science 37, 1475 (2002).

```
              ispin=2, lorbit=11,
              atoms=TaC) as calc:
        try:
            energies.append(TaC.get_potential_energy())
            volumes.append(TaC.get_volume())
        except (VaspSubmitted, VaspQueued):
            energies.append(None)
            ready = False
            pass
if not ready:
    import sys; sys.exit()

from ase.utils.eos import EquationOfState
eos = EquationOfState(volumes, energies)
v0, e0, B = eos.fit()
eqlat = (4*v0) ** (1./3.)
print 'The equilibrium lattice constant of TaC is {0:1.3f} Angstroms'.format(eqlat)
print 'The bulk modulus of TaC is {0:1.3f} eV/Angstroms^3'.format(B)
print 'The bulk modulus of TaC is {0:1.3f} Mbar'.format(B*1.60)
eos.plot('6-1.png')
```

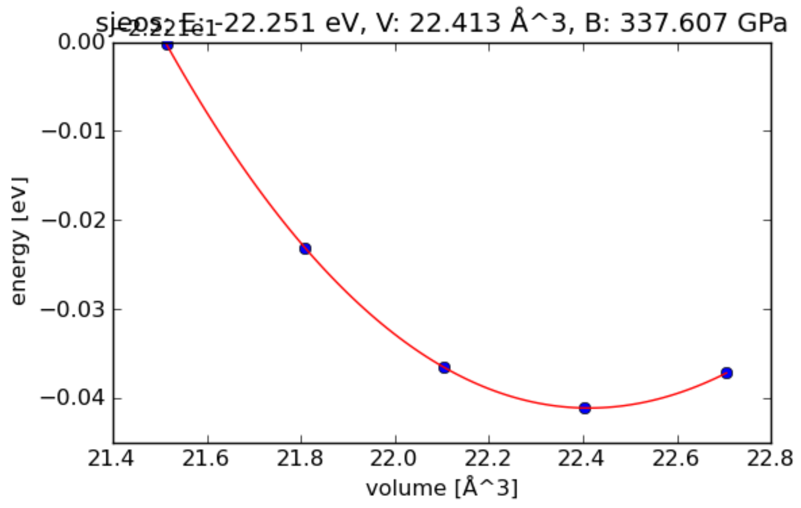

Figure 9: The equation of state of TaC

These results show that the DFT calculated lattice constant of 4.476 Å is higher than the experimental value of 4.455 Å. DFT's calculated bulk modulus of 3.371 Mbar is also lower than 3.45 Mbar.

7 Estimate the formation energy of TaC.

Compute the formation energy of TaC for the reaction Ta + $C_{\text{graphite}} \rightarrow$ TaC. Compare your anser to the value reported in http://pubs.acs.org/doi/pdf/10.1021/j100786a027. Discuss any reasons for discrepancy. Remember that you must use the same ENCUT for all the calculations in this problem, and you must use the largest ENCUT for all calculations that ensures the accuracy level you want.

To do this calculation, we need total energies at equilibrium volumes at 450 eV kinetic energy cutoff – 450 eV being the converged energy of graphite. Note that these calculations for graphite and TaC have already been done, since we used VASP to automatically relax the structure. We will however need to do a calculation of Ta at the equilibrium volume calculated by the EOS.

```python
import sys
from jasp import *

ready = True

# Ta total energy at the equilibrium volume
a = 3.305
a1 = np.array((-a/2, a/2, a/2))
a2 = np.array((a/2, -a/2, a/2))
a3 = np.array((a/2, a/2, -a/2))
Ta = Atoms([Atom('Ta', (0, 0, 0), magmom=6)],
           cell=(a1, a2, a3))
with jasp('7.1/Ta/', atoms=Ta,
          xc='PBE', lreal=False, istart=0,
          encut=450, prec='Accurate',
          kpts=(6, 6, 6), ismear=1, sigma=0.05,
          ispin=2, lorbit=11) as calc:
    try:
        e_Ta = Ta.get_potential_energy()
    except (VaspSubmitted, VaspQueued, VaspRunning):
        ready = False
        pass

# Graphite total energy at the equilibrium volume
with jasp('5.1/trial1') as calc:
    graphite = calc.get_atoms()
    e_graphite = graphite.get_potential_energy()
    e_graphite = e_graphite / 4 # Since this contains 4 units

# TaC total energy at equilibrium volume
a = 4.476
a1 = np.array((0, a/2, a/2))
a2 = np.array((a/2, 0, a/2))
a3 = np.array((a/2, a/2, 0))
TaC = Atoms([Atom('Ta', (0, 0, 0), magmom=6),
             Atom('C', a1/2 + a2/2 + a3/2, magmom=2)],
            cell=(a1, a2, a3))
with jasp('7.1/TaC',
          xc='PBE', lreal=False, istart=0,
          encut=450, prec='Accurate',
          kpts=(8, 8, 8), ismear=1, sigma=0.05,
          ispin=2, lorbit=11,
          atoms=TaC) as calc:
    try:
        e_TaC = TaC.get_potential_energy()
    except (VaspSubmitted, VaspQueued, VaspRunning):
        ready = False
        pass

# We can now compute the reaction energy for the reaction
e_rxn = e_TaC - e_Ta - e_graphite
print 'The computed enthalpy of the reaction Ta + C -> TaC is {0:1.3f} eV/atom'.format(e_rxn)
print 'Converted to kcal/mol, this is then {0:1.3f} kcal/mol'.format(e_rxn * 23.069)
```

```
The compute enthalpy of the reaction Ta + C -> TaC is -1.153 eV/atom
Converted to kcal/mol, this is then -26.598 kcal/mol
```

From the article, they reported a formation enthalpy at room temperature of -34.5 $\pm$ 0.9 kcal/mol. Our value is off by about 10 kcal/mol. This shows there is insufficient cancellation error when comparing BCC Ta, graphite C, and rocksalt TaC. This isn't surprising considering how dissimilar the three systems are. We have a metal, a mineral held together by Van der Waals interactions that are not acounted for, and a compound with potential charge transfer and covalent bonding between a metal and a non-metal.