Pymatsci

Release 1.0.0

C.L. Qin

Oct 02, 2022

CONTENTS

Pymatsci (Python Materials Science) is a robust, open-source Python library for materials analysis. It is further packaged and developed on the basis of **Pymatgen**, a very powerful package of material analysis. Pymatsci is dedicated to quick and easy operations, so that people can get the results they want without taking too much time.

Check out the *Usage* section for further information, including how to *Installation* the project. Of course, you can also click here for a pdf instruction.

Note: Although the current project is very crude, it is under active development.

CHAPTER

ONE

CONTENTS

1.1 Usage

1.1.1 Installation

1.To use pymatsci, first install pymatgen and numpy using pip:

```
pip install numpy
pip install pymatgen
```

2.Install pymatsci

```
pip install pymatsci
```

In addition, you can also visit pypi to download.

1.1.2 Tutorials

This page presents a collection of tutorials.

1. Model

This page provides a series of tutorials on nanotube, graphene, and magic graphene modeling.

1.1 Structural information of the model

Both nanotubes and magic-angle graphene are built from single-layer graphene, and they have the same definition of chirality (n, m).

Graphene

Total number of atoms(N):

$$N = \frac{2(n^2 + m^2 + mn)p}{g_1}$$

where g_1 is the greatest common divisor of n+2m and 2n+m, and p is the period.

Magic-angle graphene

Total number of atoms(N):

$$N = \frac{4(n^2 + m^2 + mn)p}{g_1}$$

Magic angle():

$$\theta = ar\cos\frac{m^2 + n^2 + 4mn}{2\sqrt{m^2 + n^2 + mn}}$$

Nanotube

Total number of atoms(N):

$$N = \frac{2(n^2 + m^2 + mn)p}{g_1}$$

Diameter(d):

$$d = \frac{\sqrt{m^2 + n^2 + mn}}{\pi}\sqrt{3}b$$

where b is the atomic bond length.

Total length(l):

$$l = \frac{3b\sqrt{m^2 + n^2 + mn}}{g_1}$$

Chirality angle():

$$\theta = ar\cos(\frac{2n + m}{2\sqrt{m^2 + n^2 + mn}})$$

[1] E.J. Mele, Interlayer coupling in rotationally faulted multilayer graphenes, J Phys D Appl Phys 45 (15) (2012).

[2] P. Moon, M. Koshino, Optical absorption in twisted bilayer graphene, Phys. Rev. B 87 (20) (2013).

[3] S. Shallcross, S. Sharma, E. Kandelaki, O.A. Pankratov, Electronic structure of turbostratic graphene, Phys. Rev. B 81 (16) (2010).

[4] M.S. Dresselhaus, G. Dresselhaus, R. Saito, Carbon fibers based on C60 and their symmetry, Phys Rev B Condens Matter 45 (11) (1992) 6234-6242.

[5] C.T. White, D.H. Robertson, J.W. Mintmire, Helical and rotational symmetries of nanoscale graphitic tubules, Phys Rev B Condens Matter 47 (9) (1993) 5485-5488.

1.2 Graphene

Input

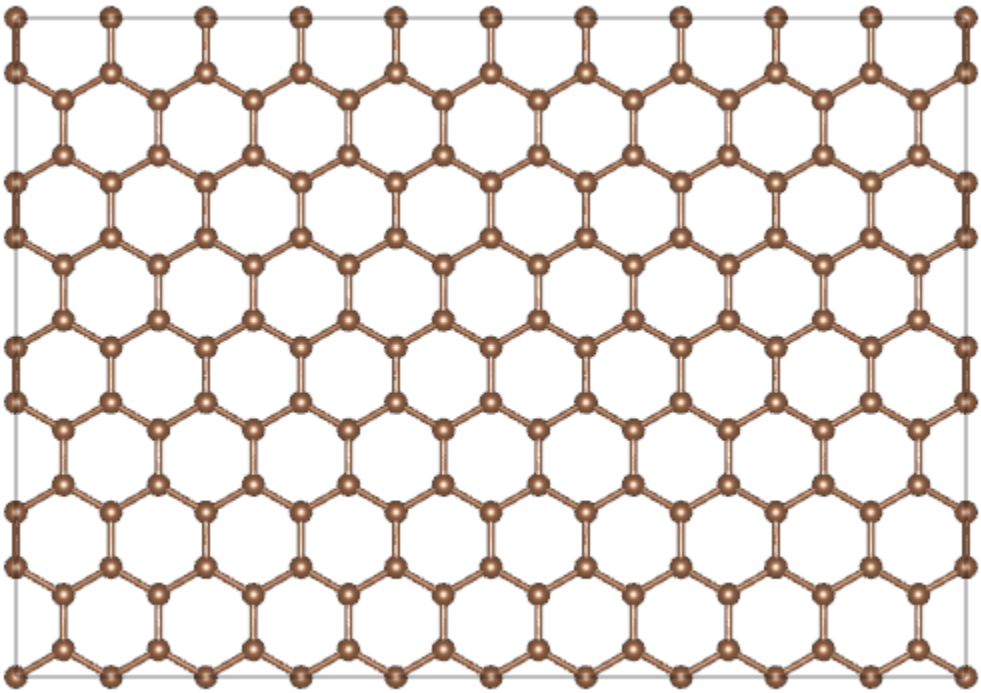
```
from pymatsci.model import Graphene          #
model = Graphene(10, 0, 1.42, ['C'], 4)      #
model.write_vasp('./POSCAR')                  # vasp
# model.write_lammps('./data.txt')           # lammps
```

Output

Console:

Generated model:

```
*******************************************************
*   Author: C.L. Qin    E-mail: clqin@foxmail.com   *
*******************************************************
The file has been successfully written!
------------>
-------------------------------------------------------
Total number of atoms   :   160
Lattice                 :   [24.595121467478055, 0, 0, 0, 17.04, 0, 0, 0, 10]
-------------------------------------------------------
```



1.3 Magic-angle graphene

Input

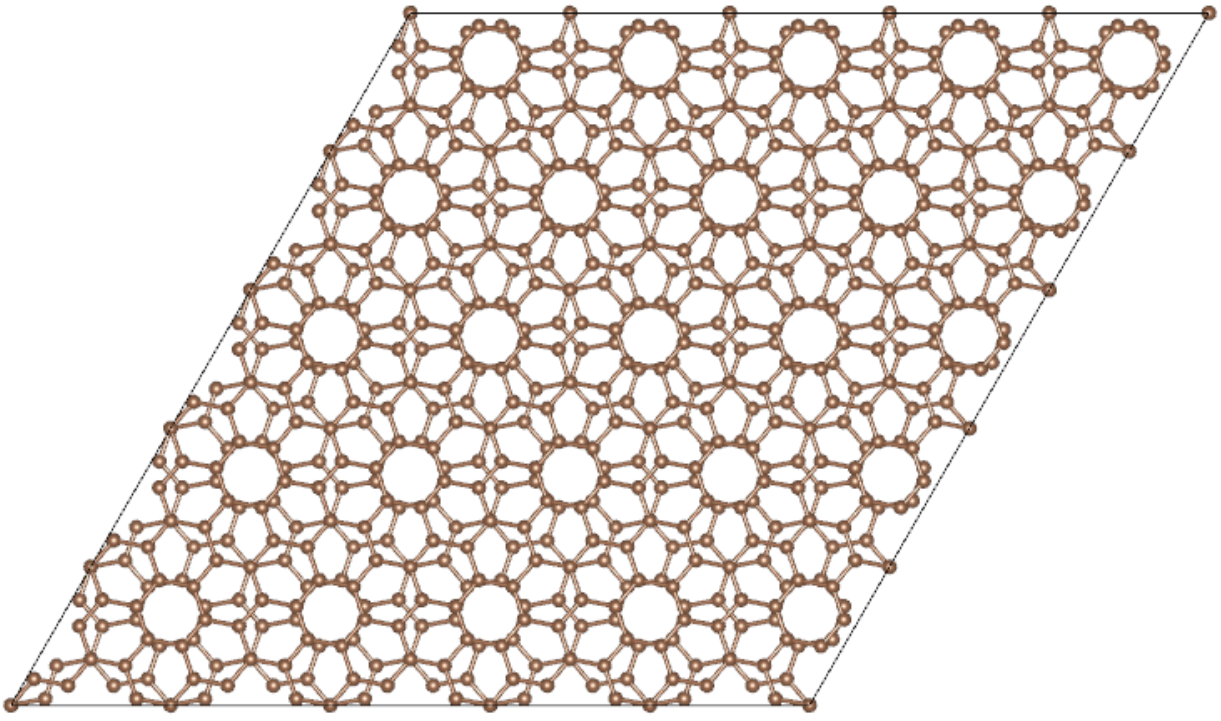
```
from pymatsci.model import MagicGraphene          #
model = MagicGraphene(10, 5, 1.42, ['C'], 3.4)    #
model.write_vasp('./POSCAR')
# model.write_lammps('./data.txt')
```

Output

Console:

```
**************************************************
*   Author: C.L. Qin    E-mail: clqin@foxmail.com   *
**************************************************
The file has been successfully written!
------------>
--------------------------------------------------
Total number of atoms   :   1400
Chirality               :   (10, 5)
Magic angle             :   0.38025120669293344
Lattice                 :   [30.743901834347568, 10.649999999999999, 0, 6.148780366869518, 31.949999999999996, 0, 0, 0, 15]
--------------------------------------------------
```

Generated model:



1.4 Nanotube

Input

```
from pymatsci.model import Nanotube        #
model = Nanotube(10, 5, 1.42, ['C'], 5)    #
model.write_vasp('./POSCAR')
# model.write_lammps('./data.txt')
```

Output

Console:

```
**************************************************
*   Author: C.L. Qin    E-mail: clqin@foxmail.com   *
**************************************************
The file has been successfully written!
------------->
--------------------------------------------------
Diameter                :   10.356621950018992
Total number of atoms   :   700
Total length            :   56.35450292567577
Chirality angle         :   19.1066053508691
Period                  :   5
Chirality               :   (10, 5)
Lattice                 :   [20.35662195001899, 0, 0, 0, 20.35662195001899, 0, 0, 0, 56.35450292567577]
--------------------------------------------------
```

Generated model:

2. Thermodynamics

2.1 Thermodynamic correction

2.1.1 Theory

Pymatsci uses a thermal correction similar to Gaussian, and the detailed thermodynamic derivation can be found in Atkins' Physical Chemistry. The gases are assumed to be indistinguishable perfect gases with no interactions between them. The expressions for the internal energy (U) and entropy (S) of N molecules can be known from statistical thermodynamics:

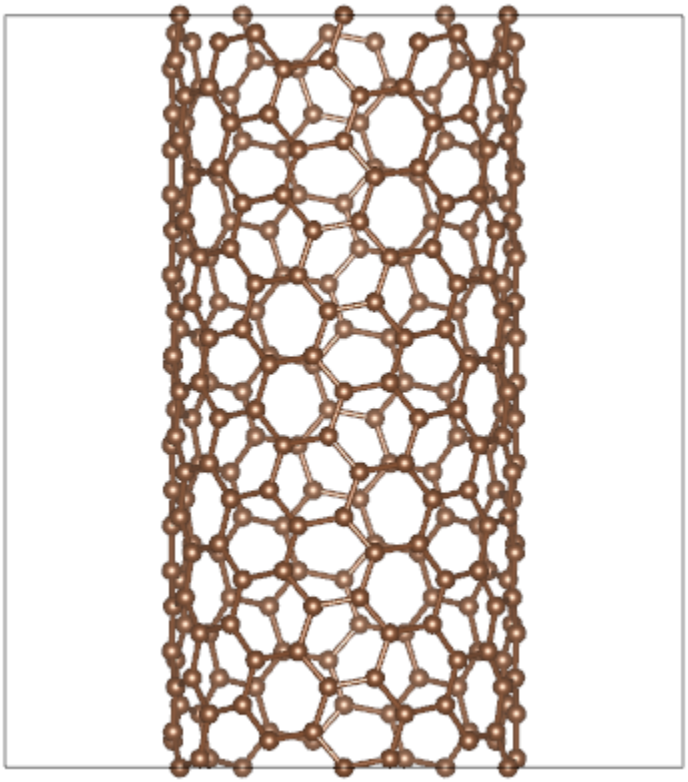
$$U = NkT^2\left(\frac{\partial \ln q}{\partial T}\right)_V$$

$$S = NkT\left(\frac{\partial \ln q}{\partial T}\right)_V + Nk\ln\frac{q}{N} + kN$$

where T is the temperature, V is the volume of the container, q is the partition function, and k is the Boltzmann constant. When considering only one molecule:

$$U = kT^2\left(\frac{\partial \ln q}{\partial T}\right)_V$$

$$S = kT\left(\frac{\partial \ln q}{\partial T}\right)_V + k\ln q + k$$



Considering the translational (q^t), rotational (q^r), vibrational (q^v), and electron (q^e) contributions we get

$$q = q^t q^r q^v q^e$$

Therefore,

$$U = \sum_i kT^2 \left(\frac{\partial \ln q_i}{\partial T} \right)_V = U_t + U_r + U_v + U_e$$

$$S = \sum_i kT \left(\frac{\partial \ln q_i}{\partial T} \right)_V + k \ln q_i + k = S_t + S_r + S_v + S_e + k$$

Enthalpy (H) and Gibbs free energy (G) can be obtained from U and S:

$$H = U + pV$$

$$G = H - TS$$

The expression for H seems to be a little trickier, since we don't know V. However, for an perfect gas, pV=kT. Therefore,

$$H = U + kT$$

The translational partition function is:

$$q^t = \frac{(2\pi mkT)^{3/2}}{h^3} V = \frac{(2\pi mkT)^{3/2}}{h^3} \frac{kT}{p}$$

where h is Planck's constant and m is the molecular mass.

The rotational partition function of a linear molecule ($I_x = 0$, $I_y = I_z$) is

$$q^r = \frac{T}{\sigma\theta^r}$$

$$\theta^r = \frac{h^2}{8\pi^2 Ik}$$

where is the symmetry number, and I ($I = I_y$) is the moment of inertia. The rotational partition function of the nonlinear molecule (I_x, I_y, I_z != 0) is

$$q^r = \frac{\pi^{1/2}}{\sigma}\left(\frac{T^{3/2}}{(\theta_x^r\theta_y^r\theta_z^r)^{1/2}}\right)$$

The vibration partition function is

$$q^v = \prod_f \frac{1}{1 - e^{-hv/kT}}$$

where v is the vibration frequency and f is the vibrational degrees of freedom of the molecule.

Electrons are generally in the ground state, so the partition function is

$$q^e = g$$

where g is the degeneracy of the electron ground state, or spin multiplicity.

Zero-point energy ($z=z^t + z^v$) comes from translational (z^t) and vibrational (z^v) contributions. However, the zero-point energy of the translational contribution is negligible, so

$$z \approx z^v = \prod_f 1/2hv$$

At last, pymatsci will output the corrected DFT energy ($E_{DFT,c}$), internal energy (U_c), enthalpy (H_c) and Gibbs free (G_c),

$$E_{DFT,c} = E_{DFT} + z$$

$$U_c = E_{DFT,c} + U$$

$$H_c = E_{DFT,c} + H$$

$$G_c = E_{DFT,c} + G$$

where E_{DFT} is the energy of density functional calculations.

[1] P. Atkins, J. De Paula, J. Keeler, Physical Chemistry, 11 ed., Oxford University Press, London, 2018.

[2] https://gaussian.com/thermo/

[3] V. Wang, N. Xu, J.C. Liu, G. Tang, W.T. Geng, VASPKIT: A User-Friendly Interface Facilitating High-Throughput Computing and Analysis Using VASP Code, Computer Physics Communications 267, 108033 (2021).

2.1.1 Free gas

For free gases, consider translational, rotational, vibrational and electron contributions. For linear molecules, degree of vibrational freedom is 3n - 5, Pymatsci will neglect smallest 5 frequencies. For non-linear molecules, degree of vibrational freedom is 3n - 6,Pymatsci will neglect smallest 6 frequencies. n is the atomic number of the molecule.

Input

First you need to put CONTCAR and OUTCAR in the current folder.

```
from pymatsci.thermodynamics import FreeGasCorrection   #
t = FreeGasCorrection(298.15, 101325, True, 3)      # (K)(Pa)
t.correction()                                  #
t.printout()                                            #
```

Output

```
*******************************************************
*   Author: C.L. Qin    E-mail: clqin@foxmail.com   *
*******************************************************
-------------------------------------------------------
Temperature (T)          :   298.15 K
Pressure (P)             :   101325.0 Pa
Linear molecule          :   True
Spin-Multiplicity (S)    :   3
Point group              :   D*h
symmetry number          :   2
-------------------------------------------------------
-------------------------------------------------------
Energy of DFT (E_DFT)        :   -8.66536423 eV
Zero-point energy (E_ZPE)    :   0.10414957298177664 eV
Thermal correction to U(T)   :   0.16844217313220516 eV
Thermal correction to H(T)   :   0.19413475225329102 eV
Thermal correction to G(T)   :   -0.4385824570924133 eV
Entropy S                    :   0.0021221439186506937 eV K^-1
Entropy contribution T*S     :   0.6327172093457043 eV
-------------------------------------------------------
-------------------------------------------------------
corrected E_DFT :    -8.561214657018223 eV
Corrected U(T)  :    -8.392772483886018 eV
Corrected H(T)  :    -8.367079904764932 eV
Corrected G(T)  :    -8.999797114110637 eV
-------------------------------------------------------
```

2.1.2 Adsorbed gas

For adsorbed molecules, pymatsci uses the calculation method of vaspkit. Unlike gas molecules, adsorbed molecules form chemical bonds with substrate, which limits the translational and rotational freedom of the adsorbed molecules. So the contribution of translation and rotation to entropy and enthalpy is significantly reduced (so called hindered translator / hindered rotor model). This does not mean no translational or rotational contribution.

One common method is to attribute the translational or rotational part of the contribution to vibration, that is, the 3n vibrations of the surface-adsorbing molecules (except the virtual frequency) are all used to calculate the correction of the thermo energy. Pymatsci neglects the electron motion because of its small contribution and pV can be ignored in condensed phase. Therefore,

$$H = U = U_v$$

$$S = S_v + k$$

The small the vibration frequencies have large contribution to entropy. It is very likely that a small vibration frequency will lead to abnormal entropy and free energy correction. So, it suggests that when the free energy of the surface adsorption molecule is corrected, the contribution of the frequency below 50 cm^{-1} is calculated as 50 cm^{-1}, and pymatsci also does this.

Input

First fix all slab atoms, do frequency calculation for the adsorbed molecule.

Then, you need to put CONTCAR and OUTCAR in the current folder.

```
from pymatsci.thermodynamics import AdsorbedGasCorrection
a = AdsorbedGasCorrection(298.15)                    # (K)
a.correction()                                       #
a.printout()                                         #
```

Output

2.2 Ab initio thermodynamics

2.2.1 Theory

The energy calculated by DFT only gives the energy at zero temperature (T) and zero pressure (p). However, for the real reaction conditions, the interaction of O2 molecules with the U surface requires the consideration of temperature and O2 pressure, which is achieved by using the "ab initio thermodynamics" approach. The Gibbs free energy of adsorption considering temperature and pressure can be described as

$$G_{\mathrm{ads}} = G_{adsorbates/slab} - G_{slab} - \mathrm{n}G_{adsorbates}(1)$$

where n is the number of adsorbates, Gadsorbates/slab is the Gibbs free energy of the adsorbates/slab system, Gslab is the Gibbs free energy of the slab, and Gadsorbates is the Gibbs free energy of the adsorbates. The energy calculated by DFT (E) is related to a thermodynamical quantity only in a restricted way, corresponding to the Gibbs free energy at zero temperature and neglecting zero-point vibrations. It is known from physical chemistry that Gibbs free energy can thus be written as

$$G(T, \mathrm{p}) = E + U(T, \mathrm{p}) + pV - TS(T, p)(2)$$

In general, the volume effect and the vibration contribution to the Gibbs free energy of the condensed phase are small. On the other hand, for adsorption systems, the adsorption of gas molecules will not significantly change the vibrational

```
******************************************************
*   Author: C.L. Qin    E-mail: clqin@foxmail.com    *
******************************************************
------------------------------------------------------
Temperature (T) :   298.15 K
------------------------------------------------------
------------------------------------------------------
Energy of DFT (E_DFT)       :   -213.96118341 eV
Zero-point energy (E_ZPE)   :   0.10808415953689414 eV
Thermal correction to U(T)  :   0.1580819781603639 eV
Thermal correction to H(T)  :   0.1580819781603639 eV
Thermal correction to G(T)  :   0.05069512991977236 eV
Entropy S                   :   0.000360177253867488 eV K^-1
Entropy contribution T*S    :   0.10738684824059154 eV
------------------------------------------------------
------------------------------------------------------
corrected E_DFT :   -213.8530992504631 eV
Corrected U(T)  :   -213.69501727230272 eV
Corrected H(T)  :   -213.69501727230272 eV
Corrected G(T)  :   -213.80240412054332 eV
------------------------------------------------------
```

modes of the surface, and the contributions from temperature and pressure will be canceled. Therefore, equation (3) can be written as

$$G_{\mathrm{ads}} = E_{adsorbates/slab} - E_{slab} - \mathrm{n}G_{adsorbates} (3)$$

Here is an example of oxygen molecule. The adsorbates considered are O atoms whose chemical potential is half that of gaseous O2 molecules. Assuming that the adsorbed gas is the perfect molecule, the chemical potential of the O atom can be expressed as

$$\mu_O = \frac{1}{2}\mu_{\mathrm{O}_2}(T,p) = \frac{1}{2}\left(E_{O_2} + \mu_{\mathrm{O}_2}(T,p^\theta) + kT\ln\left(\frac{p_{O_2}}{p^\theta}\right)\right) (4)$$

where k is the Boltzmann constant. Equation (3) can be further written as

$$\begin{aligned} G_{\mathrm{ads}} &= E_{O/slab} - E_{slab} - n\mu_O \\ &= E_{O/slab} - E_{slab} - \tfrac{n}{2}E_{O_2} - \tfrac{\mathrm{n}}{2}\left(\mu_{O_2}(T,p^\theta) + kT\ln(\tfrac{p_{O_2}}{p^\theta})\right) \\ &= E_{ads} - \tfrac{\mathrm{n}}{2}\left(\mu_{O_2}(T,p^\theta) + kT\ln(\tfrac{p_{O_2}}{p^\theta})\right) \end{aligned} (7)$$

where Eads is the adsorption energy at 0 K.

[1] K. Reuter, M. Scheffler, Composition, structure, and stability ofRuO2(110)as a function of oxygen pressure, Phys. Rev. B 65 (3) (2001).

[2] P. Maldonado, L.Z. Evins, P.M. Oppeneer, Ab Initio Atomistic Thermodynamics of Water Reacting with Uranium Dioxide Surfaces, The Journal of Physical Chemistry C 118 (16) (2014) 8491-8500.

2.2.2 Phase Diagrams (T-p)

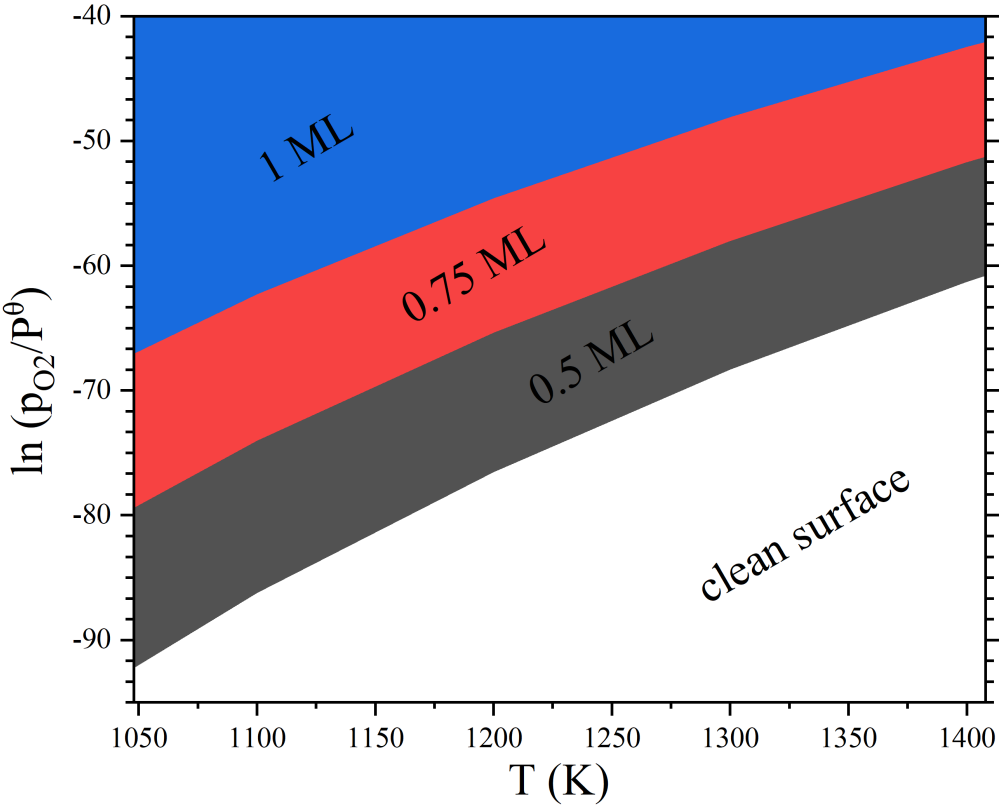
```
from pymatsci.thermodynamics import Abthermodynamics
import numpy as np
t = Abthermodynamics('https://janaf.nist.gov/tables/O-029.html')
#
data1 = t.get_Tp(3/2, -16.351, 4/2, -18.881)  #
data2 = t.get_Tp(3 / 2, -16.351, 0, 0)  #
data = np.hstack((data1, data2))
# np.savetxt('data.txt', data)  #
```

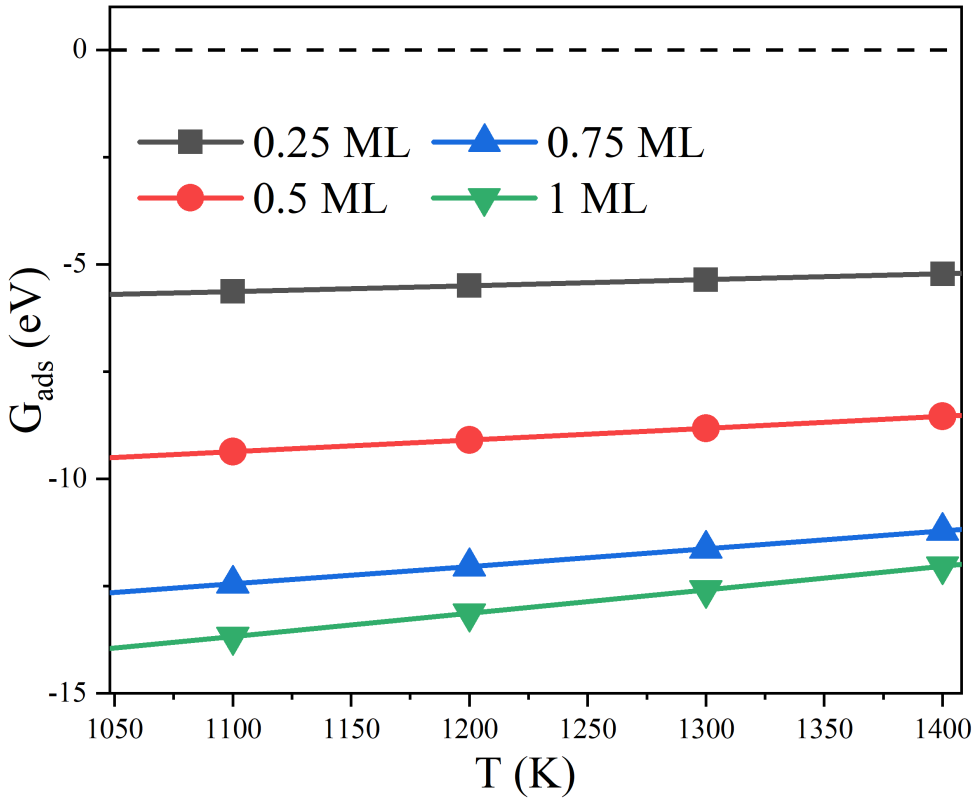
Output

2.2.3 Phase Diagrams (T)

```
from pymatsci.thermodynamics import Abthermodynamics
import numpy as np
t = Abthermodynamics('https://janaf.nist.gov/tables/O-029.html')
data1 = t.get_T(1/2, -5.134, 0.21)  #
data2 = t.get_T(2/2, -10.652, 0.21)  #
data = np.hstack((data1, data2))
# np.savetxt('data.txt', data)  #
```

Output


-40
-50
-60
-70
-80
-90
1 ML
0.75 ML
0.5 ML
clean surface
ln (p_{O2}/P^{θ})
1050
1100
1150
1200
1250
1300
1350
1400
T (K)


0
-5
-10
-15
0.25 ML
0.75 ML
0.5 ML
1 ML
G_{ads} (eV)
1050
1100
1150
1200
1250
1300
1350
1400
T (K)

2.2.4 Phase Diagrams (p)

```
from pymatsci.thermodynamics import Abthermodynamics
import numpy as np
t = Abthermodynamics('https://janaf.nist.gov/tables/O-029.html')
data1 = t.get_p(1/2, -5.134, 300)  #
data2 = t.get_p(2/2, -10.652, 300)  #
data = np.hstack((data1, data2))
print(data)  #
```

Output

```
['-4.860063605933603-(0.012925999893217768*T)'
 '-10.104127211867205-(0.025851999786435535*T)']
```

3. Surface

This page provides a series of tutorials on surface and adsorption.

3.1 Structural information of surface

Input

First you need to provide POSCAR and CONTCAR.

```
from pymatsci.surface import get_surf_info
get_surf_info([21, 22])   #
```

Output

```
**************************************************
*   Author: C.L. Qin    E-mail: clqin@foxmail.com   *
**************************************************
--------------------------------------------------
layer_num          :   5
constrain_layer_num :   2
layer_spacing      :   [2.2748445159604547, 2.559939595942317, 2.453527075159035, 2.4668063429998406]
relation           :   [-7.781809452280654, 3.775474236703217, -0.5283071159959762, -6.4809349951481996e-12]
--------------------------------------------------
```

3.2 Adsorption potential energy surface

3.2.1 Generate model file

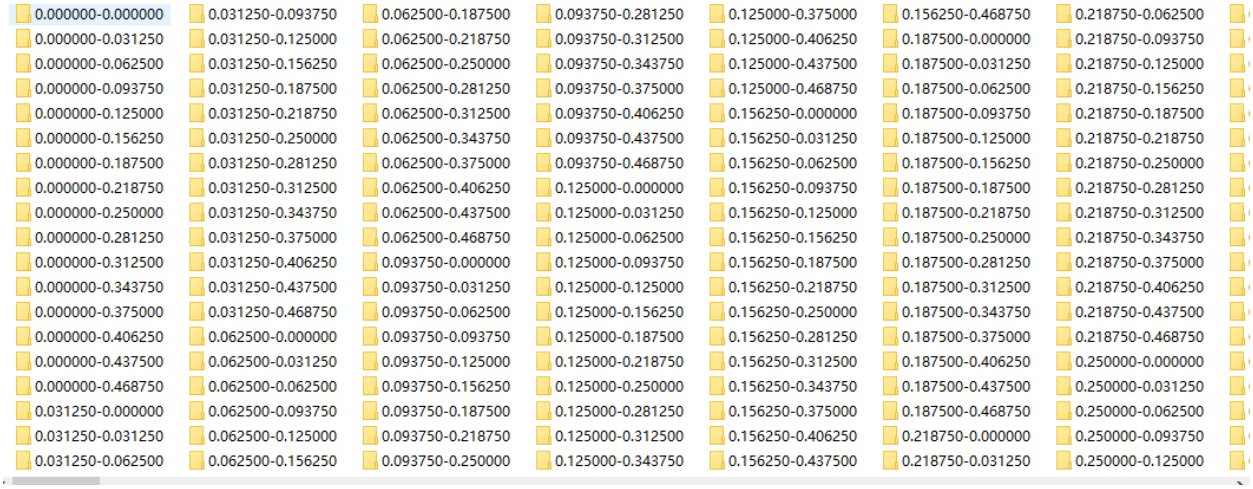
Input

First create a folder named PES in the current folder.

Second, you need to provide POSCAR, INCAR, POTCAR, KPOINTS and the submission file vasp.pbs in the PES folder.

```
from pymatsci.surface import generate_grid
# xyVESTA
generate_grid('vasp.pbs', 2, 2, 16, 16, 21)
```

Output



Finally, you can drag the created folder into the server for calculation.

3.2.2 Extract data

First make sure the PES folder is empty, then drag the calculated data folder from the server into the PES folder.

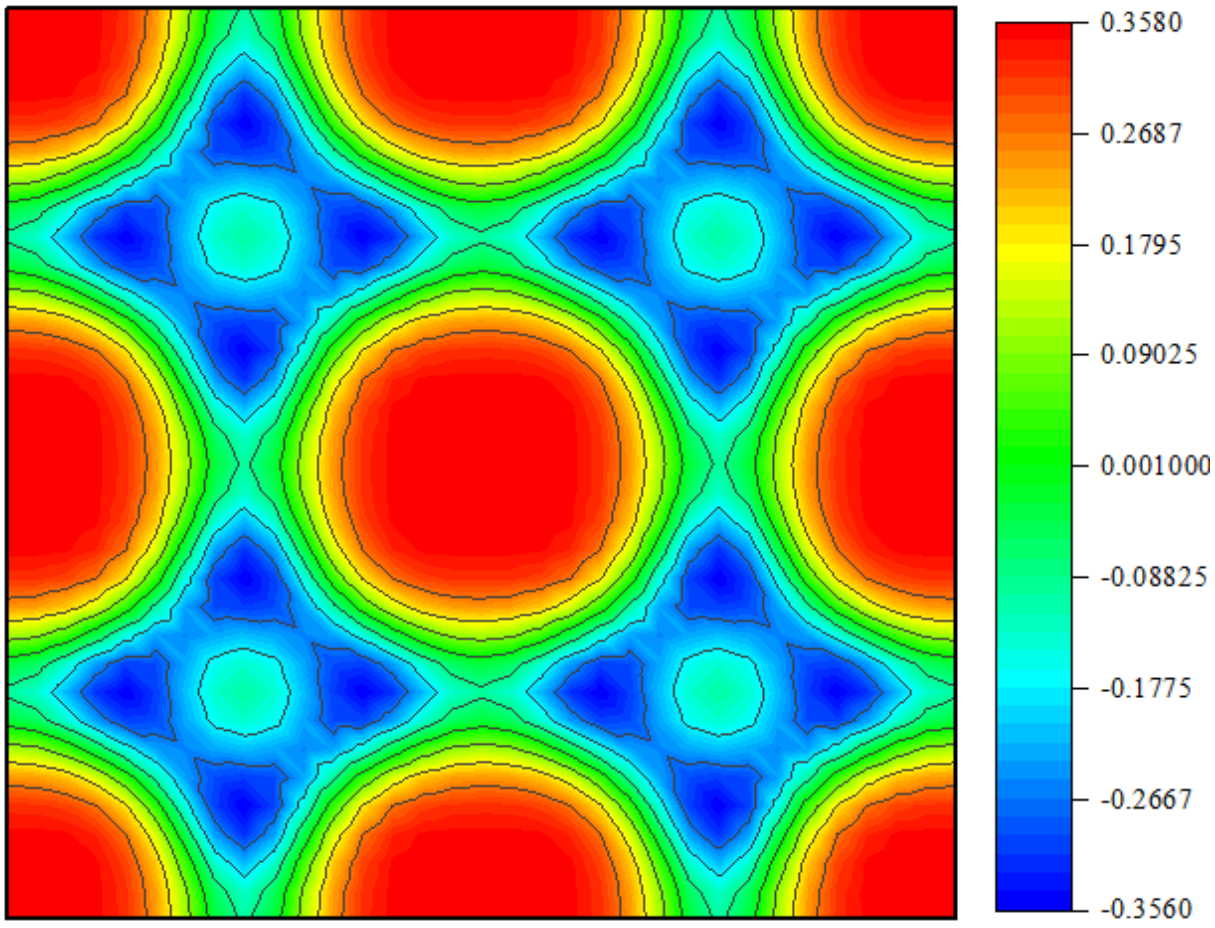
```
from pymatsci.surface import extract_data
#  xy
extract_data(2, 2, -215.8)
```

Then a data file named data.txt will be generated in the current folderand drag it into origin to draw

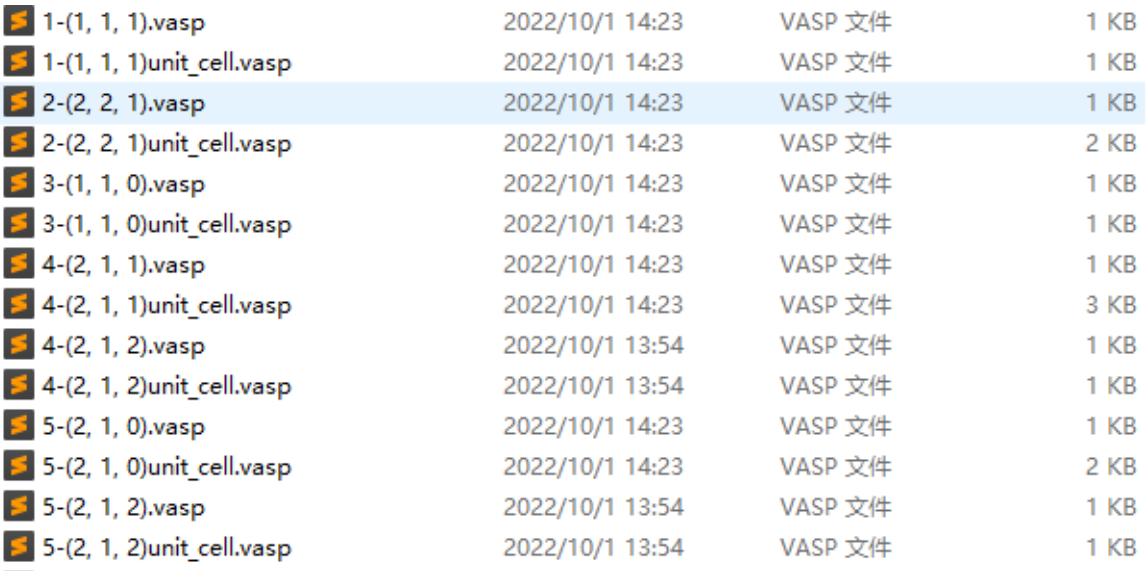
3.3 Generate all surfaces with a given maximum Miller index

```
from pymatsci.surface import generate_all_surfs
#  miller_index
generate_all_surfs('CONTCAR', 2, 1, 15, write_surf=True, write_unit_cell=True)
```

Output


0.3580
0.2687
0.1795
0.09025
0.001000
-0.08825
-0.1775
-0.2667
-0.3560

```
*****************************************************
*   Author: C.L. Qin    E-mail: clqin@foxmail.com   *
*****************************************************
最大miller_index: 2
表面总数: 6
所有表面如下:
1
-----------------------------------------------------
layer_num           :   2
constrain_layer_num :   0
layer_spacing       :   [1.4241839983483198]
miller_index        :   (1, 1, 1)
-----------------------------------------------------
2
-----------------------------------------------------
layer_num           :   4
constrain_layer_num :   0
layer_spacing       :   [2.4667590445658227, 2.46675904436594, 2.4667590445658223]
miller_index        :   (2, 2, 1)
-----------------------------------------------------
3
```



1-(1, 1, 1).vasp	2022/10/1 14:23	VASP 文件	1 KB
1-(1, 1, 1)unit_cell.vasp	2022/10/1 14:23	VASP 文件	1 KB
2-(2, 2, 1).vasp	2022/10/1 14:23	VASP 文件	1 KB
2-(2, 2, 1)unit_cell.vasp	2022/10/1 14:23	VASP 文件	2 KB
3-(1, 1, 0).vasp	2022/10/1 14:23	VASP 文件	1 KB
3-(1, 1, 0)unit_cell.vasp	2022/10/1 14:23	VASP 文件	1 KB
4-(2, 1, 1).vasp	2022/10/1 14:23	VASP 文件	1 KB
4-(2, 1, 1)unit_cell.vasp	2022/10/1 14:23	VASP 文件	3 KB
4-(2, 1, 2).vasp	2022/10/1 13:54	VASP 文件	1 KB
4-(2, 1, 2)unit_cell.vasp	2022/10/1 13:54	VASP 文件	1 KB
5-(2, 1, 0).vasp	2022/10/1 14:23	VASP 文件	1 KB
5-(2, 1, 0)unit_cell.vasp	2022/10/1 14:23	VASP 文件	2 KB
5-(2, 1, 2).vasp	2022/10/1 13:54	VASP 文件	1 KB
5-(2, 1, 2)unit_cell.vasp	2022/10/1 13:54	VASP 文件	1 KB

1.2 API

1.3 Topic

pymatgen

```
from pymatgen.core.surface import generate_all_slabs
from pymatgen.core.structure import Structure
from pymatgen.io.vasp import Poscar
structure = Structure.from_file('CONTCAR')   #
slabs = generate_all_slabs(structure, 2, 1, 0.1, max_normal_search=2)  # miller_indexmin_
↪slab_size min_vacuum_size
index = 0
for slab in slabs:
    print(slab.miller_index)
    poscar = Poscar(slab, comment=str(slab.miller_index))
    poscar.write_file(str(index)+str(slab.miller_index)+'.vasp', direct=False)
    index += 1
```

pymatgenASE

```
from pymatgen.io.ase import AseAtomsAdaptor
structure = Structure.from_file('CONTCAR')
atoms = adaptor.get_atoms(slab)
structure = read('CONTCAR')
```

ASE

```
structure = read('CONTCAR')
slab = surface(structure, [1, 1, 1], 1, vacuum=2)
vaccum = 15
slab.cell[2, 2] = slab.cell[2, 2] + vaccum - 2*2
# print(slab.get_positions())
coords = slab.get_positions()
z = np.around(coords[:, 2], 5)
unique_z = np.unique(z, axis=0)
tags_list = []
index = 1
for j in range(len(z)):
    for i in range(len(unique_z)):
        if z[j] == unique_z[i]:
            tags_list.append(i+1)
slab.set_tags(tags_list)
layers_num = len(unique_z)
print(layers_num)
write('POSCAR', slab)
```