High temperature and anharmonic phonon modes for unstable structure

2019.12.16

support@pwmat.com
Technical Note

In order to run this module, we need to run the PyPWmat module.
So please get familiar with that module.
The problem

❖ It is well known some of the high symmetry phase are unstable at low temperature (T=0).

❖ Nevertheless, by enforcing symmetry, in the relax calculation, we can relax to such high symmetry structure (at T=0).

❖ However, if we calculate the phonon spectrum, we will find imaginary phonon modes, which can be at both Gamma point and off Gamma point (such imaginary modes are not due to convergence error in the calculations).

❖ But we also know, experimentally, at higher temperature T, such structure is stable (presumably due to entropy contribution).

❖ One can do a high T MD to show such structure is stable.

❖ Here, we implement a procedure to calculate the phonon at high T, and its corresponding entropy term.
The basic idea (1)

❖ We follow the method in the paper:

❖ The original energy curve is a, the symmetry point is unstable.

❖ But, at high temperature, the vibration amplitude is large, so effectively, it can reach points like C and D.

❖ If we judge from the energy (or the force) at C and D, and draw a parabolic curve (the dashed lines c and d), then they can be considered as having positive phonon frequency.

❖ At a given temperature, if the oscillation amplitude is \(d_C\) (the C curve), then the frequency is \(\omega_C\). For a given \(\omega_C\) and T, we can come back to estimate the \(d_C\) (each degree of freedom will have energy 0.5 kT). So, there can be a selfconsistency.

❖ If the initial estimation of \(d_C\) is too small, \(\omega_C\) will be too small, then \(d_C\) derived from \(\omega_C\) and T will be larger than the original \(d_C\). This will increase \(d_C\). In another word, the self-consistency is a stable one.
The basic idea (2)

- One can estimate such self-consistency one phonon mode at a time. But that will not be best. Since all these are results of anharmonic effect, it thus will be better to estimate all the modes together, all at their corresponding amplitudes at T (including phonon modes which are real at small T).

- The following procedure is actually a rough one, not really exact. It involves several approximations. But qualitatively, it can be useful.

- We will assume the same phonon modes calculated at the high symmetry point at T=0, even though some of the phonon modes have imaginary frequency. We call such phonon modes $\mu'_k(i)$, here $k$ is the mode index, $i$ is the atom index.

- For a given temperature T, and a calculated phonon mode frequency $\omega_k$, we can calculate a mode amplitude $Q_k$.

- Now, we can have an atomic displacement (measured from the high symmetry point): $dR(i) = \sum_k sign(k)Q_k\mu'_k(i)$
  Here, $sign(k)=+/-$ is a random set, but fixed throughout the iterations.

- Now, using this $dR(i)$, redo a SCF calculation, get $dF(i)$ (the force). Using these force $dF(i)$, and phonon mode $\mu_k(i)$, recalculate $\omega_k$, thus repeat the self-consistent loop.
The formulas and procedure: get the phonon modes

(1) Using PyPWmat, do JOB=std bulk phonon calculation, using a n1*n2*n3 supercell.

(2) In the PyPWmat calculation, in PWphonon.in, specify: MP=n1, n2, n3 (for phonon q-point generation).

(3) Get mesh.yaml.dat, which contains all the phonon modes for all n1*n2*n3 q-points.

(4) Read out these phonon modes as: $\mu'_k(i)$ and its frequency $\omega_k$ (here k is the composite index, including n1*n2*n3 q-points, and each q-point, there should be 3*natom modes).

Also note, for the notation, we have: $\mu'_k(i)$ is directly read-out from mesh.yaml.dat (adding an exp(-i*q*R) factor), so it has: $\sum_i \mu'_{k1}(i)\mu'^*_{k2}(i) = \delta_{k1,k2}$, thus: $\mu'_k(i) = \sqrt{M(i)}\mu_k(i)$

here “i” is the index of atom, and x,y,z, and M(i) is the mass of the atom, and $\mu_k(i)$ is the more conventional definition of the phonon mode.
The formula and procedure: SC-steps

(5) Now, if we assume, for each k mode, we have $\omega_k$ frequency. Then at temperature $T$, we will have the amplitude of this phonon mode as (the energy is: $E_k = 0.5 \omega_k^2 Q_k^2$):

$$Q_k = \sqrt{kT/\omega_k}$$ using classical high T formula, or quantum mechanics formula:

$$Q_k = \sqrt{\frac{1}{2} + \frac{1}{e^{h\omega_k/kT}} - 1}$$

(6) Now, we have a displacement:

$$dR(i) = \frac{1}{\sqrt{M(i)}} \sum_k s(k) Q_k \mu'_k(i)$$

Here $s(k)=+/-$ is a random set of signs, but it should be fixed throughout the self-consistent calculations. Note: if $\delta_{k1,k2}$ and $\mu'_k(i)$ are all real number.

(7) Using $R(i) = R_0(i) + dR(i)$, here $R_0(i)$ is the high symmetry phase atomic position, and the $n1*n2*n3$ supercell, do SCF calculations. Get the force $F(i)$.

(8) Estimate the frequency $\omega_k$ as:

$$\omega_k = \sqrt{- \frac{\sum_i \mu'_k(i) F(i) / \sqrt{M(i)}}{s(k)Q_k}}$$ for all modes $k$.

(9) Compare $\omega_k$ with $\omega_k$ in step (5), if the resulting $\omega_k$ is imaginary, decrease the original $\omega_k$ (e.g, by a factor of 2), if resulting $\omega_k$ is real, then do some mixing for this $\omega_k$ and $\omega_k$ in Step (5), using it as a new input $\omega_k$ in step (5), repeat (5) to (9), until $\omega_k$ no longer changes.
The actual codes and procedures

(1) **gen_dR_config0.f**: This code should be run in the phonon_std directory, after the bulk phonon modes have been calculated using “extract_phonon_data.sh”. It needs the htp.input file as the input, and needs to copy the unit cell atom.config in the directory. It is the initial step to generate dQ.000, Freq.000 and atom_sup.config. Freq.000 is the frequency, taken from mesh.yaml.dat. dQ.000 is the amplitude of each phonon mode for displacement. It also generate isign.store, install the sign (+/-) for each phonon mode displacement. This codes take the frequency for each mode, and kT, generate dQ and displacement dR. It also places the phonon mode in a file: muq_super.store for later usage.

(2) After run gen_dR_config0.f, run Pwmat in the phonon_std directory. To do this, you need the etot.input, and the pseudopotential file. Have OUT.FORCE=T in etot.input, and in.atom=atom_sup.config. After run this, it will generate OUT.FORCE.

(3) **gen_dR_config2.f**: This should be run after run Pwmat in step (2). Run it as: “gen_dR_config2.iter iter”, here iter is the iteration number (+1 number of the higher dQ.xxx,Freq.xxx in the directory). It takes the mode from muq_super.store, and the force in OUT.FORCE, and dQ.iter-1 to calculate the Frequency, store in Freq.iter. It also calculates the dQ.iter based on Freq.iter, kT and the procedural option provided in htp.input (e.g., with a mixing amix). It will also generate atom_sup.config.

(4) Repeat (2)-(3), until it converges (the frequency in Freq.xxx no longer changes). Can run it using iteration.sh.

(5) Sometime this is not easy to converge, one can do this for only a few phonon modes (all other modes have zero dQ), and keep their frequency at T=0 frequency (e.g, the positive phonons). This can be done with set_freq_fix.f, and modify the Freq.fix files (set the mode to be fix as 0 in the flag column). One can also change Freq.fix during iterations.

(6) One can also manually alter the dQ.iter (usually after run gen_dR_config2.f, so you already have Freq.iter), then run: gen_config_fromdQ.f (enter step number iter), it will generate atom_sup.config from dQ.
Post-processing

(1) If the SC-calculations are converged, we will have all positive $\omega_k$.

(2) But $\omega_k$ is only defined for the $n_1*n_2*n_3$ q-points. That may or may not be enough (usually, $n_1*n_2*n_3$, the supercell, is kind of small, like 2x2x2, or 3x3x3). To get more phonon modes, we can reconstruct the dynamic matrix as:

$$D(R_1, R_2) = \frac{\partial^2}{\partial R(i_1) \partial R(i_2)} E = \sqrt{M(i_1)M(i_2)} \sum_k \omega_k^2 \mu_k(i_1) \mu_k'(i_2)$$

(3) This is done by running: \texttt{gen_force_set.f}. It will take \texttt{Freq.iter}, and \texttt{muq_super.store} (modes) to regenerate \texttt{FORCE_SETS.NEW}. Copy \texttt{FORCE_SETS.NEW} to \texttt{FORCE_SETS}, run: \texttt{extract_phonon_data2.sh}, it will generate the new phonon band structure and DOS $g(\omega)$.

(4) After get all the phonon mode, we can also calculate the phonon free energy as:

\[ F(T) = \int_0^\infty d\omega g(\omega) \left[ \frac{\hbar \omega}{2} + kT \ln(1 - \exp(-\hbar \omega/kT)) \right] \]
Examples: GaAs

We first use bulk GaAs as an example, although it does not have imaginary frequency. But it will help to see the Procedure and the anharmonic effects.

Step (1): create directory >GaAs, prepare the Pwphonon.in, and fully cell relaxed atom.config

```plaintext
1 JOB = std
2 NODES = 1 4
3 WALL TIME = 1000.00:00
4 MP_N123 = 2 2 2 0 0 0
5 DIM = 2 2 2
6 PRIMITIVE AXIS = 1.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 1.0
7 FC_SYMMETRY = .TRUE.
8 ACCELERATION = .TRUE.
9 BAND = 0.0 0.0 0.0 0.5 0.0 0.0 # special q points in Brillouin zone
10 BAND_LABLES = $\Gamma$\text{\textbackslash}Gamma\text\{}\textbackslash{\textbackslash}# labels of the special q points
11 BAND_POINTS = 101
12 FREQUENCY\ CONVERT\ FACTOR = THz
13 DOS = .TRUE.
14 FMIN = -1.0
15 FMAX = 20.0
16 MP = 2 2 2
17 TOLGE = 0.1
18 SIGMA = 0.1
19 EIGENVECTORS = .TRUE.
20 DEFECT\ CENTER = 0.0 0.0 0.0 # only useful with job: defect
21 DEFECT\ RADIUS = 3
22 DISPLACEMENT = 0.003
```

MP_N123 is the kpt for supercell: For this calc, MP must be the same as DIM (supercell size), Displacement is small, Use THz as the unit for Frequency [IMPORTANT]
Examples: GaAs

Step (1): create directory >GaAs, prepare the Pwphonon.in, and fully cell relaxed atom.config

```
1 1 4
2 IN.ATOM = atom.config
3 JOB = scf
4 RELAX DETAIL = 1, 100, 0.01, 1, 0.003
5 Ecput = 80
6 Ecput2 = 320
7 IN.PSP1 = Ga.pbe.UPF
8 IN.PSP2 = As.pbe.UPF
9 WRT_ERROR = 1.0E-6
10 E_ERROR = 0.0
11 RHO_ERROR = 1.0E-6
12 MP_n123 = 4 4 4 0 0 0
```

The etot.input in the GaAs directory.
Note: the MP_n123 in etot.input will not be used.
It will not be the kpoint for the supercell calculation.

Step (2): Run: GaAs> PWmatPhonon.py
It will create phonon_std directory, and automatically submit Pwmat jobs to finish the forces-001, force-002 calculations.

```
1 2 2
2 PRIMITIVE_AXIS = 1.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 1.0
3 BAND = 0.0 0.0 0.0 0.0 0.0 0.0
4 BAND_LABELS = $\Gamma$ X
5 BAND_POINTS = 101
6 PC_SYM = .TRUE.
7 FREQUENCY_CONVERSION_FACTOR = 15.633302
8 MP = 2 2 2
9 DOS = .TRUE.
10 FMIN = -1.0
11 FMAX = 25.0
12 FPITCH = 0.1
13 SIGMA = 0.1
14 EIGENVECTORS = .TRUE.
15 MESH_SYMMETRY = .FALSE.
16 GAMMA_CENTER = .TRUE.
```

Very important: in atom.config, the same elements need to be grouped together (listed consecutively)!!

Step (3): After finish the Pwmat calculations, in GaAs/phonon_std, modify: band_dos.conf, add line:
MESH_SYMMETRY = .FALSE., GAMMA_CENTER = .TRUE.
Then run GaAs/phonon_std > extract_phonon_data.sh, it will generate the DOS, and band structure.
Also phonon mode file: mesh.yaml.dat
Examples: GaAs

Now, this finishes the conventional bulk phonon calculations. We will like to calculate the anharmonic effects in the following calculation. We like to calculate the phonon spectrum at T=300K.

This is how the directory phonon_std looks like after this step.

Now, copy atom.config from GaAs> into this directory, as well as pseudopotential files, and the etot.input. Inside etot.input, modify: MP_n123 to “2, 2, 2, 0, 0, 0, 3”, and add: OUT.FORCE=T, to be consistent with etot.input inside the force-001 etc.

Set: IN.ATOM=atom_sup.config
Examples: GaAs

Step (4): prepare htp.input,

The htp.input file.

Run: GaAs/phonon_std > gen_dR_config0.r
It will generate: dQ.000, Freq.000, atom_sup.config

Now, run Pwmat calculation (e.g.) >mpirun –n 4 Pwmat > out &

Step (5): after the first step Pwmat is finished, one should run: “>gen_dR_config2.r  1”, here 1 is for iter=1
It will generate: dQ.001,Freq.001, atom_sup.config. This will be followed by “Pwmat run again, thus formation
An iteration loop. “Gen_dR_config2.r iter” will take the OUT.FORCE and dQ.iter-1 to generate Freq.iter, and uses
The scheme specified in htp.input to generate dQ.iter from Freq.iter.

Alternatively, we can also run: >iteration.sh, this will automatically do iterations in Step (5) (e.g., for 8 iterations here).

Either manually repeat: gen_dR_config2.r & Pwmat runs, or using iteration.sh, hopefully it will converge.
If not, manual intervention, using set_freq_fix.f, or change dQ.iter, and gen_config_fromdQ.f will necessary.
Examples: GaAs

For the case of GaAs, after 3 iterations, the frequencies are converged. Here is the plot for the frequencies of different Modes (from Freq.iter) on different iterations: Interestingly, most of the modes becomes softer at finite T (especially for the soft modes, where it has the large dQ).
Examples: GaAs

Now, run: GaAs/phonon_std> gen_force_set.r, it will generate FORCE_SETS.NEW.

Now, copy FORCE_SETS.NEW to FORCE_SETS, run: extract_phonon_data2.sh

We will get the new phonon spectrum and files like: mesh.yaml.dat, band.dat.

Extract_phonon_data2.sh

1 #!/bin/sh
2 PWmat2Phonopy --pwmat band_dos.conf -c ./atom.config -p
3 phonopy-bandplot --gnuplot band,yaml > band.dat
4 echo "The data files can be used to plot"
5 ls *.dat

Red: The phonon line at T=0.
Green: The phonon line at T=300K

The difference represent the anharmonic, finite temperature effect!

Note: in reality, it might also be affected by the sign(atom) (the random seed in htp.input). One might want to check that by doing a few Random seed calculations.