EOS*fit*

Bruno A. Calfa

December 10, 2012

Contents

1 Introduction

The EOS*fit* package, written in Python, contains a number of Equations of State (EOS) models to fit the first-principles calculated static energy, E, (without zero-point vibrational energy) *versus* volume, V, points [Shang et al., 2010]. In addition to estimating the parameters of the EOS models, it computes the *confidence interval* of each parameter using the Student's t-distribution.

The EOS models supported are as follows:

- ***5-parameter Birch-Murnaghan (BM5)***

$$E(V) = a + bV^{-2/3} + cV^{-4/3} + dV^{-2} + eV^{-8/3}$$

where a, b, c, d, and e are fitting parameters.

- ***Modified 5-parameter Birch-Murnaghan (mBM5)***

$$E(V) = a + bV^{-1/3} + cV^{-2/3} + dV^{-1} + eV^{-4/3}$$

where a, b, c, d, and e are fitting parameters.

- ***4-parameter Birch-Murnaghan (BM4)***

$$E(V) = a + bV^{-2/3} + cV^{-4/3} + dV^{-2}$$

where a, b, c, and d are fitting parameters.

- ***Modified 4-parameter Birch-Murnaghan (mBM4)***

$$E(V) = a + bV^{-1/3} + cV^{-2/3} + dV^{-1}$$

where a, b, c, and d are fitting parameters.

- ***5-parameter Logarithmic (LOG5)***

$$E(V) = a + b\ln V + c(\ln V)^2 + d(\ln V)^3 + e(\ln V)^4$$

where a, b, c, d, and e are fitting parameters.

- ***4-parameter Logarithmic (LOG4)***

$$E(V) = a + b\ln V + c(\ln V)^2 + d(\ln V)^3$$

where a, b, c, and d are fitting parameters.

- ***4-parameter Murnaghan (MU4)***

$$E(V) = a + \frac{B_0 V}{B_0'}\left(1 + \frac{(V_0/V)^{B_0'}}{B_0' - 1}\right)$$

where $a = E_0 - \frac{B_0 V_0}{B_0' - 1}$. The fitting parameters and their meaning are: E_0 (equilibrium energy), B_0 (equilibrium bulk modulus), B_0' (first derivative of B_0 with respect to pressure), and V_0 (equilibrium volume).

- ***4-parameter Vinet (VI4)***

$$E(V) = a - \frac{4B_0V_0}{(B_0'-1)^2}\left\{1 - \frac{3}{2}(B_0'-1)\left[1 - \left(\frac{V_0}{V}\right)^{1/3}\right]\right\} \times \exp\left\{\frac{3}{2}(B_0'-1)\left[1 - \left(\frac{V_0}{V}\right)^{1/3}\right]\right\}$$

where $a = E_0 + \frac{4B_0V_0}{(B_0'-1)^2}$. The fitting parameters and their meaning are same as in MU4.

- ***4-parameter Morse (MO4)***

$$E(V) = a + b\exp(dV^{1/3}) + c\exp(2dV^{1/3})$$

where a, b, c, and d are fitting parameters.

The models BM5, mBM5, BM4, mBM4, LOG5, and LOG5 are linear (in the fitting parameters), whereas the models MU4, VI4, and MO4 are nonlinear.

2 Methodology

2.1 Linear and Nonlinear Regression with Confidence Intervals

In order to estimate the parameters of the linear models, the Moore-Penrose Pseudoinverse is calculated and the solution is obtained in a least-squares sense. The procedure is as follows. Each linear EOS model can be written in the form $y = Xp$, where y is the vector of energy values, X is the regressor matrix whose columns contain each term in V, and p is the vector of parameters to be estimated. Therefore, the parameters are calculated by:

$$\begin{aligned} Xp &= y \\ X^T X p &= X^T y \\ (X^T X)^{-1} X^T X p &= (X^T X)^{-1} X^T y \\ p &= X^+ y \end{aligned}$$

where $X^+ = (X^T X)^{-1} X^T$ is the pseudoinverse of X. Remark: in the actual implementation, no matrix is directly inverted. The computationally efficient way to calculate the optimal p is to solve the linear system $X^T X p = X^T y$ for p using `numpy.linalg.solve`.

The parameters of the nonlinear models are estimated using the function `curve_fit` available in the SciPy package. The function `curve_fit` not only computes the optimal parameters, given a ***good*** (this cannot be emphasized more!) initial guess, but also the estimated covariance matrix evaluated at the solution. The covariance matrix can be used to estimate the confidence intervals for each parameter as described in [Bates and Watts, 1988].

The basic procedure to obtain the confidence intervals for the linear models is given below:

1. Compute optimal parameters: $p^* = X^+ y$
2. Calculate residuals: $r = y - Xp^*$
3. Calculate degrees of freedom: $dof = n_y - n_p$, where $n_y = \text{length}(y)$ and $n_p = \text{length}(p^*)$
4. Obtain diagonal elements of covariance matrix: $dcov_i = \text{diag}\left(\sum_{j=1}^{n_p} R^{-1}_{i,j} R^{-1}_{i,j}\right)$, where $\text{diag}(\cdot)$ retrieves the elements in the main diagonal of its argument and R^{-1} is the inverse of the upper triangular matrix R obtained from QR decomposition of X^+
5. Obtain estimated residual variance: $rmse = \frac{||r||_2^2}{dof}$, where $||\cdot||_2$ is the 2-norm
6. Obtain confidence intervals on parameters: $ci = p^* \pm tinv(1-\alpha, dof) rmse\sqrt{dcov}$, where α is the quantile ($\alpha = 0.05$ means 95% of confidence), $tinv(\cdot,\cdot)$ is the Student's t inverse survival function

For nonlinear models, the covariance matrix is already estimated by the function `curve_fit` and only the last step above (with $rmse = 1$) has to be performed to get the confidence intervals.

2.2 Confidence Intervals of Physical Parameters via Simulation

The previous subsection describes the steps to obtain the confidence intervals (CIs) of the fitting parameters. For models whose fitting parameters do not have physical meaning, we have to somehow translate the uncertainty in them to physical parameters of interest. Even though those fitting parameters, say $(a, b, c, d, \ldots)$, are mathematically related to the physical parameters [Shang et al., 2010], such as $(V_0, E_0, B_0, B_0', B_0'')$, it is very difficult to analytically express them as functions of the fitting parameters. Therefore, EOS*fit* takes the following approach.

After estimating the fitting parameters and computing their CIs, the uncertainty or error in those parameters are *propagated* to the physical parameters by means of simulation. Sets of points, 10,000 points for example, are sampled for each fitting parameter from a Gaussian probability distribution function with the same average and standard deviation as the fitting parameters'. For example, the parameter a can be expressed as $a = \bar{a} \pm \Delta a$, where $\bar{a}$ is the average of a and Δa is the uncertainty or error associated with estimating a that is computed using the inverse of the t-Distribution as explained in the previous subsection. Therefore, the Gaussian random generator takes $\bar{a}$ as the average and $\frac{\Delta a}{tinv(1-\alpha, dof) rmse \sqrt{dcov_a}}$ as the standard deviation.

Next, distributions of physical parameters values are obtained by applying those sample points into analytical expressions for each physical parameter (obtained via a powerful symbolic computation software, such as Maplesoft Maple or Wolfram Mathematica). Finally, from each distribution of physical parameters values and given a percentile, we extract the corresponding values at both ends of the distribution and compute the lower and upper bounds of CIs of the physical parameters. **Remark:** the main assumption here is that all parameters are uncorrelated.

As an example, take the BM4 model, whose energy is given by:

$$E(V) = a + bV^{-2/3} + cV^{-4/3} + dV^{-2}$$

The equilibrium volume, V_0, is obtained by differentiating $E(V)$, setting the result to zero and solving the equation for V. For the BM4 model, four roots arise:

$$V_0^{(1)} = \frac{\sqrt{b\left(-c+\sqrt{-3bd+c^2}\right)}\left(-c+\sqrt{-3bd+c^2}\right)}{b^2}$$

$$V_0^{(2)} = -\frac{\sqrt{b\left(-c+\sqrt{-3bd+c^2}\right)}\left(-c+\sqrt{-3bd+c^2}\right)}{b^2}$$

$$V_0^{(3)} = -\frac{\sqrt{-b\left(c+\sqrt{-3bd+c^2}\right)}\left(c+\sqrt{-3bd+c^2}\right)}{b^2}$$

$$V_0^{(4)} = \frac{\sqrt{-b\left(c+\sqrt{-3bd+c^2}\right)}\left(c+\sqrt{-3bd+c^2}\right)}{b^2}$$

There may be complex conjugate roots depending on the values of the fitting parameters sampled. Therefore, for each sample point, we use the $V_0^{(i)}$ that is real and is contained in the interval of the volume data points given by the user. The valid values for V_0 form its distribution. The following definitions are known [Shang et al., 2010]:

$$P(V) = -V\frac{\partial E}{\partial V}$$
$$B(V) = V\frac{\partial^2 E}{\partial V^2}$$
$$B'(V) = \frac{\partial B}{\partial P} = \frac{\partial B}{\partial V}\bigg/\frac{\partial P}{\partial V}$$

Evaluating each term the analytical expressions are:

$$P(V) = \frac{2}{3}\frac{b}{V^{2/3}} + \frac{4}{3}\frac{c}{V^{4/3}} + 2\frac{d}{V^2}$$
$$B(V) = \frac{10}{9}\frac{b}{V^{5/3}} + \frac{28}{9}\frac{c}{V^{7/3}} + 6\frac{d}{V^3}$$
$$B'(V) = \frac{1}{6}\frac{25bV^{5/3} + 98Vc + 243d\sqrt[3]{V}}{V\left(bV^{5/3} + 4Vc + 9d\sqrt[3]{V}\right)}$$

Thus, we obtain the distributions of the remaining physical parameters by evaluating them at the valid V_0 points. Finally the CIs for each physical parameters are computed by using the average values, $\bar{p}$, of all their distributions, and lower and upper bounds computed by $p_{\alpha/2}$ and $p_{(1-\alpha)/2}$, respectively, where α is the percentile ($\alpha = 0.05$ denotes 95% confidence).

2.3 Coefficient of Determination (R^2)

The coefficient of determination, commonly known as R^2, is calculated as follows. However, it does not reflect the accuracy of the estimated parameters given by the confidence intervals.

$$R^2 = 1 - \frac{SSE}{SST}$$

where $SSE = \sum_i (E_i - EOS_i)^2$ is the sum of squares of residuals and $SST = \sum_i (E_i - \bar{E})^2$ is the total sum of squares ($\bar{E} = \frac{1}{n_E}\sum_i E_i$).

3 Using the EOS*fit* Package

The EOS*fit* package is composed of a Python module called `eosfit`, which contains the class `EOS`. The user instantiates an object from the class `EOS` by passing the volumetric and energy data arrays. An optional argument is the EOS model (default: MU4). The selection of the model is done through class `EOSmodel`, which serves as an "`enum`" type (commonly found in programming languages such as C and others).

3.1 Fitting Data to an EOS

The following listing shows an example of fitting data to MU4 model (nonlinear). The keyword argument `ID` can be used to set a "name tag" for the EOS object.

```
from eosfit import EOS, EOSmodel
import numpy as np

eos = EOS(V, E, ID='MU4')
p0 = [12., -3., 1., 5.] # [V0, E0, B0, B0']
V0_MU4, E0_MU4, B0_MU4 = eos.fit(p0) # Initial guess required
ci_MU4 = eos.get_ci() # Confidence intervals
E_MU4 = eos.eval() # Evaluates de EOS model at the V points
R2_MU4 = eos.get_rsquared() # Coefficient of determination
eos.plot(filename='eosfit_example_MU4.png')
print '''MU4
===
V0 = {0} Ang^3
E0 = {1} eV/atom
B0 = {2} GPa
ci = {3}
E = {4} eV/atom
R^2 = {5}
'''.format(V0_MU4, E0_MU4, B0_MU4*160.217, ci_MU4, E_MU4, R2_MU4)
```

Python output:

```
MU4
===
V0 = 10.8180826775 Ang^3
E0 = -5.59091558733 eV/atom
B0 = 193.481487478 GPa
ci = [[ 10.76183537  10.87432998]
 [ -5.6001798   -5.58165137]
 [  1.16635588   1.24888704]
 [  5.32144772   6.28156027]]
E = [-4.64917776 -5.04952738 -5.30375636 -5.48119385 -5.56648891 -5.59054738
 -5.56168396 -5.50893294 -5.41161975 -5.29348378 -5.17858792] eV/atom
R^2 = 0.999422638026
```

The resulting graph is shown in Figure 1. The estimated parameters and their confidence intervals are displayed in the title of the figure.

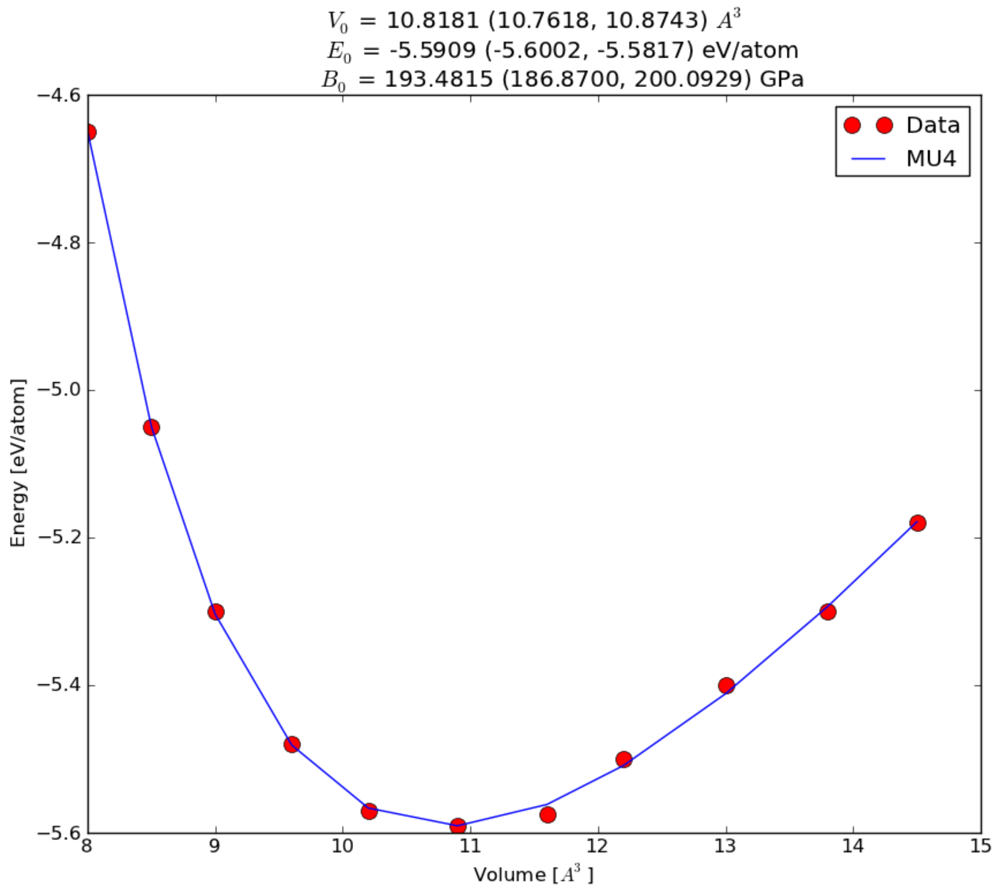


Figure 1: Fitting parameters of MU4 model to experimental data.

3.2 Comparing EOS Models

To illustrate the relevance of computing confidence intervals, the models mBM4 (linear) and VI4 (nonlinear) were fit to the same data as above and the results are given in Table 1. Note that the confidence intervals for VI4 are much *tighter* than for mBM4. Moreover, the uncertainty in V_0 is propagated to the other physical parameters and the estimated values for B_0 and B_0' differ significantly between the two models. The method `get_ci()` returns the CIs for the models' parameters (a, b *etc.*), whereas the method `get_phys_ci()` retrieves the CIs for the physical parameters (V_0, E_0 *etc.*). The static method `EOS.plotm` can be used to plot *multiple* EOS models in the same figure (Figure 2). We use the values of the physical parameters calculated from the linear model as the initial guesses for fitting the nonlinear model.

```
from eosfit import EOS, EOSmodel
import numpy as np

# Data
V = np.array([8., 8.5, 9., 9.6, 10.2, 10.9, 11.6, 12.2, 13., 13.8,
    14.5]) # [Ang^3]
```

```python
E = np.array([-4.65, -5.05, -5.3, -5.48, -5.57, -5.59, -5.575, -5.5,
    -5.4, -5.3, -5.18]) # [eV/atom]

# Linear EOS model
eos1 = EOS(V, E, ID='mBM4', model=EOSmodel.mBM4)
V0_mBM4, E0_mBM4, B0_mBM4 = eos1.fit() # No need for initial guess (
    linear model)
ci_mBM4 = eos1.get_phys_ci()
print '''mBM4
===
V0 = {0} Ang^3
E0 = {1} eV/atom
B0 = {2} GPa
phys ci = {3}
'''.format(V0_mBM4, E0_mBM4, B0_mBM4*160.217, ci_mBM4)

# Nonlinear EOS model
eos2 = EOS(V, E, ID='VI4', model=EOSmodel.VI4)
p0 = [V0_mBM4, E0_mBM4, B0_mBM4, eos1.get_B0p()] # Initial guesses from
     fitting linear model
V0_VI4, E0_VI4, B0_VI4 = eos2.fit(p0) # Provide custom initial guesses
ci_VI4 = eos2.get_ci()
print '''VI4
===
V0 = {0} Ang^3
E0 = {1} eV/atom
B0 = {2} GPa
ci = {3}
'''.format(V0_VI4, E0_VI4, B0_VI4*160.217, ci_VI4)

# Plot multiple EOS objects
EOS.plotm([eos1, eos2], filename='eosfit_example_mBM4_VI4.png')
```

Python output:

```
mBM4
===
V0 = 10.7872029523 Ang^3
E0 = -5.59723430941 eV/atom
B0 = 215.190378096 GPa
phys ci = [[  1.34884130e-01   2.17170090e+01]
 [ -1.21278177e+01  -9.31396839e-01]
 [  1.07699102e-01   2.79254664e+00]
 [  2.92989209e-03   1.12319857e+00]]

VI4
```

```
===
V0 = 10.7854999291 Ang^3
E0 = -5.59594934529 eV/atom
B0 = 210.395960324 GPa
ci = [[ 10.7287233   10.84227656]
 [ -5.60490728  -5.58699141]
 [  1.27277874   1.35360873]
 [  5.6198055    6.59195636]]
```

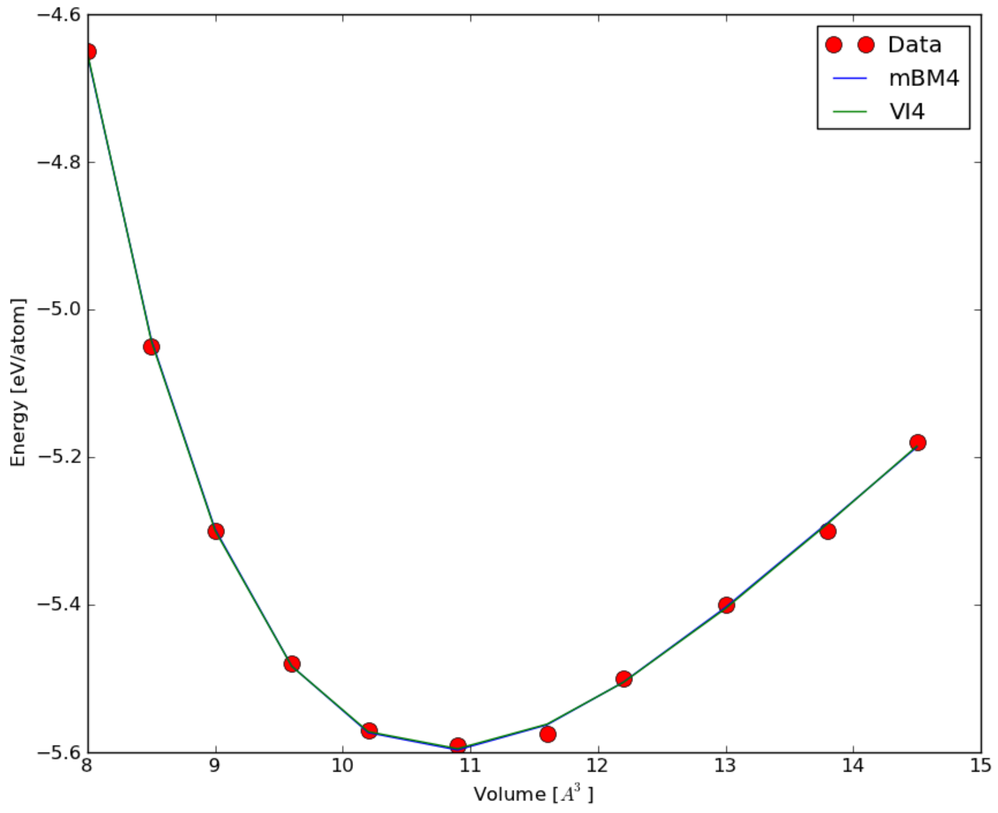


Figure 2: Fitting parameters of mBM4 and VI4 model to experimental data.

Table 1: Optimal parameters and their confidence intervals for models BM4 and VI4 (95% confidence).

	mBM4		VI4	
Parameter	p^*	ci	p^*	ci
V_0 [Å^3]	10.7872	[0.1349, 21.7170]	10.7855	[10.7287, 10.8423]
E_0 [eV]	−5.5972	[−12.1252, −0.9130]	−5.5959	[−5.6049, −5.5870]
B_0 [GPa]	215.1904	[16.9456, 448.1337]	210.3960	[203.9208, 216.8711]
B_0' [-]	0.5598	[0.0029, 1.1232]	6.1059	[5.6198, 6.5920]

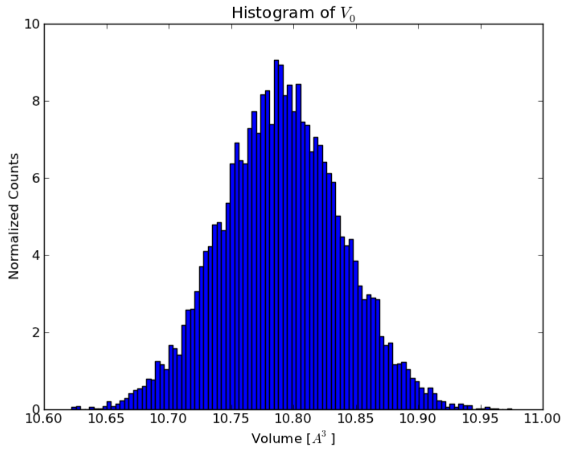
3.3 Plotting Distribution of Physical Parameters

The estimated values and the confidence intervals of the physical parameters are computed via simulation as described in subsection 2.2. Therefore, it is possible to graph the distribution of values (histograms) for each physical parameter calculated. The listing below demonstrates how to do that for the BM4 model.

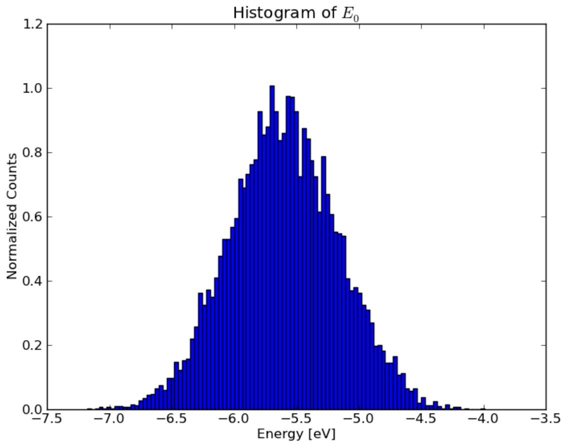
```
from eosfit import EOS, EOSmodel
import numpy as np

# Data
V = np.array([8., 8.5, 9., 9.6, 10.2, 10.9, 11.6, 12.2, 13., 13.8,
    14.5]) # [Ang^3]
E = np.array([-4.65, -5.05, -5.3, -5.48, -5.57, -5.59, -5.575, -5.5,
    -5.4, -5.3, -5.18]) # [eV/atom]

# Construct EOS model
eos = EOS(V, E, ID='BM4', model=EOSmodel.BM4)
V0_BM4, E0_BM4, B0_BM4 = eos.fit() # No need for initial guess (linear
    model)
ci_BM4 = eos.get_phys_ci() # Done via simulation (need many samples for
    good statistical representation)
print '''BM4
===
V0 = {0} Ang^3
E0 = {1} eV/atom
B0 = {2} GPa
phys ci = {3}
'''.format(V0_BM4, E0_BM4, B0_BM4*160.217, ci_BM4)

# Plot distributions
eos.plot_hist_V0('eosfit_example_BM4_V0_dist.png')
eos.plot_hist_E0('eosfit_example_BM4_E0_dist.png')
eos.plot_hist_B0('eosfit_example_BM4_B0_dist.png')
eos.plot_hist_B0p('eosfit_example_BM4_B0p_dist.png')
```

Python output:

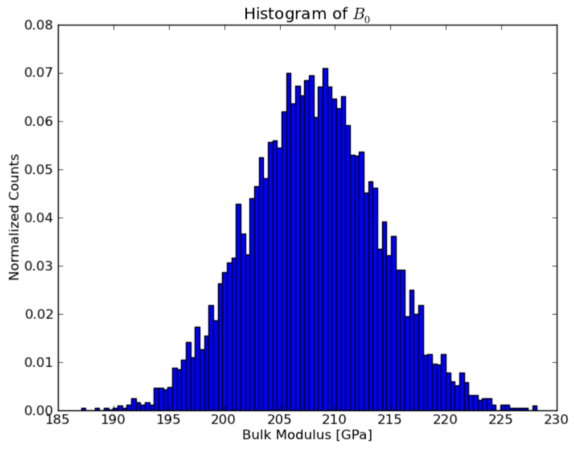
```
BM4
===
V0 = 10.790820159 Ang^3
E0 = -5.59139487431 eV/atom
B0 = 208.000370067 GPa
phys ci = [[  9.54185640e-02   2.16789085e+01]
 [ -1.20175172e+01  -8.43538613e-01]
 [  7.26400369e-02   2.67023104e+00]
 [  2.02831075e-03   1.12430906e+00]]
```



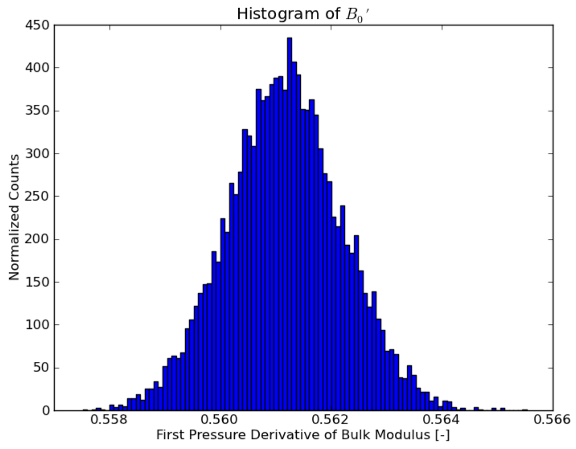
(a) Histogram for V_0 for BM4 model.



(b) Histogram for E_0 for BM4 model.



(c) Histogram for B_0 for BM4 model.



(d) Histogram for B_0' for BM4 model.

Figure 3: Distribution of physical parameters for BM4 model.

4 Conclusions and Extensions

The EOS*fit* module enables a more comprehensive statistical analysis of fitting EOS models to calculated first-principles energy and volume data. It was demonstrated that the uncertainty associated with fitting parameters of certain EOS models, and propagated to the physical parameters of interest, resulted in wider confidence intervals for the latter and also significantly different values from those obtained with models whose fitting parameters are already the physical parameters, which had tighter confidence intervals.

Possible extensions to this module include the implementation of other EOS models, the calculation of higher-order pressure derivatives of the bulk modulus for 4-parameter models, and the calculation of additional physical parameters from the $E(V)$ expressions of each EOS model.

References

D. M. Bates and D. G. Watts. *Nonlinear Regression Analysis and Its Applications*. John Wiley & Sons, 1988.

S-L. Shang, Y. Wang, D. Kim, and Z-K Liu. First-principles Thermodynamics from Phonon and Debye Model: Application to Ni and Ni_3Al. *Computational Materials Science*, 47(4): 1040–1048, 2010.