Project 1 - Due 10/31/2012

Alex Massicotte

2012-10-20 Sat

Contents

1 Introduction

In all areas of research, there are two types of uncertainty: systematic and statistical. Systematic uncertainty arises from the inherent sensitivity of measurement equipment and the theoretical approximations made to simplify simulations, whereas statistical uncertainty arises in the analysis of research data. It is of critical importance to understand the accuracy and precision of one's experiment or simulation, which necessitates a thorough understanding of the relative importance of these two types of uncertainty.

2 Methods

In this mini-project, I compute the statistical uncertainty yielded by fitting the Birch-Murnaghan equation of state to the lattice energy of bcc Tantalum. I use the scipy command 'leastsq' to minimize the sum of the residuals with respect to the paramaters of the Birch-Murnaghan EOS. One could replace

the Birch-Murnaghan equation with any other EOS (or any other non-linear model fit), and this procedure should still work properly. To compute the lattice energies of bcc Ta, I use the PBE exchange correlation with an energy cutoff of 400 eV and a [6 × 6 × 6] kpt-grid, which have been chosen to achieve convergence to 50 meV. I use this value to quantify the systematic uncertainty of the DFT calculation.

For non-linear fitting schemes, there are two available techniques for computing confidence intervals around the fitted parameters: bootstrapping and asymptotic theory. Bootstrapping relies on re-sampling the residuals from a normal distribution to compute standard deviations of the fitted parameters (from which confidence intervals are trivial to compute), whereas asymptotic theory calculates the standard error by estimating the covariance matrix. For constant variance data, there is no clear advantage to using asymptotic theory over boot-strapping other than differences in computation time (asymptotic theory is always faster than bootstrapping). For non-constant variance data, the choice depends on local variation in the data in regions of importance for the estimation of the parameters. If the local variation is small, there will be no significant difference in the prediction of each method, but if the local variation is large, the standard errors for bootstrapping will be larger than those of asymptotic theory, and hence more conservative and also more accurate. For constant X (in this case, volume) 'constant' and 'non-constant' variance refer to the variance in Y (lattice energy) as a function of X. Since bootstrapping is at least more conservative than asymptotic theory in all cases and only suffers a longer computation time, establishing the condition of the variance in lattice energies as a function of volume is not performed here, and the bootstrapping method is used exclusively.

3 Discussion

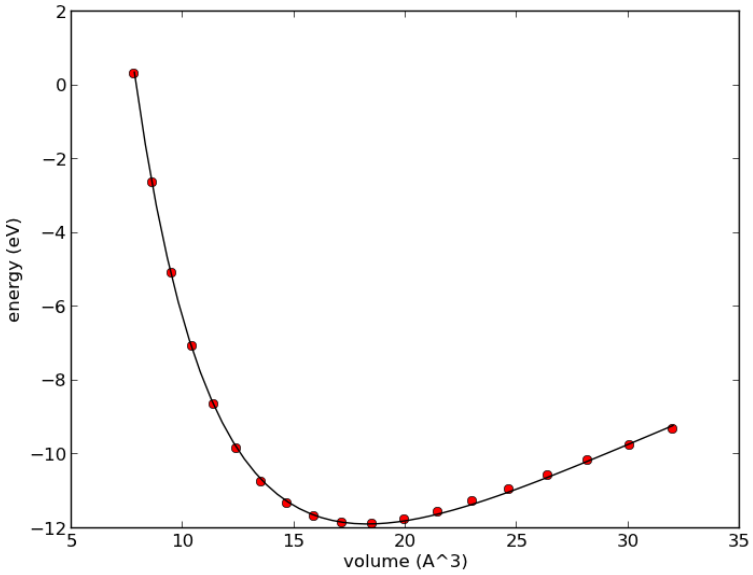
To investigate the relative significance of systematic and statistical uncertainty in the EOS fit to bcc Ta, I compare the confidence intervals of bulk modulus, minimum volume, and minimum energy when the EOS fit is performed on data computed for two different ranges of volumes. Not surprisingly, the statistical uncertainty increases as the volume range increases, indicating that the EOS fit gets worse further from the volume minimum. For the purposes of this project, the systematic uncertainty (50 meV) is a fact of life, so the question then becomes: "How small does the volume range need to be in order for the statistical uncertainty to be less than the systematic uncertainty?" What follows is the computation of the 95% confidence

intervals for both volume ranges.

3.1 Large volume range (2.4 - 4 A)

```
from ase import Atom, Atoms
from jasp import *
from scipy.optimize import leastsq

import numpy as np

LC = np.linspace(2.5,4.,19)
TE = []
VOL = []

ready = True
for a in LC:
    atoms = Atoms([Atom('Ta',(0,0,0))],
                  cell=0.5*a*np.array([[1.0, 1.0, -1.0],
                                       [-1.0, 1.0, 1.0],
                                       [1.0, -1.0, 1.0]]))

    with jasp('bulk/Ta-LC-{0:.2f}'.format(a),
                  xc='PBE',
                  kpts = (6,6,6),
                  encut = 400,
                  atoms = atoms) as calc:
        try:
            TE.append(atoms.get_potential_energy())
            VOL.append(atoms.get_volume())
        except (VaspSubmitted, VaspQueued):
            ready = False

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

vols = np.array(VOL)
energies = np.array(TE)

def BirchMurnaghan(parameters,vol):
    E0 = parameters[0]
    B0 = parameters[1]
    BP = parameters[2]
    V0 = parameters[3]

    E = E0 + 9*B0*V0/16*(BP*((V0/vol)**(2./3.)-1.)**3. + (6-4*(V0/vol)**(2./3.))*((V0/vol)**(2./3.)-1.)**2.)

    return E

def objective(pars,y,x):
    err =  y - BirchMurnaghan(pars,x)
    return err

x0 = [ -11., 3., 4., 18]
best_fit = leastsq(objective, x0, args=(energies,vols))
res = objective(best_fit[0],energies,vols)
```

```
s_res = np.std(res)

E0b = []
B0b = []
BPb = []
V0b = []
for i in range(1000):
    new_en = energies + np.random.normal(0,s_res,len(energies))
    new_fit = leastsq(objective, x0, args=(new_en,vols))[0]
    E0b.append(new_fit[0])
    B0b.append(new_fit[1])
    BPb.append(new_fit[2])
    V0b.append(new_fit[3])

E0mean = np.mean(np.array(E0b))
E0std = np.std(np.array(E0b))
B0mean = np.mean(np.array(B0b))
B0std = np.std(np.array(B0b))
BPmean = np.mean(np.array(BPb))
BPstd = np.std(np.array(BPb))
V0mean = np.mean(np.array(V0b))
V0std = np.std(np.array(V0b))


from pylab import *
plot(vols,energies,'ro')

x = np.linspace(min(vols),max(vols),50)
y = BirchMurnaghan(best_fit[0],x)
plot(x,y,'k-')
xlabel('volume (A^3)')
ylabel('energy (eV)')
savefig('bccTaFAR.png')
show()

print '95% confidence intervals:'
print 'B0 = {0:.3f} +/- {1:.3f} eV/A^3'.format(B0mean,B0std*1.96)
print 'V0 = {0:.3f} +/- {1:.3f} A^3'.format(V0mean,V0std*1.96)
print 'E0 = {0:.3f} +/- {1:.3f} eV'.format(E0mean,E0std*1.96)
```



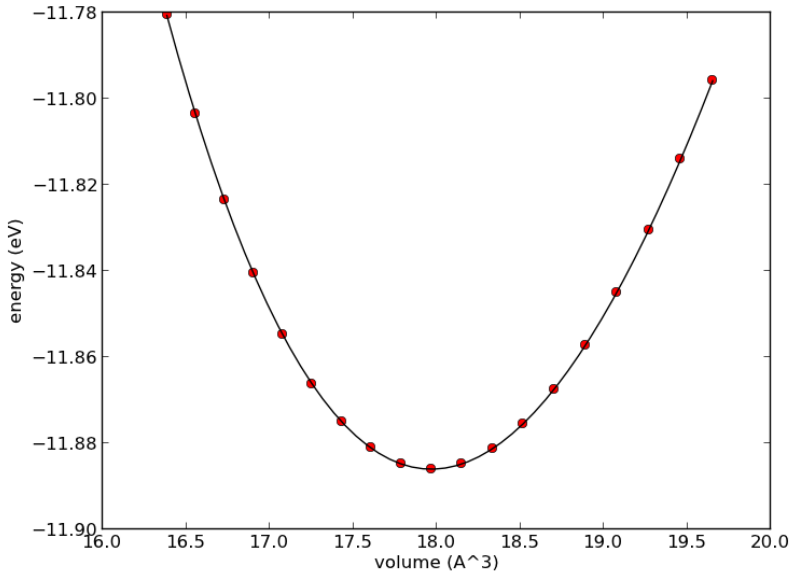

```
95% confidence intervals:
B0 = 1.245 +/- 0.030 eV/A^3
V0 = 18.275 +/- 0.067 A^3
E0 = -11.881 +/- 0.039 eV
```

3.2 Small volume range (3.2 - 3.4 A)

```
from ase import Atom, Atoms
from jasp import *
from scipy.optimize import leastsq

import numpy as np

LC = np.linspace(3.2,3.4,19)
TE = []
VOL = []

ready = True
for a in LC:
    atoms = Atoms([Atom('Ta',(0,0,0))],
              cell=0.5*a*np.array([[1.0, 1.0, -1.0],
                                   [-1.0, 1.0, 1.0],
                                   [1.0, -1.0, 1.0]]))

    with jasp('bulk/Ta-LC-{0:.2f}'.format(a),
              xc='PBE',
              kpts = (6,6,6),
```

```
                encut = 400,
                atoms = atoms) as calc:
        try:
            TE.append(atoms.get_potential_energy())
            VOL.append(atoms.get_volume())
        except (VaspSubmitted, VaspQueued):
            ready = False

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

vols = np.array(VOL)
energies = np.array(TE)

def BirchMurnaghan(parameters,vol):
    E0 = parameters[0]
    B0 = parameters[1]
    BP = parameters[2]
    V0 = parameters[3]

    E = E0 + 9*B0*V0/16*(BP*((V0/vol)**(2./3.)-1.)**3. + (6-4*(V0/vol)**(2./3.))*((V0/vol)**(2./3.)-1.)**2.)

    return E

def objective(pars,y,x):
    err =  y - BirchMurnaghan(pars,x)
    return err

x0 = [ -11., 3., 4., 18]
best_fit = leastsq(objective, x0, args=(energies,vols))
res = objective(best_fit[0],energies,vols)
s_res = np.std(res)

E0b = []
B0b = []
BPb = []
V0b = []
for i in range(1000):
    new_en = energies + np.random.normal(0,s_res,len(energies))
    new_fit = leastsq(objective, x0, args=(new_en,vols))[0]
    E0b.append(new_fit[0])
    B0b.append(new_fit[1])
    BPb.append(new_fit[2])
    V0b.append(new_fit[3])

E0mean = np.mean(np.array(E0b))
E0std = np.std(np.array(E0b))
B0mean = np.mean(np.array(B0b))
B0std = np.std(np.array(B0b))
BPmean = np.mean(np.array(BPb))
BPstd = np.std(np.array(BPb))
V0mean = np.mean(np.array(V0b))
V0std = np.std(np.array(V0b))


from pylab import *
```

```
77  plot(vols,energies,'ro')
78
79  x = np.linspace(min(vols),max(vols),50)
80  y = BirchMurnaghan(best_fit[0],x)
81  plot(x,y,'k-')
82  xlabel('volume (A^3)')
83  ylabel('energy (eV)')
84  savefig('bccTaNEAR.png')
85  show()
86
87  print '95% confidence intervals:'
88  print 'B0 = {0:.3f} +/- {1:.3f} eV/A^3'.format(B0mean,B0std*1.96)
89  print 'V0 = {0:.3f} +/- {1:.3f} A^3'.format(V0mean,V0std*1.96)
90  print 'E0 = {0:.3f} +/- {1:.3f} eV'.format(E0mean,E0std*1.96)
```




```
95% confidence intervals:
B0 = 1.311 +/- 0.001 eV/A^3
V0 = 17.967 +/- 0.001 A^3
E0 = -11.886 +/- 0.000 eV
```

For the larger of the two ranges (2.4 - 4 A), the 95% confidence level surrounding the energy minimum is 76 meV wide. This value is on the order of our systematic uncertainty (50 meV). For the smaller range (3.2 - 3.4 A), the 95% confidence level around the energy minimum is virtually 0

eV, from which it can be concluded that a fit performed over this range of volumes yields a negligible statistical uncertainty.

However, my primary interest is the uncertainties in the bulk modulus and minimum volume estimates. These are bounded by the systematic uncertainty stemming from our choice of ENCUT and k-point grid, since we can easily choose an EOS fit which yields a negligible statistical uncertainty. So how can we quantify the effect of these choices on the uncertainties in the bulk modulus and minimum volume? I do this by selecting a volume range over which the EOS fit yields an energy uncertainty just a bit larger than the systematic uncertainty, ensuring that any prediction made by the fitting scheme suffers a total uncertainty upper-bounded by the statistical uncertainty. The computation we performed on the larger volume range (2.4 - 4 A) fits this description well, so we can use the values it predicts as conservative estimates of the bulk modulus and minimum volume uncertainties.

Again these values are as follows:

B0 = 1.245 +/- 0.029 eV/A^3

V0 = 18.271 +/- 0.071 A^3