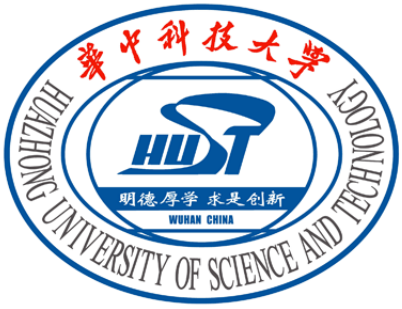
Homework of Computational Materials

Equilibrium Lattice Constant and Young's Modulus of Diamond

with Tersoff potential

Domicor

g1114128525@gmail.com



Huazhong University of Science and Technology
Department of Materials Science and Technology
Faculty for Materials Science and Technology

Contents

Abstract

This article discuss the molecular dynamics simulation solution to the calculation of the equilibrium lattice constant and Young's modulus of diamond. With the observation that the cohesive energy of diamond would reach the global minimum value when in equilibrium, or in terms of calculation, when the lattice constant reaches the extreme point, we have the calculated result of the lattice constant of 3.5659 Å, with cohesive energy of -7.3682 eV, which is close to the numerical result given by literature, 3.57 Å. The Young's modulus of diamond is investigated through the intrinsic thermal vibration in diamond which can be simulated by molecular dynamics and the result 1100.19107894 GPa agrees very well with the experiment data, about 1100 GPa, provided by two sources including a journal-like article but barely meets the quantity proposed by wikipedia, 1220 GPa.

1. Analysis

1.1. Tersoff Potential

The formulae adopted by Lammps to calculate the 3-body Tersoff potential employed with the command pair_style tersoff are as follows[1]:

$$
\begin{aligned}
E &= \frac{1}{2}\sum_i \sum_{j\neq i} V_{ij} \\
V_{ij} &= f_C(r_{ij})\left[f_R(r_{ij}) + b_{ij} f_A(r_{ij})\right] \\
f_C(r) &= \begin{cases} 1 & : \quad r < R - D \\ \frac{1}{2} - \frac{1}{2}\sin\left(\frac{\pi}{2}\frac{r-R}{D}\right) & : \quad R - D < r < R + D \\ 0 & : \quad r > R + D \end{cases} \\
f_R(r) &= A\exp(-\lambda_1 r) \\
f_A(r) &= -B\exp(-\lambda_2 r) \\
b_{ij} &= (1 + \beta^n {\zeta_{ij}}^n)^{-\frac{1}{2n}} \\
\zeta_{ij} &= \sum_{k\neq i,j} f_C(r_{ik}) g(\theta_{ijk}) \exp\left[{\lambda_3}^m (r_{ij} - r_{ik})^m\right] \\
g(\theta) &= \gamma_{ijk}\left(1 + \frac{c^2}{d^2} - \frac{c^2}{[d^2 + (\cos\theta - \cos\theta_0)^2]}\right)
\end{aligned}
$$

where f_R is a two-body term and f_A includes three-body interactions. The summations in the formulae are over all neighbors J and K of atom I within a cutoff distance = R + D.

In Lammps, the potential should be defined in the format:

```
element 1, element 2, element 3, m, gamma, lambda3, c, d,
    costheta0, n, beta, lambda2, B, R, D, lambda1, A
```

For a single-element simulation, only a single entry is required (e.g. CCC), which is the case of our simulation. For a two-element simulation, the file must contain 8 entries (for

[1] The formulae can been found in the article by Tersoff [Ter88]

SiSiSi, SiSiC, SiCSi, SiCC, CSiSi, CSiC, CCSi, CCC), that specify Tersoff parameters for all permutations of the two elements interacting in three-body configurations. Thus for 3 elements, 27 entries would be required, etc.

The SiC.tersoff file[2] provided by Tersoff[Ter89, Ter90] includes the permutation CCC[3] thus is qualified for our calculation of the properties of diamond. The data is:

```
C C C 3.0 1.0 0.0 38049 4.3484 -.57058 .72751
0.00000015724 2.2119 346.7 1.95 0.15 3.4879
1393.6
```

1.2. Equilibrium Lattice Constant

1.2.1. Definition

The lattice constant, or lattice parameter, refers to the physical dimension of unit cells in a crystal lattice[lat14].

1.2.2. Method

The cohesive energy of diamond would reach the global minimum value when in equilibrium, or in terms of calculation, when the lattice constant reaches the extreme point. So to calculate the equilibrium lattice constant, acquire the Cohesive energy-Lattice constant curve and then find the minimum point of it. This will yield the solution to this problem.

1.2.3. Simulation design

The boundaries are set to be periodic with units metal. The shape of the simulation region is a box (or block in terms), sized 10x10x10 and filled with "atomic" carbon atoms (so the "mass" is set to be equal to the molar mass of carbon, 12 grams/mol).

The lattice style is set to be diamond with the lattice constant[4] starting from 3.5, incrementing 0.01 every loop until the script loops 20 times. With a time step[5] of 0.005, the thermodynamic info are printed every 10 time steps.

The energy minimization process by iteratively adjusting atom coordinates are performed with the Polak-Ribiere version of the conjugate gradient (CG) algorithm, which repeats at most 1000 times.

After acquiring the local potential energy minimum (hopefully), we use the formula 1.1 to calculate the cohesive energy with the specific lattice constant.

$$CohesiveEnergy = \frac{TotalPotentialEnergy}{TheNumberOfAtoms} \tag{1.1}$$

[2] which resides in the "potentials" folder of Lammps distribution

[3] This is the one and the only one required in computation of properties of diamond.

[4] the distance unit is in angstrom Å

[5] the time unit is in picoseconds

1.2.4. Data processing techniques

The simulation generates data with a preceding @ for every line and thus are re-formatted with the Linux bash command grep and sed, and then fitted with 3-order polynomial, we differentiate the acquired polynomial and find its zero points which designates the position the extreme point, which in this case is a minimum point and represents the equilibrium lattice constant value. This is done with the Matlab script Listing A.3 calLatticeConstant.m.

1.3. Young's Modulus

1.3.1. Definition

Young's modulus[6], also known as the tensile modulus or elastic modulus, is a measure of the stiffness of an elastic material and is a quantity used to characterize materials. It is defined as the ratio of the stress (force per unit area) along an axis over the strain (ratio of deformation over initial length) along that axis in the range of stress in which Hooke's law holds.

1.3.2. Method

Diamond belongs to the isometric-hexoctahedral crystal system (cubic system)[dia14], and the number of independent C_{ij} is 3. The stiff tensor[Kam06, JR07] combining stress and strain can be calculated in the same way as the example proposed in our lesson.

The elastic energy of the system can be signified by the formula 1.2.

$$E^{elas}/V = \frac{1}{2}\sum_{ij} C_{ij}\varepsilon_i\varepsilon_j \tag{1.2}$$

After several operations and simplification, we have formula 1.3

$$E = C_{11} = \frac{1}{V_c}\frac{\partial^2 E_{tot}}{\partial\gamma^2}|_{\gamma=0} = \frac{2c_2}{V_c} \tag{1.3}$$

where c_2 represents the coefficient of x^2 of the fitting polynomial, and V_c represents the volume of the whole system.

1.3.3. Simulation design

The boundaries are set to be periodic with units metal. The shape of the simulation region is a box (or block in terms), sized 20x20x20 and filled with "atomic" carbon atoms (so the "mass" is set to be equal to the molar mass of carbon, 12 grams/mol).

[6] represented by E in this article

The lattice style is set to be diamond with the lattice constant[7] being 3.567 according to the calculation results in sec. A.1.1 and the literature in Fig. 2.2. With the default time step[8] of 0.001, the thermodynamic info are printed every 100 time steps in the custom format "time step, box length in z axis, volume", from which we can get the volume of the whole system (which is V_c in formula 1.3).

The energy minimization process by iteratively adjusting atom coordinates are performed with the Polak-Ribiere version of the conjugate gradient (CG) algorithm, which repeats at most 1000 times.

Every time we acquire the local minimum potential energy (hopefully) of the box with the recently performed deformation (which makes the system be in equilibrium with the strain added to it) , we deform the simulation box with a extra strain, from its initial 0, to the final value[9] 0.3% along the z axis. After 21 loops of script[10], the strain will add up to 6% .

At last we will get the "Total energy - strain" data set, with polynomial fitting mechanism, we can get the coefficient of x^2, which is signified as c_2, and calculate the Young's modulus of diamond with the formula 1.3.

1.3.4. Data processing techniques

The simulation generates data with a preceding "Energy" keyword one line above the actual data, and thus are re-formatted the python script Listing A.5 ex02_process_data.py, and then fitted with 5-order polynomial, and get the desired coefficient, which is done with the python script Listing A.4 fittingPoly.py. To make sure the order of polynomial is proper, we perform an experiment to see the influence of the order to the final result with the python script Listing A.6 fittingPolyDegree.py .

[7] the distance unit is in angstrom Å

[8] the time unit is in picoseconds

[9] The strain: $\frac{216.9\times10^{-3}}{3.567\times20} = 0.3\%$

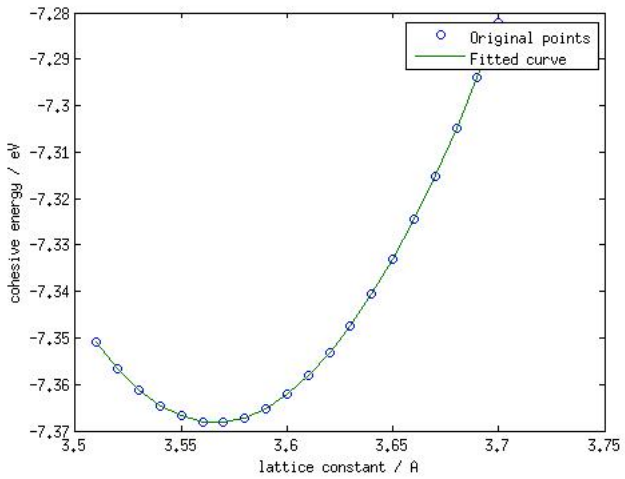
[10] while the extra strain is added only 20 times, since the counter starts at 1 and ends with 21

2. Results and discussion

2.1. Equilibrium Lattice Constant

With the shell command and the matlab script, the fitting curve is shown as Fig. 2.1.

Figure 2.1.: Fitting curve to the data.



The global minimum is retrieved at point (3. 5659, -7.3682).

So the equilibrium lattice constant of diamond is 3.5659 Å.

The commonly accepted value is 3.57 Å as suggested by Fig. 2.2.

So the result is quite satisfactory.

Figure 2.2.: Crystal structures of the elements

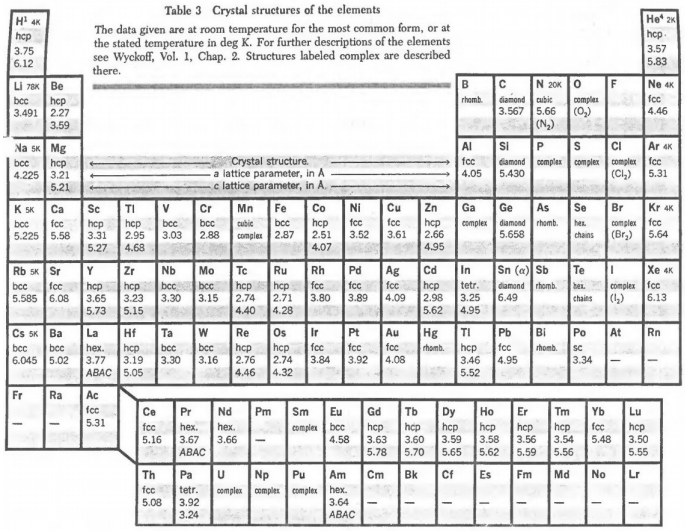


Table 3 Crystal structures of the elements

The data given are at room temperature for the most common form, or at the stated temperature in deg K. For further descriptions of the elements see Wyckoff, Vol. 1, Chap. 2. Structures labeled complex are described there.

Crystal structure
a lattice parameter, in Å
c lattice parameter, in Å

H^1 4K hcp 3.75 6.12																	He^4 2K hcp 3.57 5.83
Li 78K bcc 3.491	Be hcp 2.27 3.59											B rhomb.	C diamond 3.567	N 20K cubic 5.66 (N_2)	O complex (O_2)	F	Ne 4K fcc 4.46
Na 5K bcc 4.225	Mg hcp 3.21 5.21											Al fcc 4.05	Si diamond 5.430	P complex	S complex	Cl complex (Cl_2)	Ar 4K fcc 5.31
K 5K bcc 5.225	Ca fcc 5.58	Sc hcp 3.31 5.27	Ti hcp 2.95 4.68	V bcc 3.03	Cr bcc 2.88	Mn cubic complex	Fe bcc 2.87	Co hcp 2.51 4.07	Ni fcc 3.52	Cu fcc 3.61	Zn hcp 2.66 4.95	Ga complex	Ge diamond 5.658	As rhomb.	Se hex. chains	Br complex (Br_2)	Kr 4K fcc 5.64
Rb 5K bcc 5.585	Sr fcc 6.08	Y hcp 3.65 5.73	Zr hcp 3.23 5.15	Nb bcc 3.30	Mo bcc 3.15	Tc hcp 2.74 4.40	Ru hcp 2.71 4.28	Rh fcc 3.80	Pd fcc 3.89	Ag fcc 4.09	Cd hcp 2.98 5.62	In tetr. 3.25 4.95	Sn (α) diamond 6.49	Sb rhomb.	Te hex. chains	I complex (I_2)	Xe 4K fcc 6.13
Cs 5K bcc 6.045	Ba bcc 5.02	La hex. 3.77 *ABAC*	Hf hcp 3.19 5.05	Ta bcc 3.30	W bcc 3.16	Re hcp 2.76 4.46	Os hcp 2.74 4.32	Ir fcc 3.84	Pt fcc 3.92	Au fcc 4.08	Hg rhomb.	Tl hcp 3.46 5.52	Pb fcc 4.95	Bi rhomb.	Po sc 3.34	At —	Rn —
Fr —	Ra —	Ac fcc 5.31															

Ce fcc 5.16	Pr hex. 3.67 *ABAC*	Nd hex. 3.66	Pm —	Sm complex	Eu bcc 4.58	Gd hcp 3.63 5.78	Tb hcp 3.60 5.70	Dy hcp 3.59 5.65	Ho hcp 3.58 5.62	Er hcp 3.56 5.59	Tm hcp 3.54 5.56	Yb fcc 5.48	Lu hcp 3.50 5.55
Th fcc 5.08	Pa tetr. 3.92 3.24	U complex	Np complex	Pu complex	Am hex. 3.64 *ABAC*	Cm —	Bk —	Cf —	Es —	Fm —	Md —	No —	Lr —

2.2. Young's Modulus

With 5-degree polynomial, the fitting curve is shown in Fig. 2.3:

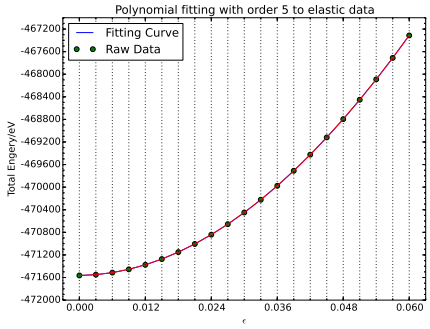


Figure 2.3.: Polynomial fitting with order 5 to elastic data.

With the coefficient of x^2, the Young's modulus can be calculated which is 1100.19107894 GPa. The value given by wikipedia is 1220 GPa and by Baidu ~1100GPa.

2.2.1. Discussion

We experimented with different orders of polynomials, the result is shown in Fig. 2.4.

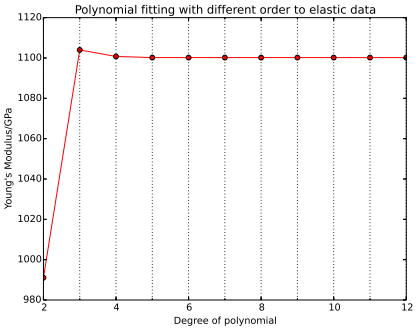


Figure 2.4.: Polynomial fitting with different order to elastic data.

As can be seen, the Young's modulus calculated is somewhat different from the experiment, especially when we compare it with the data supplied by Wikipedia[you14].

However, with the knowledge of the experiment in article [NBR+96], which fluctuates between we see that with different physical appearance (see table Tab.2.1), the Young's modulus is actually different. Considering that in our script, actually we are simulating a large bulk of carbon atoms in the diamond structure and with the periodic condition, we are essentially simulating the inner core elastic properties of diamond. So, the data given by Wikipedia may be measured with specimen in a different physical shape.

Table 2.1.: Sample details and modulus measured by the tensile test[NBR+96]

Table 2
Sample details and modulus measured by the tensile test

Sample type	Average diameter (μm)	Film thickness (μm)	Volume fraction diamond (%)	Young's modulus (GPa)
BP Sigma SiC				
1	102	—	—	362
2	98	—	—	357
W (125 μm diameter)/diamond				
1	205.3	40.15	62.9	892
2	204.2	39.6	62.5	810
3	204.2	39.6	62.5	713
4	196.8	35.9	59.7	772
5	206.0	40.5	63.2	789
6	205.4	40.2	63.0	684
7	203.0	39.0	62.1	668

Also the temperature is another factor that should be considered in analyzing the effectiveness of the result. By default the temperature of the system is 0 K, while most measures happens with room temperature, so this might be the reason of the difference.

A. Appendix

A.1. LAMMPS script

A.1.1. Equilibrium Lattice Constant

The LAMMPS script to generate raw data for Equilibrium Lattice Constant.

Listing A.1: homework.input

```
# Homework solution by Domicor (g1114128525@gmail.com)
# Settings
units           metal
boundary        p p p
atom_style      atomic
# Variables
variable        i loop 20
variable        x equal 3.5+0.01*$i # the lattice constant to test
# Build the model
lattice         diamond $x
region          box block 0 10 0 10 0 10
create_box      1 box
create_atoms    1 box
# Specify the potential
pair_style      tersoff
pair_coeff      * * SiC.tersoff C
mass            1 12
variable        n equal count(all)
variable        P equal pe/$n
timestep        0.005
thermo          10
# minimize the total energy
min_style       cg
minimize        1.0e-12 1.0e-12 1000 1000
print           "@ $x $P"
# loop
clear
next            i
jump            homework.input
```

A.1.2. Young's Modulus

To generate "cohesive energy-ε" data for later usage.

Listing A.2: homework_Young.input

```
# Homework solution by Domicor (g1114128525@gmail.com)
# Settings
units           metal
boundary        p p p
atom_style      atomic
# Build the model
lattice         diamond 3.567 # according to the table
region          box block 0 20 0 20 0 20
create_box      1 box
create_atoms    1 box
# Specify the potential
pair_style      tersoff
pair_coeff      * * SiC.tersoff C
mass            1 12
# set the format
thermo_style    custom step lz vol
thermo          100
# minimize the total energy
min_style       cg
label           begin
variable        t loop 21
fix             1 all deform 1 z delta 0.0 216.9E-3 units box
minimize        1.0e-6 1.0e-6 1000 10000
run             1 post no
unfix           1
# loop
next            t
jump            homework_Young.input begin
```

A.2. Data Processing Scripts

A.2.1. Matlab script for calculating the minimum point

Fit to the "cohesive energy-ε" data generated by Listing A.1 homework.input.

Listing A.3: calLatticeConstant.m

```
function calLatticeConstant(N, step, inFileName)
% calcuate the lattice constant according to "lat_const
    cohesive_energy"
```

```
3  % Input:
4  %   N: order of the polynomial fitting.
5  %   step: the fitting interval, if too small, the figure
       will be odd while the
6  %         numeric result may be correct.
7  %   inFileName: name of the file storing "lat_const
       cohesive_energy"
8  % Example:
9  %   calLatticeConstant(3,0.01,'lattice_02_processed.data')
10 %   Here, 'lattice_02_processed.data' is a file stores "
       lat_const cohesive_energy"
11 % The result is:
12 % lat_const = 3.5659
13 % coh_energy = -7.3682
14 % A snapshot of the figure is named lattice_2_processed.jpg
       .
15 % Originally Created by Xianbao Duan
16 % Email: xianbao.d@gmail.com
17 % Source: http://www.52souji.net/matlab-calculate-lattice-
       constant/
18 % Improved by Domicor (g1114128525@gmail.com) in Nov. 1,
       2014.
19 % read in data from the file
20 data = load(inFileName,'-ascii');
21 x = data(:,1);
22 y = data(:,2);
23 Ecoh = polyfit(x,y,N);
24 % polynomial fitting
25 dEcoh = polyder(Ecoh);       % derivation of the polynomial
       equation
26 zero_points = roots(dEcoh);      % solve the zero points
27 % calculate the lattice constant and coresponding cohesive
       energy
28 for i = 1: length(zero_points)
29     if isreal(zero_points(i))
30         if zero_points(i) > x(1)
31             if zero_points(i) < x(end)
32                 lat_const = zero_points(i)
33                 coh_energy = spline(x,y,lat_const)
34             end
35         end
36     end
37 end
38 % plot the curve
39 xx = x(1):step:x(end);
```

```
40 yy = polyval(Ecoh,xx);
41 figure;
42 plot(x,y,'o',xx,yy);
43 xlabel('lattice constant / A');
44 ylabel('cohesive energy / eV');
45 legend('Original points','Fitted curve');
```

A.2.2. Fit to the data of Young's modulus

The raw data is provided by Listing A.2 homework_Young.input.

Listing A.4: fittingPoly.py

```
1  # Curve fitting with polyfit
2  # to use this script, type:
3  # python fittingPoly.py ./log.lammps.output
4  from numpy import *
5  import pylab as pl
6  import sys
7  import matplotlib.pyplot as plt
8  import numpy as np
9  from matplotlib.ticker import MultipleLocator,
       FormatStrFormatter
10
11 filepath = sys.argv[1]
12 myfile = open(filepath)
13 xRange = arange(0, 0.063, 0.003)
14 xData = arange(0, 0.063, 0.003)
15 yData = xData
16 p0 = [197270, 100, 100]
17 i=0;
18 for line in myfile.readlines():
19      data = line.split()
20      #print(data)
21      yData[i] = data[0]
22      #print(yData[i])
23      i = i+1
24
25 plsq = np.polyfit(xRange,yData,5)
26 #plsq = P.polyfit(xData,yData,2)
27 #plsq = leastsq(y = a * x**2 + b * x + c, p0,args=(yData,
       xData))
28 print(plsq)
29 print 'the result C11 is', plsq[len(plsq)
       -3]*2*1000/(363077*6.2415), 'GPa'
```

```
30 fittingFunction = np.poly1d(plsq)
31 yFitting = fittingFunction(xRange)
32 #yFitting = P.polyval(xData, plsq)
33 #print(xData)
34 #print(xRange)
35 #print(yFitting)
36 plt.plot(xRange, yFitting, xRange, yData,'o')
37 plt.plot(xRange, yFitting)
38 plt.title('Polynomial fitting with order 5 to elastic data'
     )
39 plt.legend(['Fitting Curve', 'Raw Data'],'upper left')
40 plt.xlabel(r'$\epsilon$')
41 plt.ylabel('Total Engery/eV')
42 plt.xlim(-0.003,0.063)
43 #plt.ylim(-113350,-112600)
44 #plt.gca().set_yticks(arange(-113350,-112600, 100))
45 plt.gca().xaxis.set_minor_locator(MultipleLocator(0.003))
46 plt.gca().xaxis.set_major_locator(MultipleLocator(0.012))
47 plt.gca().yaxis.set_major_locator(MultipleLocator(400))
48 plt.gca().yaxis.set_major_formatter(FormatStrFormatter('%d'
     ))
49 plt.gca().yaxis.set_minor_locator(MultipleLocator(100))
50 plt.gca().xaxis.grid(True,which='minor')
51 plt.show()
52 #plt.savefig('fitting.jpg',dpi=300)
53 #plt.clf()
```

A.2.3. Extract data

Extract data in the log file generated by Listing A.2 homework_Young.input.

Listing A.5: ex02_process_data.py

```
1  import sys
2
3  def ex02_process_data(path):
4      """This is for extracting data from the log.lammps
5
6      Originally found in http://www.cnblogs.com/bacazy/
       archive/2013/09/28/3344687.html
7      Modified by Domicor (g1114128525@gmail.com) in Nov. 3,
       2014."""
8      b = 2
9      f = open(path);
10     fw = open(path + '.output', 'w')
```

```
11      for line in f.readlines():
12          b = b+1
13          if 'Energy initial' in str(line):
14              b = 0;
15          if b == 1:
16              print(line)
17              fw.write(line)
18
19  ex02_process_data(sys.argv[1])
```

A.2.4. Experiment with the order

Based on the observation in Listing A.4 fittingPoly.py, with some modification, experiment with the order to see the influence of it.

Listing A.6: fittingPolyDegree.py

```
1  # To verify the degree:
2  from numpy import *
3  import pylab as pl
4  import sys
5  import matplotlib.pyplot as plt
6  import numpy as np
7  from matplotlib.ticker import MultipleLocator,
       FormatStrFormatter
8  filepath = sys.argv[1]
9  myfile = open(filepath)
10 i=0;
11 xRange = arange(0, 0.063, 0.003)
12 xData = arange(0, 0.063, 0.003)
13 yData = xData
14 orderTest = [2,3,4,5,6,7,8,9,10,11,12]
15 C11 = []
16 for line in myfile.readlines():
17     data = line.split()
18     yData[i] = data[0]
19     i = i+1
20
21 i=0
22 for orderItem in orderTest:
23         plsq = np.polyfit(xRange,yData,orderItem)
24     print(plsq)
25     C11.append(plsq[len(plsq)-3]*2*1000/(363077*6.2415))
26     print 'the result C11 is', C11[i], 'GPa',      'with
           Degree',orderItem
```

```
27        i=i+1
28
29  plt.plot(orderTest, C11,'-or')
30  plt.title('Polynomial␣fitting␣with␣different␣order␣to␣
        elastic␣data')
31  plt.xlabel(r'Degree␣of␣polynomial')
32  plt.ylabel('Young\'s␣Modulus/GPa')
33  plt.gca().xaxis.set_minor_locator(MultipleLocator(1))
34  plt.gca().xaxis.grid(True,which='minor')
35  plt.show()
```

A.3. Sample experiment procedure

A.3.1. Equilibrium Lattice Constant

The procedure is:

1. cd to the directory storing the scripts, where the lammps executable is named lmp_serial_mine.
2. Input the script to lammps:

   ```
   ./lmp_serial_mine < homework.input
   ```

3. Use Linux bash command grep and sed to format raw data for Matlab fitting:

   ```
   grep ^@ log.lammps |sed 's/@ //g' > homework_processed.data
   ```

4. Launch Matlab and invoke the fitting script with the processed data:

   ```
   calLatticeConstant(3,0.01,'homework_processed.data')
   ```

The result is shown in sec.2.1 Equilibrium Lattice Constant.

A.3.2. Young's Modulus

The procedure is:

1. cd to the directory storing the scripts, where the lammps executable is named lmp_serial_mine.
2. Input the script to lammps:

   ```
   ./lmp_serial_mine < homework_Young.input
   ```

3. Use the python script Listing A.5 ex02_process_data.py to format raw data for the fitting:

```
python␣ex02_process_data.py␣'/home/domicor/Documents/
    code/compiled/examples_for_lmps/homework/log.lammps
    '
```

4. Use the python script Listing A.4 fittingPoly.py to fit:

```
python␣fittingPoly.py␣'/home/domicor/Documents/code/
    compiled/examples_for_lmps/homework/log.lammps.
    output'
```

5. Use the python script Listing A.6 fittingPolyDegree.py to test the influence of order of polynomials:

```
python␣fittingPolyDegree.py␣'/home/domicor/Documents/
    code/compiled/examples_for_lmps/homework/log.lammps
    .output'
```

The result is shown in sec.2.2 Young's Modulus.

Bibliography

[dia14] Diamond, wikipedia, October 2014. Page Version ID: 626930953.

[JR07] Harald. Harders Joachim Rosler, Martin. Baker. *Mechanical Behaviour of Engineering Materials: Metals, Ceramics, Polymers, and Composites*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007.

[Kam06] M. M. Kaminski. *Computational Mechanics of Composite Materials : Sensitivity, Randomness and Multiscale Behaviour*. Springer, Dordrecht, 2006.

[lat14] Lattice constant, wikipedia, August 2014. Page Version ID: 608331410.

[NBR+96] E.D. Nicholson, T.W. Baker, S.A. Redman, E. Kalaugher, K.N. Rosser, N.M. Everitt, M.N.R. Ashfold, and P.G. Partridge. Young's modulus of diamond-coated fibres and wires. *Diamond and Related Materials*, 5(6-8):658 – 663, 1996. Proceedings of the 6th European Conference on Diamond, Diamond-like and Related Materials Part 2.

[Ter88] J. Tersoff. New empirical approach for the structure and energy of covalent systems. *Physical Review B*, 37(12):6991–7000, April 1988.

[Ter89] J. Tersoff. Modeling solid-state chemistry: Interatomic potentials for multicomponent systems. *Physical Review B*, 39(8):5566–5568, March 1989.

[Ter90] J. Tersoff. Erratum: Modeling solid-state chemistry: Interatomic potentials for multicomponent systems. *Physical Review B*, 41(5):3248–3248, February 1990.

[you14] Young's modulus, wikipedia, September 2014. Page Version ID: 625853216.