2024 School on Electron-Phonon Physics, Many-Body Perturbation Theory, and Computational Workflows

Electron-phonon and anharmonic properties using the EPW/ZG code

Hands-on Session (Fri.8), 14th of June

Hands-on based on QE-v7.3.1 and EPW-v5.9

In this session we will learn how to use `ZG.x` for applying the special displacement method (SDM), and how to combine it with standard Density Functional Theory (DFT) calculations for evaluating temperature-dependent properties. The tutorial is long, so you are expected to finish the first three exercises. The rest are left as homework. More advanced users of the SDM are encouraged to proceed to the last three exercises on diamond, PbTe, and $CsPbBr_3$. You are advised to prepare the following script file, e.g. `script.sh`:

```
#!/bin/bash
#SBATCH --job=SDM
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=48
#SBATCH --time=01:00:00
#SBATCH --account=DMR23030
#SBATCH --partition=skx
#SBATCH --reservation=NSF_Summer_School_Fri
module purge
module load TACC
QE=/work2/05193/sabyadk/stampede3/EPWSchool2024/q-e/bin # path to "bin" of q-e
cd $PWD
# those settings are for stampede3 system of TACC
```

For each calculation, pass the command at the bottom of the script and submit the job, e.g. `sbatch script.sh`. Also set the following environment variable in your shell:

```
$ export QE=/work2/05193/sabyadk/stampede3/EPWSchool2024/q-e/bin/
```

For the school, the codes (both main ZG and local) are already compiled. If you take the tutorial separately, some routines in PW, PH, and PP directories are required to run these exercises (e.g. `make pw ph pp`). All necessary executables and the `ZG` module are compiled with `EPW`, i.e. when you type `make epw`. To use local executables go to `q-e/EPW/ZG/src/local` and type `./compile_ifort.sh`.

The description of all input flags is available in the following the link:
https://docs.epw-code.org/doc/InputsZG.html

▶ Copy the tutorial tarball from `/work2/05193/sabyadk/stampede3/EPWSchool2024/tutorials` and untar. Then, go to `exercise1`, and create directory `workdir`:

```
$ cd $SCRATCH
$ cp /work2/05193/sabyadk/stampede3/EPWSchool2024/tutorials/Fri.8.Zacharias.tar .
$ tar -xvf Fri.8.Zacharias.tar; cd Fri.8.Zacharias/exercise1; mkdir workdir
```

Note: in this tutorial we show how to obtain all input and output files. The directories `inputs` and `outputs` are used as a reference or to speed up the process. Most of the temperature-dependent calculations will be performed for $T = 0$ K; one can repeat the steps using a different temperature.

Exercise 1

In this exercise we will generate the ZG configuration of silicon in a $3 \times 3 \times 3$ supercell for the temperature $T = 0$ K and run your first DFT-ZG calculation. In the following, all steps for obtaining the phonons and interatomic force constants (IFCs) of silicon are provided; but to speed up the process you can skip the first *four* steps indicated by the small red arrow and go to the bottom of p. 3.

Note: For generating successfully a ZG configuration you need to make sure that the phonon dispersion is as you would expect (compare with literature). If there exist modes with negative frequencies, the code excludes them. If the system is anharmonic one should consider to evaluate anharmonic phonons using the A-SDM as in Exercise 3.

▶ Run a self-consistent calculation for silicon in the `workdir`.
Note: The energy cutoff ecutwfc needed for convergence should be 30 Ry.

```
$  cd workdir; cp ../inputs/si.scf.in .; cp ../inputs/Si.pz-vbc.UPF .
$  ibrun -np 4 $QE/pw.x -nk 4 < si.scf.in > si.scf.out
```

▶ Run a `ph.x` calculation on a homogeneous $4 \times 4 \times 4$ $\mathbf{q}$-point grid using the input:

```
--                                                                    si.ph.in
&inputph
   amass(1) = 28.0855,
   prefix   = 'si'
   outdir   = './'
   ldisp    = .true.
   fildyn   = 'si.dyn'
   tr2_ph   = 1.0d-12
   nq1      = 4, nq2 = 4, nq3 = 4
/
```

```
$ cp ../inputs/si.ph.in .
$ ibrun -np 4 $QE/ph.x -nk 4 < si.ph.in > si.ph.out
```

This will generate 8 **si.dynX** output files containing the dynamical matrix calculated for each irreducible $\mathbf{q}$-point. The list of irreducible $\mathbf{q}$-points is written in the **si.dyn0** file.

▶ Run a `q2r.x` calculation to obtain the interatomic force constants (IFC) file using the input:

```
--                                                                      q2r.in
&input
 fildyn='si.dyn', flfrc = 'si.444.fc'
/
```

```
$  cp ../inputs/q2r.in .
$  ibrun -np 1 $QE/q2r.x < q2r.in > q2r.out
```

This will generate **si.444.fc** which contains the IFCs.

▶ Run a `matdyn.x` calculation to check the phonon dispersion:

```
--                                                                    matdyn.in
&input
    asr='all', amass(1)=28.0855,
    flfrc='si.444.fc', fldyn='si.dyn.mat', flfrq='si.freq', fleig='si.dyn.eig',
    q_in_cryst_coord = .false., q_in_band_form = .true.
/
 9
 0.00  0.00  0.00  100
 0.75  0.75  0.00  1
 0.25  1.00  0.25  100
 0.00  1.00  0.00  100
```

```
0.00  0.00  0.00  100
0.50  0.50  0.50  100
0.00  1.00  0.00  100
0.50  1.00  0.00  100
0.50  0.50  0.50  100
```

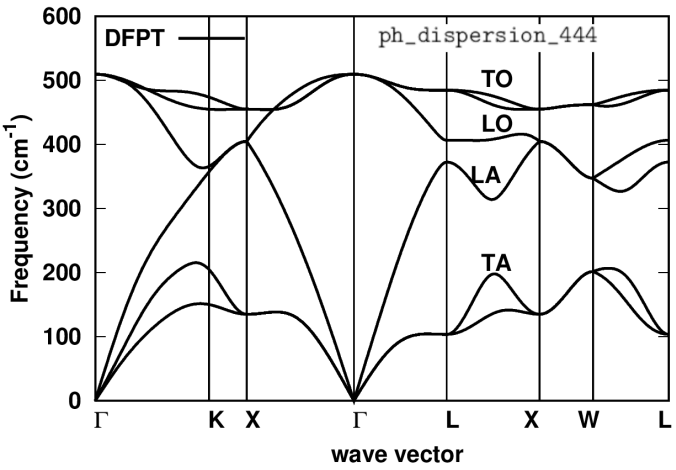
```
$  cp ../inputs/matdyn.in .
$  $QE/matdyn.x < matdyn.in > matdyn.out
$  $QE/plotband.x
```

```
     Input file > si.freq
Reading    6 bands at    702 k-points
Range:    0.0000  510.0194eV  Emin, Emax, [firstk, lastk] > 0 600
high-symmetry point:  0.0000 0.0000 0.0000   x coordinate   0.0000
high-symmetry point:  0.7500 0.7500 0.0000   x coordinate   1.0607
...
high-symmetry point:  0.5000 0.5000 0.5000   x coordinate   5.3534
output file (gnuplot/xmgr) > si_ph_dispersion.xmgr
bands in gnuplot/xmgr format written to file si_ph_dispersion.xmgr

output file (ps) >
stopping ...
```

`matdyn.x` will generate **si.freq** which contains the phonon frequencies along the specified path in the Brillouin zone. Then we combine **si.freq** with `plotband.x` to obtain **si_ph_dispersion.xmgr** in gnuplot/xmgr format for the phonon dispersion. To plot the phonon dispersion type:

```
$  cp si_ph_dispersion.xmgr ../gnuplot/.
$  cd ../gnuplot/; gnuplot gp_coms.p; evince ph_dispersion_444.pdf
```



Is the phonon dispersion as you expect? One can verify the results with literature data.

Once you have obtained the IFC file (e.g. here **si.444.fc**) and verified the correctness of your phonons, you are ready to perform a `ZG.x` calculation.

optional: If you have skipped the four steps above, you can copy the IFC file from `inputs` into `workdir`:

```
$ cd ../workdir; cp ../inputs/si.444.fc .
```

▶ Run a ZG calculation using the input:

```
--                                                                    ZG_333.in
&input
    flfrc='si.444.fc',
    asr='all', amass(1)=28.0855, atm_zg(1) = 'Si',
    flscf = 'si.scf.in'
    T = 0.00,
    dim1 = 3, dim2 = 3, dim3 = 3
    compute_error = .true., synch = .true., error_thresh = 0.1, niters = 15000
    incl_qA = .false.
/
```

Note: The $\mathbf{q}$-point grid (nq1×nq2×nq3) used to obtain the dynamical matrices should not be necessarily the same with the supercell size (dim1×dim2×dim3). In particular, dimX is independent of nqX, since ZG.x takes advantage of Fourier interpolation as implemented in matdyn.x; thus any size of ZG configuration can be generated from **si.444.fc**.

```
$  cd ../workdir
$  cp ../inputs/ZG_333.in .
$  ibrun -np 4 $QE/ZG.x -nk 4 < ZG_333.in > ZG_333.out
```

The input format of ZG_333.in is similar to matdyn.in. Lines 3 and 4 contain input variables as in matdyn.in (note that a new feature for the acoustic sum rule is added with asr = 'all', it can also be simple or crystal); in line 5 we provide the name of the unit-cell scf file (flscf) used to calculate the phonons. The information therein is read to generate the corresponding supercell scf file. In lines 6 and 7 we provide the temperature (T) in Kelvin and supercell dimensions (dimX) for which ZG displacements are generated; line 8 contains all flags related to the error minimization (discussed later); in line 8 we specify whether we want to include, or not, $\mathbf{q}$-points in set $\mathcal{A}$ using the flag incl_qA.

The output files from a ZG.x run are: (a) **ZG-configuration_0.00K.dat** and **equil_pos.dat** which contain the ZG (quantum nuclei at 0.00 K) and equilibrium (classical nuclei at 0 K) coordinates in angstroms, and (b) **ZG-scf_333_0.00K.in** and **equil-scf_333.in** which are the corresponding scf files for performing calculations in a $3 \times 3 \times 3$ supercell.

▶ Run a self-consistent calculation for silicon with the nuclei clamped at their equilibrium and ZG coordinates (for $T = 0.00$ K) in the supercell.

```
$  ibrun -np 48 $QE/pw.x -nk 3 < equil-scf_333.in > equil-scf_333.out
$  ibrun -np 48 $QE/pw.x -nk 8 < ZG-scf_333_0.00K.in > ZG-scf_333_0.00K.out
```

Note: parallelization over 3 and 8 **k**-points.

The file **equil-scf_333.in** is also provided here:

```
&control                                                          equil-scf_333.in
  calculation = 'scf'
  restart_mode = 'from_scratch'
  prefix = 'equil-si'
  pseudo_dir = './'
  outdir = './'
/
&system
  ibrav = 0
  nat =   54
  ntyp =    1
  ecutwfc =  20.00
  occupations = 'fixed'
  smearing = 'gaussian'
  degauss =   0.0D+00
```

```
/
&electrons
  diagonalization = 'david'
  mixing_mode= 'plain'
  mixing_beta = 0.70
  conv_thr =   0.1D-06
/
ATOMIC_SPECIES
   Si   28.085 Si.pz-vbc.UPF
K_POINTS automatic
   2   2   2   0   0   0
CELL_PARAMETERS {angstrom}
     -8.09641133      0.00000000      8.09641133
      0.00000000      8.09641133      8.09641133
     -8.09641133      8.09641133      0.00000000
ATOMIC_POSITIONS {angstrom}
   Si    0.00000000   0.00000000   0.00000000
   Si   -2.69880378   0.00000000   2.69880378
   Si   -5.39760755   0.00000000   5.39760755
...
```

Note: Cell parameters, number of atoms, **k**-grid, and atomic coordinates have been modified automatically based on the supercell dimensions. The code will always generate the lattice information in angstroms.

The file **ZG-scf_333_0.00K.in** is also provided here:

```
&control                                                    ZG-scf_333_0.00K.in
  calculation = 'scf'
  restart_mode = 'from_scratch'
  prefix = 'ZG-0.00K-si'
  pseudo_dir = './'
  outdir = './'
/
&system
  ibrav = 0
  nat =   54
  ntyp =    1
  ecutwfc =  20.00
  occupations = 'fixed',  smearing = 'gaussian', degauss =   0.0D+00
/
&electrons
  diagonalization = 'david'
  mixing_mode= 'plain'
  mixing_beta = 0.70
  conv_thr =   0.1D-06
/
ATOMIC_SPECIES
   Si   28.085 Si.pz-vbc.UPF
K_POINTS automatic
   2   2   2   0   0   0
CELL_PARAMETERS {angstrom}
     -8.09641133      0.00000000      8.09641133
      0.00000000      8.09641133      8.09641133
     -8.09641133      8.09641133      0.00000000
ATOMIC_POSITIONS {angstrom}
...
```

Note: Same information with **equil-scf_333.in**, except for the prefix and atomic positions. In general, you should add `nosym = .true.`, since symmetries do not apply after ZG displacements. The information printed by the code for constructing these scf files is basic; so for more complex cases one can modify the files based on their calculation needs.

While the calculations are running, let us discuss some aspects about the generation of the **ZG-**

configuration file. Type:

```
$ grep -3 "Optimum" ZG_333.out | tail -4
```

```
 Optimum configuration found !

 Sum of diagonal terms per q-point:    0.416464
 Error and niter index:    0.044777       6
```

This information is printed during a `ZG.x` run when the optimum configuration is found. Line 3 gives the sum of all diagonal terms per q-point, representing the denominator of Eq. (54) in Ref. [Phys. Rev. Res. 2, 013357 (2020)]. The first entry in line 4 represents the value of the minimization function $E(\{S_{\mathbf{q},\nu}\})$ given by Eq. (54) in Ref. [Phys. Rev. Res. 2, 013357 (2020)]. The integer in line 4 represents the number of attempts required to achieve $E(\{S_{\mathbf{q},\nu}\})$ smaller than the value specified for `error_thresh`. If this number exceeds the integer specified by `niters` flag, then `ZG.x` will stop without printing the ZG-configuration file. A general rule is: the larger the supercell, the fewer attempts are required to achieve $E(\{S_{\mathbf{q},\nu}\}) <$ `error_thresh`. The reason is: the order of choosing the unique set of signs, assigned to separate **q**-points, is less important as we approach the thermodynamic limit. Now type:

```
$ head -3 ZG-configuration_0.00K.dat
```

```
   Temperature is:     0.00  K
 Atomic positions          54
   Si       0.12836811      0.00747083      0.06987595
```

Lines 1 and 2 give the temperature for which ZG displacements are generated and the total number of atomic coordinates. In line 3, we have the first ZG atomic coordinate. It is perfectly reasonable to find different ZG coordinates, since the modes obtained by diagonalizing the dynamical matrix can differ by a phase factor (or a unitary matrix in case of degeneracy) if the processor, or compiler, or libraries have changed. The eigenvalues should remain the same. The flag `synch = .true.` applies a smooth Berry connection and aligns the sign of the modes with respect to a reference mode, but the sign of this reference depends on the machine. Degeneracy is not taken into account. The best way to check the validity of your configuration is by comparing the anisotropic mean-squared displacement tensor with the exact values, both printed in **ZG_333.out**. To this aim type:

```
$ grep -15 "Anisotropic" ZG_333.out | tail -16
```

```
 Anisotropic mean-squared displacement tensor vs exact values (Ang^2):

 Atom: 1
 -------
  ZG_conf:   Si    0.002583   0.002118   0.002354
    Exact:   Si    0.002339   0.002339   0.002339

 Atom: 2
 -------
  ZG_conf:   Si    0.002281   0.002400   0.002298
    Exact:   Si    0.002339   0.002339   0.002339


   off-diagonal terms
   Si   -0.000026  -0.000449  -0.000525
   Si    0.000395  -0.000460  -0.001429
```

Reducing error_thresh brings the anisotropic displacement tensor closer to the exact values. Note that in the previous versions of the code these values were multiplied by $8\pi^2$.

Now check whether the calculations have finished and type:

```
$ grep ! *scf_333*out
```

```
equil-scf_333.out:!    total energy              =    -427.83892822 Ry
ZG-scf_333_0.00K.out:!    total energy              =    -427.72146756 Ry
```

These are the total energies (Kohn-Sham potential energy) of the equilibrium and ZG structures. What is the difference between the two values and what is the reason of this difference? The difference is $\Delta E_{\rm KS} = 0.1175$ Ry and is due to the vibrational potential energy. To check this type:

```
$ grep -5 "Potent" ZG_333.out
```

```
    Temperature is:     0.00  K
 ===============================================

    Total vibrational energy:    0.24054327 Ry at    0.00 K

             Potential energy:    0.12027163 Ry at    0.00 K

               Kinetic energy:    0.12027163 Ry at    0.00 K

     Vibrational free energy:       0.24054327 Ry at    0.00 K
```

Our computed $\Delta E_{\rm KS}$ is indeed almost equal to half the total vibrational energy, since a DFT calculation will not account for the vibrational kinetic energy contribution. The value 0.117 Ry deviates from 0.120 Ry, since (i) we exclude from our calculations the q-points belonging in set $\mathcal{A}$ and (ii) due to the error coming by using small ZG configurations. As we use larger supercells this small deviation reduces to zero and $\Delta E_{\rm KS} =$ `Total vibrational energy`$/2$. The code also prints the vibrational free energy evaluated using Eq. (A5) of Ref. [Phys. Rev. B 108, 035155 (2023)]. The information for the vibrational free energy is important for checking the convergence of the trial free energy in the A-SDM calculations (Exercise 3). Here, because our calculations are for $T = 0$ K, the vibrational free energy is equal to the total vibrational energy.

We will only need the charge-density files from `equil-si.save` and `ZG-0.00K-si.save` for the next exercises. Thus remove:

```
$ rm -r *wfc* si.save _ph0/  *-si.save/*wfc*
```

Exercise 2

In this exercise we will learn how to calculate the phonon-induced band gap renormalization and temperature-dependent band structures using the example of silicon and the $3 \times 3 \times 3$ ZG supercell. To obtain temperature-dependent band structures we will employ the band structure unfolding (BSU) technique with plane-waves as basis sets. For the theory of BSU please refer to Ref. [Phys. Rev. B 85, 085201 (2012)]. To speed up the process one can skip the following two steps.
First go to the directory `exercise2` and copy the following input files in your `workdir`:

```
$  cd ../../exercise2/; mkdir workdir; cd workdir
$  cp ../inputs/Si.pz-vbc.UPF .; cp ../inputs/si.scf.in .
$  cp ../inputs/si.bands.in .; cp ../inputs/bands.in .
```

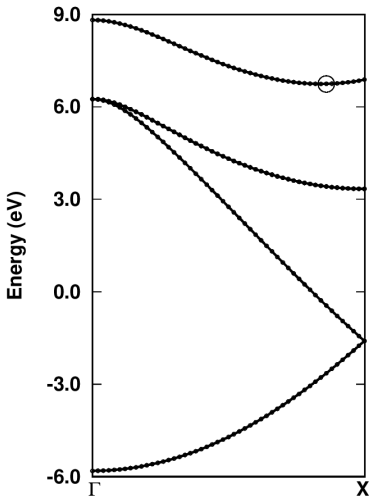
Note: in file `si.bands.in` we set `nbnd = 5` to include one empty band. We also sample the Γ-X path using 50 **k**-points in Cartesian coordinates (units of 2 pi/a).

▶ Run a standard band structure calculation in the unit-cell of silicon along Γ-X using 50 **k**-points.

```
$  ibrun -np 4 $QE/pw.x -nk 4 < si.scf.in > si.scf.out
$  ibrun -np 4 $QE/pw.x -nk 4 < si.bands.in > si.bands.out
$  ibrun -np 4 $QE/bands.x -nk 4 < bands.in > bands.out
```

▶ Plot the band structure using the file **bands.dat.gnu**.

```
$  cp bands.dat.gnu ../gnuplot/; cd ../gnuplot/; gnuplot gp_coms.p
$  evince band_structure.pdf
```



Can you find the Cartesian k-coordinates in units of `\tpiba` of the valence band maximum (VBM) and conduction band minimum (CBM)? Inspecting file **bands.dat**, those are:

```
             0.000000  0.000000  0.000000
-5.811    6.253    6.253    6.253    8.817
...
             0.000000  0.840000  0.000000
-2.783   -0.278    3.440    3.440    6.743
```

We can see that the VBM lies at the Γ-point and is threefold degenerated with energy $E_{\mathrm{VBM}} = 6.253$ eV. The CBM lies at $\sim$0.84 Γ-X with energy $E_{\mathrm{CBM}} = 6.743$ eV. Therefore, the DFT-LDA band gap with these settings is $E_{\mathrm{G}} = E_{\mathrm{CBM}} - E_{\mathrm{VBM}} = 0.49$ eV.

▶ Run a non self-consistent calculation for two **K**-points that define the VBM and CBM in the equilibrium and ZG supercell structures.

Note: Find the reciprocal lattice vector of the supercell $\mathbf{G}$ that maps **K** onto **k**, i.e. $\mathbf{K} = \mathbf{k} - \mathbf{G}$. The trivial choice is $\mathbf{G} = \mathbf{\Gamma}$ and thus set $\mathbf{K} = \mathbf{k}$.

To prepare the calculation proceed as follows:

```
$  cd ../workdir/
$ cp ../../exercise1/workdir/equil-scf_333.in equil-nscf_333.in
$ cp ../../exercise1/workdir/ZG-scf_333_0.00K.in ZG-nscf_333_0.00K.in
$ cp -r ../../exercise1/workdir/equil-si.save/ .
$ cp -r ../../exercise1/workdir/ZG-0.00K-si.save/ .
```

In `equil-nscf_333.in` and `ZG-nscf_333_0.00K.in` apply the following changes:

1. Set `calculation = 'nscf'`
2. Include one empty band / unit-cell by setting `nbnd = 135` below `ecutwfc = 20.0` flag.
3. Replace the automatic **K**-grid:

   ```
   K_POINTS automatic
      2   2   2   0   0   0
   ```

   with the two **K**-points of the VBM and CBM:

   ```
   K_POINTS crystal
   2
   0.000000 0.000000 0.000000 1
   0.000000 1.260000 1.260000 1
   ```

   Remark: Since the coordinates are given in units of the reciprocal lattice parameters, we obtain the K-coordinates by multiplying the k-coordinates of VBM and CBM states with the dimensions of the supercell, i.e. if $\mathbf{k} = [x \;\; y \;\; z]$ then $\mathbf{K} = [m \times x \quad n \times y \quad p \times z]$, where integers m, n, p define an $m \times n \times p$ supercell.
4. For the ZG input file add `nosym = .true.` below `nbnd = 135`.

optional: If you did not complete `exercise1` and the steps above, then copy files from `inputs`:

```
$ cd ../workdir/; cp ../inputs/*nscf_333*in .; cp -r ../inputs/*-si.save/ .
```

For example, the ZG input file should look like this:

```
 &control                                                    ZG-nscf_333_0.00K.in
  calculation = 'nscf'
  restart_mode = 'from_scratch'
  prefix = 'ZG-0.00K-si'
  pseudo_dir = './',
  outdir = './'
/
&system
  ibrav = 0, nat =   54, ntyp =    1
  nbnd = 135, ecutwfc =  20.00, nosym = .true.
  occupations = 'fixed',  smearing = 'gaussian',  degauss =   0.0D+00
/
```

```
&electrons
  diagonalization = 'david',  mixing_mode= 'plain',
  mixing_beta = 0.70, conv_thr =   0.1D-06
/
ATOMIC_SPECIES
   Si   28.085 Si.pz-vbc.UPF
K_POINTS crystal
2
0.000000 0.000000 0.000000 1
0.000000 1.260000 1.260000 1
CELL_PARAMETERS (angstrom)
     -8.09641133      0.00000000      8.09641133
      0.00000000      8.09641133      8.09641133
     -8.09641133      8.09641133      0.00000000
ATOMIC_POSITIONS (angstrom)
...
```

```
$  ibrun -n 28 $QE/pw.x -nk 2 < equil-nscf_333.in > equil-nscf_333.out
$  ibrun -n 28 $QE/pw.x -nk 2 < ZG-nscf_333_0.00K.in > ZG-nscf_333_0.00K.out
```

The calculations should take around 5 secs each. When the equilibrium calculation is finished, check that you obtain the VBM and CBM energies you would expect. To this aim type:

```
$ grep highest equil-nscf_333.out
highest occupied, lowest unoccupied level (ev):     6.2534    6.7435
```

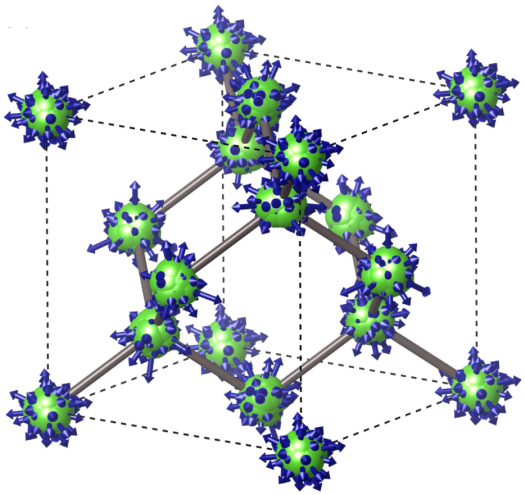
Indeed, we obtain the correct VBM and CBM energies and a gap of 0.4902 eV.
To get the band gap from the ZG calculation at 0 K type:

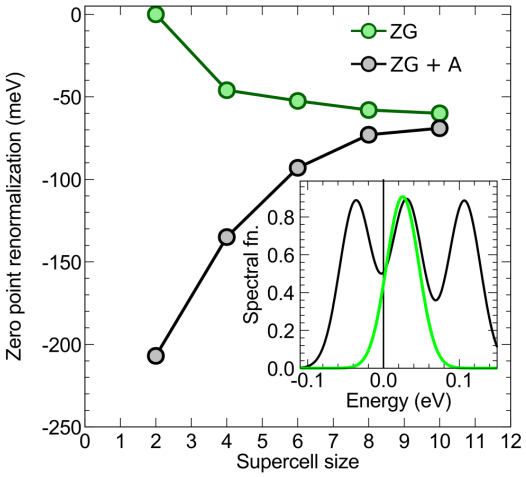
```
$   a=$(grep highest ZG-nscf_333_0.00K.out | awk '{print $7}')
$   grep $a ZG-nscf_333_0.00K.out | head -1  # VBM
    5.4332   6.2765   6.2828   6.2912   6.8273   6.8375   6.8609   6.8953
$   a=$(grep highest ZG-nscf_333_0.00K.out | awk '{print $8}')
$   grep $a ZG-nscf_333_0.00K.out | head -1  # CBM
    5.6192   5.6710   5.8206   5.8362   6.7184   7.2537   7.3388   7.5575
```

We can see that there is a splitting of the originally threefold degenerated VBM of the order: $\Delta E = 6.2912 - 6.2835 \simeq 8$ meV. This degeneracy breaking results from the ZG displacements in a finite size supercell and should reduce to zero for larger ZG simulation cells. This is a consequence of the harmonic approximation, where the symmetries of the system should be maintained upon thermal averaging (see thermal ellipsoids in the plot below).

We note that for minimizing the effect of band degeneracy splitting we excluded the modes in set $\mathcal{A}$. In Ref. [Phys. Rev. Res. 2, 013357 (2020)] we show that these modes dominate VBM splitting (see the spectral function in the figure below), giving $\Delta E > 80$ meV for a $4 \times 4 \times 4$ ZG supercell. Hence, for finite size systems with degenerated band edges, it is suggested to keep `incl_qA = .false.`.

To deal with this finite size supercell artefact, we determine the VBM by taking the average of the three bands as done in the commands above. For example in **ZG-nscf_333_0.00K.out**, the energy of the VBM is: $E_{\rm VBM} = (6.2765 + 6.2828 + 6.2912)/3 = 6.2835$ eV. Thus the band gap at 0K is $E(0\,{\rm K}) = 6.7184 - 6.2835 = 0.4349$ eV, and we can determine a zero-point renormalization $E_{\rm ZPR} = 0.4901 - 0.4349 \simeq 55$ meV. This value is in good agreement with literature data (check, for example, Refs. [J. Chem. Phys. 143, 102813 (2015)] and [New J. Phys. 20, 123008 (2018)]).





Note: In general, to obtain reliable values of the zero-point renormalization, you should always check convergence with respect to the ZG supercell size (see the right part of the figure above).

Before going to the next step remove unnecessary files:

```
$ rm -r *wfc* *save
```

▶ Prepare a band structure unfolding calculation using the $3 \times 3 \times 3$ ZG configuration. Create a new directory and copy the necessary input files:

```
$ mkdir band_structure_unfolding/; cd band_structure_unfolding/
$ cp ../../inputs/bands_unfold.in bands.in; cp ../../inputs/Si.pz-vbc.UPF .
$ cp -r ../../../exercise1/workdir/ZG-0.00K-si.save/ .
$ cp ../ZG-nscf_333_0.00K.in ZG-bands_333_0.00K.in
```

In `ZG-bands_333_0.00K.in` apply the following modifications:

1. Set `calculation = 'bands'`
2. Replace the following lines containing information about the **K**-points:

```
K_POINTS crystal
2
0.000000 0.000000 0.000000 1
0.000000 1.260000 1.260000 1
```

with

```
K_POINTS crystal_b
2
0.000000 0.000000 0.000000 35
0.000000 1.500000 1.500000 1
```

Note: Here we choose to sample the Γ-X path with 36 equally-spaced **K**-points.

If you did not complete the previous step, then copy:

```
$ cp ../../inputs/ZG-bands_333_0.00K.in .
```

The file `ZG-bands_333_0.00K.in` should look like this:

```
&control                                                    ZG-bands_333_0.00K.in
  calculation = 'bands'
  restart_mode = 'from_scratch'
  prefix = 'ZG-0.00K-si'
  pseudo_dir = './'
  outdir = './'
/
&system
  ibrav = 0
  nat =   54
  ntyp =    1
  nbnd = 135, nosym = .true., ecutwfc =  20.00
  occupations = 'fixed'
  smearing = 'gaussian'
  degauss = 0.0D+00
/
&electrons
  diagonalization = 'david'
  mixing_mode= 'plain'
  mixing_beta = 0.70
  conv_thr =   0.1D-06
/
ATOMIC_SPECIES
   Si   28.085 Si.pz-vbc.UPF
K_POINTS crystal_b
2
0.000000 0.000000 0.000000 35
0.000000 1.500000 1.500000 1
CELL_PARAMETERS {angstrom}
     -8.09641133      0.00000000      8.09641133
      0.00000000      8.09641133      8.09641133
     -8.09641133      8.09641133      0.00000000
ATOMIC_POSITIONS {angstrom}
...
```

We also show here the file `bands.in`:

```
--                                                                  bands.in
&bands
 prefix  = 'ZG-0.00K-si'
 outdir  = './'
 filband = 'bands.dat'
 lsym    = .false.
 dim1    = 3,
 dim2    = 3,
 dim3    = 3
/
```

Note: The input variables in `bands.in` are the same as in a standard `bands.x` calculation. The only difference is the flags `dimX` used to specify the dimensions of the ZG supercell.

▶ Run a band structure unfolding calculation.

```
$  ibrun -n 48 $QE/pw.x -nk 12 < ZG-bands_333_0.00K.in > ZG-bands_333_0.00K.out
$  ibrun -n 28 $QE/bands_unfold.x < bands.in > bands.out
```

Note: Parallelization over **k**-points in `bands_unfold.x` is not implemented.

The bands calculation should take around 40 secs and the unfolding around 4 sec. Now read the outputs of **bands01.dat** and **spectral_weights01.dat**. Can you guess what's the meaning of the spectral weights for each band?

```
$  head -16 bands01.dat
```

```
 &plot nbnd= 135, nks=    36 /
           0.000000  0.000000  0.000000
    -5.835502     -4.538594     -4.530763     -4.523530     -4.499082   ...
    -3.908153     -3.894924     -3.872739     -3.810481     -3.778147   ...
    -2.482541     -2.472723     -2.444686     -2.416635     -2.374668   ...
    -0.601738     -0.585920     -0.573318     -0.535188     -0.508788   ...
     0.770211      0.785536      0.830510      0.877961      0.890418   ...
     1.270759      1.327158      1.332847      1.629158      1.634093   ...
     ...
```

```
$ head -16 spectral_weights01.dat
```

```
 &plot nbnd= 135, nks=    36 /
           0.000000  0.000000  0.000000
   0.988833    0.000257    0.000674    0.004029    0.000067    ...
   0.000272    0.000063    0.000675    0.000295    0.000317    ...
   0.000381    0.000283    0.000135    0.000132    0.000134    ...
   0.000072    0.000161    0.000127    0.000191    0.000317    ...
   0.000172    0.000207    0.000168    0.000157    0.000201    ...
   0.000211    0.000183    0.000078    0.000132    0.000125    ...
   ...
```

Can you spot the weights corresponding to the VBM? What would the weights be if we use the equilibrium $3 \times 3 \times 3$ supercell? In this case, one should obtain the perfect unfolding reproducing the unit cell band structure. The data in the files **bands01.dat** and **spectral_weights01.dat**, represent the band energies, $\varepsilon_{m\mathbf{K}}(T)$, and spectral weights, $P_{m\mathbf{K},\mathbf{k}}(T)$, for all **K**-points, respectively.
Now you have all the ingredients to evaluate the spectral function given by:

$$A_{\mathbf{k}}(\varepsilon, T) = \sum_{m\mathbf{K}} P_{m\mathbf{K},\mathbf{k}}(T)\delta[\varepsilon - \varepsilon_{m\mathbf{K}}(T)] \tag{1}$$

▶ Calculate the spectral function using the energies and spectral weights. To this aim convert first **bands01.dat** and **spectral_weights01.dat** into a different format using `plotband.x` of QE:

```
$ cp ../../inputs/energies.in .
$ cp ../../inputs/spectral_weights.in .
$ $QE/plotband.x spectral_weights01.dat < spectral_weights.in > spw.out
$ $QE/plotband.x bands01.dat < energies.in > energies.out
$ sed -i '/^\s*$/d' spectral_weights.dat # remove empty lines
$ sed -i '/^\s*$/d' energies.dat # remove empty lines
$ paste energies.dat spectral_weights.dat > tmp
$ awk '{print $1,$2,$4}' tmp > energies_weights.dat; rm tmp
```

You have just generated **energies_weights.dat** file in a similar format to a ".gnu" file where the first column has the momentum-axis values, the second the energies, and the third the spectral weights. Now proceed with the calculation of the spectral function using `pp_spctrlfn.x`:

```
$ cp ../../inputs/energies_weights.dat . # If you missed the step above
$ cp ../../inputs/pp_spctrlfn.in .
$ ibrun -n 4 $QE/pp_spctrlfn.x -nk 4 < pp_spctrlfn.in > pp_spctrlfn.out
```

The file `pp_spctrlfn.in` takes the following input variables:

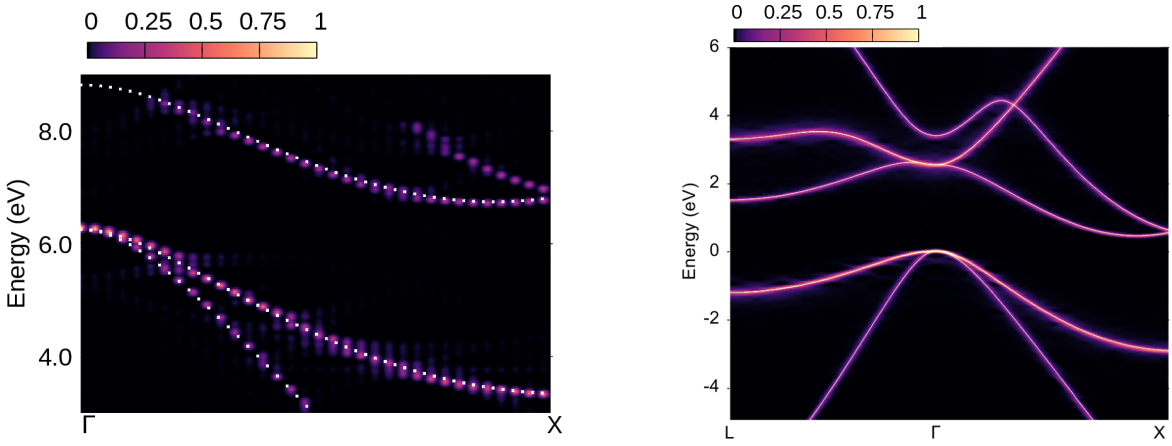
```
--                                                                    pp_spctrlfn.in
&input
  flin = 'energies_weights.dat'
  steps = 4860,
  ksteps = 200,
  esteps = 200,
  kmin = 0,
  kmax = 0.7071,
  emin =  3.00
  emax = 9.00
  flspfn = 'spectral_function.dat'
/
```

Here, (i) `steps` is the number of rows in the input file `flin`, (ii) `ksteps` and `esteps` define the resolution of the spectral function along the momentum-axis and energy-axis, (iii) (`kmin - kmax`) and (`emin - emax`) define the momentum and energy windows, and (iv) in `flspfn` we provide the name of the output file.

Now plot the spectral function:

```
$ cp spectral_function.dat ../../gnuplot/. ; cd ../../gnuplot/
$ cp ../inputs/bands.dat.gnu .
$ gnuplot gp_pm3d.p; evince bsu_si_0K.pdf
```

We also copied `bands.dat.gnu` for comparing the spectral function with the band structure calculated with the atoms at their equilibrium positions (white lines in the plots below) - note also that we assign a spectral weight of 1.0 in the third column of `bands.dat.gnu`. In the right part of the figure below, we also reproduced the converged calculation from Ref. [Phys. Rev. Res. 2, 013357 (2020)], employing an $8 \times 8 \times 8$ ZG supercell, more **K**-points and empty bands.



In **APPENDIX / Exercise 2** we provide a recipe for performing band structure unfolding for each **K**-point separately. This approach is most effective when larger supercells are employed and `pw.x` cannot handle all **K**-points in one run.

Exercise 3

In this exercise we will learn how to calculate temperature-dependent anharmonic phonons of Zr at T =1188 K using $2 \times 2 \times 2$ ZG supercells and the anharmonic special displacement method (A-SDM). The A-SDM combines the special displacement method with the self-consistent phonon (SCP) theory; details for the theory related to this exercise please refer to Refs. [Phys. Rev. B **108**, 035155 (2023)] and [D. Hooton, LI. a new treatment of anharmonicity in lattice thermodynamics: I, London Edinburgh Philos. Mag. J. Sci. 46, 422 (1955)]. The main aim is to find the effective matrix of IFCs that best describes anharmonicity in the potential energy surface. For this reason we minimize the trial free energy (as seen in the lecture) with respect to the matrix of IFCs which requires, essentially, to solve self-consistently the following equation:

$$\tilde{C}_{p\kappa\alpha,\kappa'\alpha'}(T) = \left\langle \frac{\partial U}{\partial \tau_{p\kappa\alpha} \partial \tau_{p'\kappa'\alpha'}} \right\rangle_T \tag{2}$$

In the following we are going to show how to solve the above equation self-consistently by performing a series of finite difference calculations.

First go to the directory `exercise3` and copy the following input files in your `workdir`:

```
$  cd ../../exercise3/; mkdir workdir; cd workdir
$  cp ../inputs/Zr.upf ../inputs/scf.in ../inputs/ph.in .
$  cp ../inputs/q2r.in ../inputs/matdyn.in .
```

Below we provide the `scf.in` file:

scf.in

```
&control
        calculation='scf'
        restart_mode='from_scratch'
        prefix = 'scf'
        outdir='./'
        pseudo_dir='./'
        verbosity='high'
/
&system
        ibrav=3
        nat=1
        ntyp=1
        ecutwfc=  30
        A = 3.51695
        occupations = 'smearing'
        smearing='gaussian',
        degauss=0.005d0
/
&electrons
        diagonalization='david'
        mixing_mode='plain'
        mixing_beta=0.7
        conv_thr=1e-7
/
ATOMIC_SPECIES
Zr 91.224 Zr.upf
K_POINTS {automatic}
6 6 6 0 0 0
ATOMIC_POSITIONS (crystal)
Zr 0.0 0.0 0.0
```

Note: Gaussian smearing for the occupations since Zr is a metal. For speed up we use a small energy cutoff of 30 Ry.

▶ Run a self-consistent calculation for Zr in the `workdir`.

```
$ ibrun -np 48 $QE/pw.x -nk 16 < scf.in > scf.out
```

▶ Run a `ph.x` calculation on a homogeneous $3 \times 3 \times 3$ $\mathbf{q}$-point grid using the input:

```
--                                                                    ph.in
&inputph
   amass(1)= 91.224,
   prefix = 'scf'
   outdir = './'
   verbosity = 'high'
   ldisp = .true.
   fildyn = 'Zr.dyn'
   tr2_ph = 1.0d-14
   nq1 = 3
   nq2 = 3
   nq3 = 3
/
```

```
$ ibrun -np 48 $QE/ph.x -nk 16 < ph.in > ph.out
```

This will generate 4 **Zr.dynX** output files containing the dynamical matrix calculated for each irreducible $\mathbf{q}$-point. The list of irreducible $\mathbf{q}$-points is written in the **Zr.dyn0** file.

▶ Run a `q2r.x` calculation to obtain the interatomic force constants (IFC) file using the input:

```
--                                                                    q2r.in
&input
 fildyn='Zr.dyn', zasr='crystal', flfrc = 'Zr.333.fc'
/
```

```
$  ibrun -np 1 $QE/q2r.x < q2r.in > q2r.out
```

This will generate **Zr.333.fc** which contains the harmonic IFCs.

▶ Run a `matdyn.x` calculation to obtain the harmonic phonon dispersion:

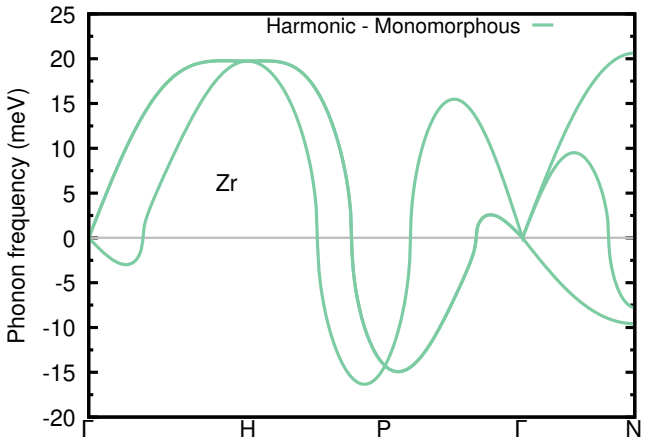
```
--                                                                    matdyn.in
&input
    &input
    asr='all', amass(1)= 91.224,
    flfrc='Zr.333.fc', fldyn=' ', flfrq='Zr_harm.freq', fleig=' ', q_in_cryst_coord = .true.
    q_in_band_form = .true.
/
5
 0.00   0.00   0.00 100
 0.50   0.50   0.50 100
 0.75   0.25  -0.25 100
 0.00   0.00   0.00 100
 0.50   0.50   0.00 1
```

```
$  $QE/matdyn.x < matdyn.in > matdyn.out
$  $QE/plotband.x
```

```
     Input file > Zr_harm.freq
Reading    3 bands at    401 k-points
Range: -131.7354  166.3357eV  Emin, Emax, [firstk, lastk] > -200 200
high-symmetry point:  0.0000 0.0000 0.0000   x coordinate   0.0000
high-symmetry point:  0.0000 0.0000 1.0000   x coordinate   1.0000
high-symmetry point:  0.5000 0.5000 0.5000   x coordinate   1.8660
high-symmetry point:  0.0000 0.0000 0.0000   x coordinate   2.7321
high-symmetry point:  0.0000 0.5000 0.5000   x coordinate   3.4392
output file (gnuplot/xmgr) > Zr_harm_ph_dispersion.xmgr
bands in gnuplot/xmgr format written to file Zr_harm_ph_dispersion.xmgr
output file (ps) >
stopping ...
```

matdyn.x will generate **Zr_harm.freq** which contains the harmonic phonon frequencies. Then we combine **Zr_harm.freq** with plotband.x to obtain **Zr_harm_ph_dispersion.xmgr** in gnuplot/xmgr format for the harmonic phonon dispersion. To plot the phonon dispersion type:

```
$  cp Zr_harm_ph_dispersion.xmgr ../gnuplot/.
$  cd ../gnuplot/; gnuplot gp_coms_harm.p; evince Zr_harm_ph_dispersion.pdf
```



Is the matrix of IFCs positive definite (i.e. all phonon frequencies are positive)? Unlike Si phonon dispersion in Exercise 1, the answer is no and the phonon dispersion exhibits severe instabilities. Here the harmonic approximation is not enough since anharmonic effects are important in Zr. To treat anharmonicity we are going to use the A-SDM in the following.

The starting point of the A-SDM for obtaining self-consistent phonons is a crucial step towards convergence and stability. This is because the SCP theory is designed to work with positive definite IFCs matrices. Although one could employ the harmonic phonons as a starting point (and neglect imaginary phonons), we strongly suggest the use of stable phonons as a starting point. For systems with a double well potential like Zr, well-defined dispersions can be obtained by exploring the ground state polymorphous structure, i.e a low-symmetry structure whose energy corresponds to a minimum of the potential energy surface [see also Ref. Phys. Rev. B 101, 155137 (2020)].

In the following we are going to show how to obtain the ground state polymorphous network of Zr and its stable phonons. If you have skipped the steps above, you can copy the harmonic IFCs file from inputs into workdir:

```
$ cd ../workdir; cp ../inputs/Zr.333.fc .
```

▶ Run an A_ZG calculation using the input:

```
--                                                                    ZG_1.in
&input
   asr='all', amass(1)= 91.224, flfrc='Zr.333.fc'
   flscf = 'scf.in'
   atm_zg(1) = 'Zr',
   T = 0
   dim1 = 2, dim2 =2, dim3 =2, incl_qA = .true.
   compute_error = .true., synch = .true., error_thresh = 0.2
   ASDM = .true.
/
&A_ZG
```

```
   poly = .true.
   poly_fd_forces = .false.
/
```

Note: The q-point grid (nq1×nq2×nq3) used to obtain the initial dynamical matrices should not be necessarily the same with the supercell size (dim1×dim2×dim3). In particular, dimX is independent of nqX, since ZG.x takes advantage of Fourier interpolation as implemented in matdyn.x; thus any size of ZG configuration can be generated from **Zr.333.fc**.

```
$  cp ../inputs/ZG_1.in .
$  $QE/ZG.x < ZG_1.in > ZG_1.out
```

The input format of &input list in ZG_1.in is the same as we have seen in Exercise 1. We remark that here we are using a 2×2×2 supercell and therefore `incl_qA = .true.` The temperature is set initially to `T = 0`, since we want to explore the polymorphous geometry. Line 7 contains the flag `ASDM` which enables the A-SDM procedure and allows the code to read the input variables in the list &A_ZG. At this step we ask the code by turning `poly = .true.` to generate the input file for performing a coordinates' optimization of the structure that will lead to the polymorphous geometry. Apart from the standard ZG.x output files described in Exercise 1, the code now prints **ZG-relax.in**. This file contains displaced coordinates of the atoms away from their high-symmetry positions (that define the monomorphous network) in a 2×2×2 supercell using ZG displacements.

The file **ZG-relax.in** is also provided here:

```
 &control                                                    ZG-relax.in
  calculation = relax
  restart_mode = 'from_scratch'
  prefix = 'ZG-relax'
  pseudo_dir = './'
  outdir = './'
  tprnfor=.true.
  disk_io = 'none'
  forc_conv_thr = 1.0D-5
  nstep = 300
/
&system
  ibrav = 0
  nat =    8
  ntyp =    1
  ecutwfc =  30.00
  occupations = 'smearing'
  smearing = 'gaussian'
  degauss =   0.5D-02
/
&electrons
  diagonalization = 'david'
  mixing_mode= 'plain'
  mixing_beta = 0.70
  conv_thr =   0.1D-06
/
&ions
  ion_dynamics = 'bfgs'
/
ATOMIC_SPECIES
   Zr   91.224 Zr.upf
K_POINTS automatic
   3   3   3   0   0   0
CELL_PARAMETERS (angstrom)
      3.51695000      3.51695000      3.51695000
     -3.51695000      3.51695000      3.51695000
     -3.51695000     -3.51695000      3.51695000
ATOMIC_POSITIONS (angstrom)
...
```

Note: Cell paramaters, number of atoms, **k**-grid, and atomic coordinates have been modified automatically based on the supercell dimensions. The code will always generate the lattice information in angstroms.

▶ Run nuclei coordinates optimization for Zr with the nuclei clamped at their ZG harmonic coordinates (for $T = 0.00$ K) in the 2×2×2 supercell.

```
$  ibrun -np 42 $QE/pw.x -nk 14 < ZG-relax.in > ZG-relax.out
```

The calculation should take around 4.5 minutes. Meanwhile you can check how the total energy evolves by typing "`grep !  ZG-relax.out`". Can you compare the total energy of the relaxed polymorphous structure with the one of the monomorphous structure (unit cell calculation in **scf.out**)? The polymorpous network is more stable by 22 meV/f.u. If you calculate an energy lowering much less than this value, then (i) increase the initial temperature, e.g. `T = 300` in `ZG_1.in`, (ii) remove file `ZGrelax.out`, and (iii) repeat the process. A higher temperature increases the phonon population, leading to larger atomic displacements. This can facilitate the exploration of polymorphous structures by preventing the atoms from returning to their high-symmetry positions.

▶ Run an A_ZG calculation using the same input file.

```
$  cp ../inputs/ZG_1.in ZG_2.in
$  $QE/ZG.x < ZG_2.in > ZG_2.out
```

The code now will search for the output file **ZG-relax.out**, read the relaxed final coordinates, and apply finite differences. This will generate **ZG-scf_poly_iter_00_XXXX.in** files corresponding to the scf inputs for evaluating the IFCs of the polymorphous network by finite differences. The default displacement on each atom is 0.01058 Å (i.e. 0.02 Bohr). One can change the default value by specifying a value in Å using the flag `fd_displ` under the namelist `&A_ZG`.

▶ Run scf calculations for files **ZG-scf_poly_iter_00_XXXX.in** and save all scf outputs in **fd_forces**.

```
$  file=ZG-scf_poly_iter_00
$  mkdir fd_forces/
$  for i in {0001..0048}; do
$    mkdir $i
$    cd $i
$    cp ../Zr.upf .
$    mv ../"$file"_"$i".in .
$    ibrun -np 42 $QE/pw.x -nk 14 < "$file"_"$i".in > "$file"_"$i".out
$    mv "$file"_"$i".out ../fd_forces/.
$    rm Zr.upf
$    cd ../
$  done
```

Each scf run should take around 3 seconds and the total calculation around 4 minutes. In the meantime, you can revisit the tutorial slides to refresh the A-SDM procedure.

▶ Run an A_ZG calculation using the same input file but turn the flag `poly_fd_forces = .true.`.

```
$  cp ../inputs/ZG_3.in ZG_3.in
$  $QE/ZG.x < ZG_3.in > ZG_3.out
```

`ZG_3.in` is also provided:

```
--                                                                    ZG_3.in
&input
   asr='all', amass(1)= 91.224, flfrc='Zr.333.fc'
   flscf = 'scf.in'
   atm_zg(1) = 'Zr',
   T = 0
   dim1 = 2, dim2 =2, dim3 =2, incl_qA = .true.
   compute_error = .true., synch = .true., error_thresh = 0.3
   ASDM = .true.
/
&A_ZG
   poly = .true.
   poly_fd_forces = .true.
   incl_epsil = .false.
/
```

The code now will read all the forces from the files **ZG-scf_poly_iter_00_XXXX.out** in **fd_forces** and evaluate the IFCs of the polymorphous structure. Apart from the standard `ZG.x` output files, the code now prints: (i) **FORCE_CONSTANTS_poly_iter_00** which contains the IFCs in the same format as `phonopy`, (ii) **FORCE_CONSTANTS_sym_poly_iter_00** which contains the symmetrized IFCs (i.e. after applying all symmetry operations of the crystal), and (iii) **poly_iter_00.fc** which contains the symmetrized IFCs in the same format as `q2r.x` output, ready to be used for the next A-SDM iteration. You can check the symmetries found by the code using "`grep "Sym." ZG_3.out`" or read **ZG_3.out** directly. For systems where the dielectric constant and Born effective charges are not zero, one may include this information in the **XXXX.fc** file by setting `incl_epsil = .true.`. Note that for including the non-analytic contribution to the dynamical matrix one needs to set `na_ifc = .true.` for the next ZG iteration run (`&input` list). This is because IFCs are evaluated by finite differences instead of DFPT. The same flag is also necessary for obtaining the phonon dispersion with `matdyn.x`; if not included the calculation will lead to erroneous results.

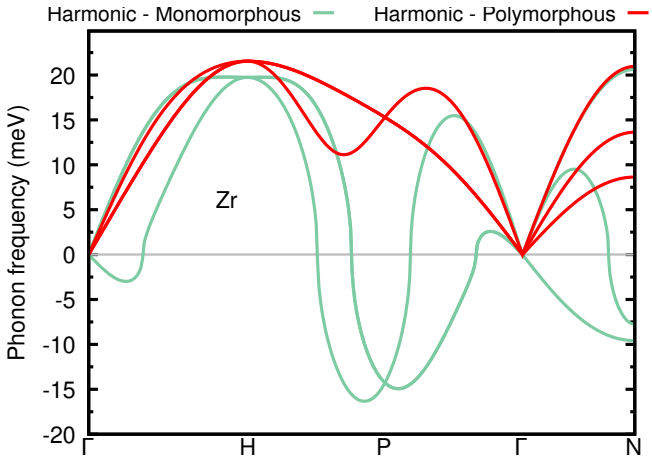
▶ Run a `matdyn.x` calculation to obtain the harmonic phonons of the Zr polymorphous network:

```
$  cp ../inputs/matdyn_poly.in .
$  $QE/matdyn.x < matdyn_poly.in > matdyn_poly.out
$  $QE/plotband.x
```

`matdyn.x` will generate **Zr_poly.freq** which contains the harmonic phonon frequencies of the Zr polymorphous network. Then combine **Zr_poly.freq** with `plotband.x` to create the phonon dispersion file **Zr_poly_ph_dispersion.xmgr** as before. To plot the new phonon dispersion type:

```
$  cp Zr_poly_ph_dispersion.xmgr ../gnuplot/.
$  cd ../gnuplot/; gnuplot gp_coms_poly.p; evince Zr_poly_ph_dispersion.pdf
```

The plot is provided below. Now the matrix of IFCs is positive definite (i.e. all phonon frequencies are positive) and we are ready to account for anharmonicity via the A-SDM using as a starting point the symmetrized IFCs of the Zr polymorphous network.



▶ Run an A_ZG calculation using the input file:

```
--                                                                    ZG_4.in
&input
   asr='all', amass(1)= 91.224, flfrc='poly_iter_00.fc'
   flscf = 'scf.in'
   atm_zg(1) = 'Zr',
   T = 1188
   dim1 = 2, dim2 =2, dim3 =2, incl_qA = .true.
   compute_error = .true., synch = .true., error_thresh = 0.3
   ASDM = .true.
/
&A_ZG
   iter_idx = 1
   apply_fd = .true.
/
```

Note: This is essentially the first iteration indicated by `iter_idx`. The file of IFCs is now `flfrc='poly_iter_00.fc'` and the target temperature is T =1188 K.

```
$  cd ../workdir; cp ../inputs/ZG_4.in ZG_4.in
$  $QE/ZG.x < ZG_4.in > ZG_4.out
```

This will generate a ZG configuration on which finite differences is applied after setting the flag `apply_fd = .true.`. The scf files are **ZG-scf_1188.00K_iter_01_XXXX.in** to be used for the evaluation of the temperature-dependent IFCs of the ZG configuration generated for T =1188 K.

▶ Run scf calculations for files **ZG-scf_1188.00K_iter_01_XXXX.in** and save all scf outputs in the folder **fd_forces**. The script below is as before but with a new name for `file` variable.

```
$  file=ZG-scf_1188.00K_iter_01  # note that only the filename changes
$  # mkdir fd_forces/
$  for i in {0001..0048}; do
$    # mkdir $i
$    cd $i
$    cp ../Zr.upf .
$    mv ../"$file"_"$i".in .
$    ibrun -np 42 $QE/pw.x -nk 14 < "$file"_"$i".in > "$file"_"$i".out
$    mv "$file"_"$i".out ../fd_forces/.
$    rm Zr.upf
$    cd ../
$  done
```

Each scf run should take around 3 seconds and the total calculation around 4.5 minutes. We can use this time for some questions.

▶ Run an A_ZG calculation using the input file:

```
--                                                                    ZG_5.in
&input
   asr='all', amass(1)= 91.224, flfrc='poly_iter_00.fc'
   flscf = 'scf.in', atm_zg(1) = 'Zr',
   T = 1188
   dim1 = 2, dim2 =2, dim3 =2, incl_qA = .true.
   compute_error = .true., synch = .true., error_thresh = 0.3
   ASDM = .true.
/
&A_ZG
   iter_idx0 = 0, iter_idx = 1
   apply_fd = .false.
   read_fd_forces = .true.
   mixing = .true.
   incl_epsil = .false.
/
```

Note: The flag `apply_fd` is set to `.false.` since `apply_fd` and `read_fd_forces` cannot be both true.

```
$  cp ../inputs/ZG_5.in ZG_5.in
$  $QE/ZG.x < ZG_5.in > ZG_5.out
```

With the flag `read_fd_forces = .true.` the code reads all the forces from files **ZG-scf_1188.00K_iter_01_XXXX.out** in the folder **fd_forces** and evaluates the IFCs of the ZG configuration. If one scf calculation is not successful, then the code will complain and print the error:

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
     Error in routine ZG (XXXX):
     forces in file are missing ...
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Here `XXXX` will be replaced by the code with the file index of the unsuccessful calculation.

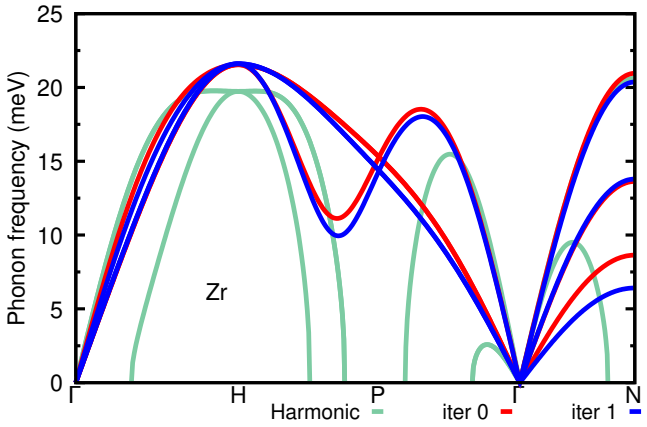
If all calculations run successfully, the code evaluates the IFCs using the finite differences formula and prints the following new files related to the first iteration:
(i) **FORCE_CONSTANTS_1188.00K_iter_01**,
(ii) **FORCE_CONSTANTS_sym_1188.00K_iter_01** and
(iii) **1188.00K_iter_01.fc** which contains the symmetrized IFCs in the same `q2r.x` output format, ready to be used for the next A-SDM iteration. Here, we set the flag `mixing = .true.`, so that iterative mixing of the IFCs is performed. The input variables `iter_idx0` and `iter_idx` denote the initial and final index of iterations, respectively, for which mixing of IFCs is performed.

▶ Run `matdyn.x` to obtain the temperature-dependent anharmonic phonons of the first iteration:

```
$  cp ../inputs/matdyn_01.in .
$  $QE/matdyn.x < matdyn_01.in > matdyn_01.out
$  $QE/plotband.x
```

`matdyn.x` will generate **Zr_01.freq** which contains the temperature-dependent anharmonic phonon frequencies of Zr at the first iteration. Then combine **Zr_01.freq** with `plotband.x` to create the phonon dispersion file **Zr_01_ph_dispersion.xmgr** as before. To plot the new phonon dispersion and compare with the previous results type:

```
$  cp Zr_01_ph_dispersion.xmgr ../gnuplot/.
$  cd ../gnuplot/; gnuplot gp_coms_iter_01.p; evince Zr_01_ph_dispersion.pdf
```



Here, we compare the phonon dispersion of Zr at T =1188 K obtained at the first iteration (blue) with the phonon dispersion of the polymorphous network (red) and the harmonic phonon dispersion (green). As we can see the result from the first iteration follows a similar trend with the one calculated for the polymorphous network. In the following we proceed to the second iteration and re-evaluate the phonon dispersion of Zr at T =1188 K starting from the IFCs of the previous iteration.

▶ Run an A_ZG calculation using the input file:

```
--                                                                                    ZG_6.in
&input
   asr='all', amass(1)= 91.224, flfrc='1188.00K_iter_01.fc'
   flscf = 'scf.in'
   atm_zg(1) = 'Zr',
   T = 1188
   dim1 = 2, dim2 =2, dim3 =2, incl_qA = .true.
   compute_error = .true., synch = .true., error_thresh = 0.35
   ASDM = .true.
/
&A_ZG
   iter_idx = 2
   apply_fd = .true.
/
```

```
$  cd ../workdir; cp ../inputs/ZG_6.in ZG_6.in
$  $QE/ZG.x < ZG_6.in > ZG_6.out
```

This will generate a new ZG configuration and the scf files **ZG-scf_1188.00K_iter_02_XXXX.in** to be used for the evaluation of the temperature-dependent IFCs of the second iteration.

▶ Run scf calculations for files **ZG-scf_1188.00K_iter_02_XXXX.in** and save all scf outputs in the folder **fd_forces**. If you have no much time left, you can skip this calculation, go to p. 25, and copy the ".xmgr" phonon dispersion file from directory `exercise3/outputs`.

```
$  file=ZG-scf_1188.00K_iter_02  # note that only the index changes
$  # mkdir fd_forces/
$  for i in {0001..0048}; do
$    # mkdir $i
$    cd $i
$    cp ../Zr.upf .
$    mv ../"$file"_"$i".in .
$    ibrun -np 42 $QE/pw.x -nk 14 < "$file"_"$i".in > "$file"_"$i".out
$    mv "$file"_"$i".out ../fd_forces/.
$    rm Zr.upf
$    cd ../
$  done
```

▶ Run an A_ZG calculation using the input file:

```
--                                                                    ZG_7.in
&input
   asr='all', amass(1)= 91.224, flfrc='1188.00K_iter_01.fc'
   flscf = 'scf.in', atm_zg(1) = 'Zr',
   T = 1188
   dim1 = 2, dim2 =2, dim3 =2, incl_qA = .true.
   compute_error = .true., synch = .true., error_thresh = 0.3
   ASDM = .true.
/
&A_ZG
   iter_idx0 = 0, iter_idx = 2
   apply_fd = .false.
   read_fd_forces = .true.
   mixing = .true.
   incl_epsil = .false.
/
```

```
$  cp ../inputs/ZG_7.in ZG_7.in
$  $QE/ZG.x < ZG_7.in > ZG_7.out
```

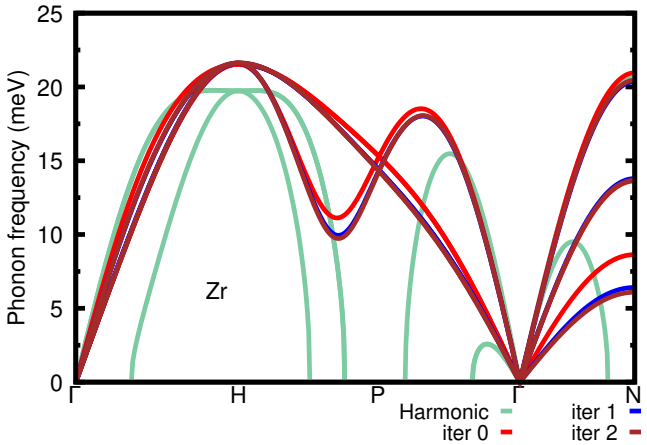
The code will read all the forces from the files **ZG-scf_1188.00K_iter_02_XXXX.out** in **fd_forces** and evaluate the IFCs of the new ZG configuration. The code now prints all files related to the second iteration:
(i) **FORCE_CONSTANTS_1188.00K_iter_02**,
(ii) **FORCE_CONSTANTS_sym_1188.00K_iter_02** and
(iii) **1188.00K_iter_02.fc**. Mixing is performed between the current and all previous IFCs matrices as specified by the input variables `iter_idx0` and `iter_idx`. We found that for the current system convergence is achieved faster. Instead, one can perform mixing between consecutive iterations (current and previous iteration) by choosing appropriately `iter_idx0` and `iter_idx`. In any case the converged result should be similar.

▶ Run a `matdyn.x` calculation to obtain the temperature-dependent anharmonic phonons at the second iteration:

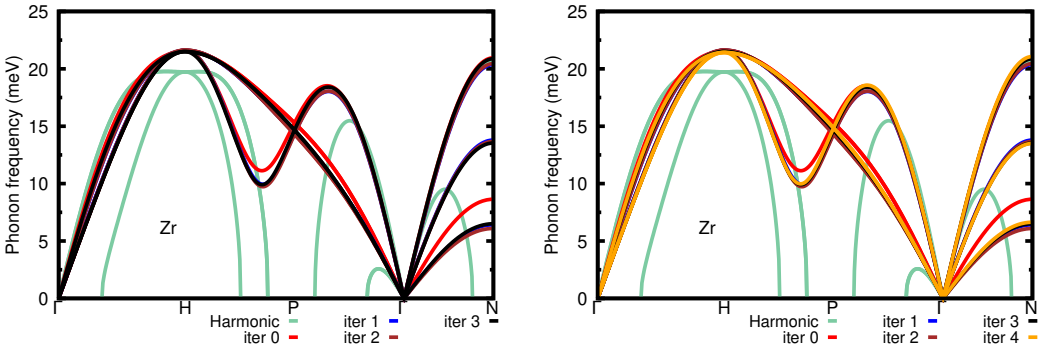
```
$  cp ../inputs/matdyn_02.in .
$  $QE/matdyn.x < matdyn_02.in > matdyn_02.out
$  $QE/plotband.x
```

`matdyn.x` will generate **Zr_02.freq** which contains the temperature-dependent anharmonic phonon frequencies of Zr at the second iteration. Then combine **Zr_02.freq** with `plotband.x` to create the phonon dispersion file **Zr_02_ph_dispersion.xmgr** as before. To plot the new phonon dispersion and compare with the previous results type:

```
$  cp Zr_02_ph_dispersion.xmgr ../gnuplot/.
$  cd ../gnuplot/; gnuplot gp_coms_iter_02.p; evince Zr_02_ph_dispersion.pdf
```



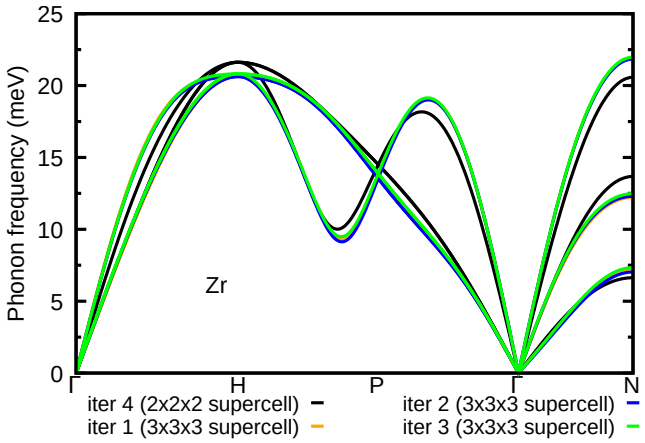
Here, we compare the phonon dispersion of Zr at T =1188 K obtained at the second iteration (brown) with the one at first iteration (blue), the phonon dispersion of the polymorphous network (red), and the harmonic phonon dispersion (green). As we can see the result at the second iteration is almost converged. One can repeat further iterative steps to check convergence.
As a homework repeat the previous steps for the third and fourth iterations. The results should look like the following:



We can readily see that convergence is obtained at the third iteration; note that the result from the first iteration is still in excellent agreement with the converged result.

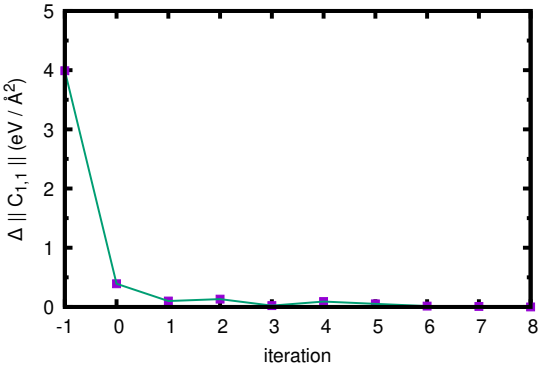
As a homework, also calculate the phonons of Zr at $T = 1188$ K using 3×3×3 supercells.
HINT: Start the A-SDM procedure from the IFCs of Zr at $T = 1188$ K obtained at the third iteration with 2×2×2 supercells (i.e. using `1188.00K_iter_03.fc`). The result is provided in the plot below.

For the convergence of the phonon dispersion when larger supercells are used, please refer to the results in Ref. [Phys. Rev. B **108**, 035155 (2023)]. Therein, results are obtained using more accurate settings, i.e. denser **k**-grid and higher kinetic energy cutoff.



To this end we also discuss four final important points:

- A quicker and more accurate way than checking the phonon dispersion is to plot the Frobenius norm ($||.||$) of the leading component of the IFCs matrix ($\tilde{C}_{11\alpha,11\alpha}(T)$) as suggested in Ref. [Phys. Rev. B 84, 180301(R) (2011)]. This information can be found by analysis of the first 3×3 matrix in **FORCE_CONSTANTS_sym_1188.00K_iter_XX**, where XX indicates the iteration. We note that convergence in IFCs guarantees essentially minimization of the system's free energy. The result for a few more iterations is shown below. "iteration -1" corresponds to the harmonic $||\tilde{C}_{11\alpha,11\alpha}||$ and the difference is taken with respect to $||\tilde{C}_{11\alpha,11\alpha}(T)||$ of the last iteration. Note also that IFCs unit has been converted from Ry/Bohr^2 to eV/Å^2.



- A full self-consistent phonon theory also minimizes the trial free energy with respect to nuclear coordinates. As seen in the lectures, one can employ the Newton-Raphson method and find the nuclear coordinates for which $\langle \partial U/\partial \tau_{\kappa\alpha} \rangle_T = 0$. This requires to solve self-consistently the equation:

$$\tau^{\text{new}}_{p\kappa\alpha}(T) = \tau_{p\kappa\alpha}(T) + \tilde{C}^{-1}_{p\kappa\alpha,p\kappa\alpha}(T)\,\langle F_{p\kappa\alpha}\rangle_T \tag{3}$$

using at each iterative step $\tilde{C}_{p\kappa\alpha,p\kappa\alpha}(T)$ as obtained from Eq. (2). In order to account for this feature in the A-SDM we need to set the flag `update_equil = .true.` when `read_fd_forces = .true.`. If we use this flag for the case of bcc Zr (point group symmetry $m\bar{3}m$), then the code will give $\tau^{\text{new}}_{p\kappa\alpha}(T) = \tau_{p\kappa\alpha}(T)$ (i.e. no change of the nuclear coordinates) at every iterative step. This comes as no surprise since symmetries are enforced and atoms in bcc Zr occupy only special Wyckoff positions. However, for systems containing

atoms at general Wyckoff positions, Eq. (3) will give new thermal equilibrium positions at each temperature that contribute to the minimization of the free energy. See for example the case of hydride compounds explored in Ref. [Commun. Phys. **6**, 298 (2023)].

- Obtaining the polymorphous structure is not a necessary step. We found that this is a very useful step for obtaining positive definite initial IFCs matrix for systems with a double well potential. If the harmonic approximation gives a positive definite initial IFCs matrix, then one could start the iterative procedure without obtaining the polymorphous network.
- After computing temperature-dependent anharmonic phonons using the A-SDM, one could then do a ZG calculation using the new configuration and evaluate anharmonic electron-phonon properties. See for example Exercise PbTe.

Exercise 4

In this exercise we will learn how to calculate the phonon-assisted optical spectra using the example of silicon and the $3 \times 3 \times 3$ ZG supercell for $T = 0$ K. To obtain temperature-dependent optical spectra we will evaluate the optical matrix elements in the independent-particle approximation using a routine very similar to `epsilon.x` of QE. We emphasize that this methodology can be extended to higher-level theories of optical absorption, like RPA, BSE, etc ... The present approach of calculating phonon-assisted optical spectra is based on Ref. [Phys. Rev. B 94, 075125 (2016)].

Note: The calculation of phonon-assisted optical spectra using ZG configurations has some drawbacks and advantages when compared to the perturbative methodologies, e.g. QDPT, implemented in `EPW`.

First go to the directory `exercise4`, create your working directories and copy the input files:

```
$  cd ../../exercise4/; mkdir workdir; cd workdir; mkdir equil; mkdir ZG
$   cd equil
$  cp ../../inputs/equil/K_list.in .
$  cp ../../inputs/equil/epsilon.in .
$  cp -r ../../../exercise1/workdir/equil-si.save/ .
$  cp ../../../exercise2/workdir/equil-nscf_333.in .
```

The file `K_list.in` contains the crystal coordinates of 200 randomly generated **K**-points. We use random **K**-points, instead of a uniform grid, in order to speed up convergence. In `equil-nscf_333.in` apply the following changes:

1. Add `nosym = .true.` below `nbnd = 135`.
   Note: although we use the equilibrium structure we set `nosym = .true.` since random **K**-points are employed.
2. Delete the information for the **K**-points:
   ```
   K_POINTS crystal
   2
   0.000000 0.000000 0.000000 1
   0.000000 1.260000 1.260000 1
   ```
   and at the **bottom** of the file add the following:
   ```
   K_POINTS crystal
   30
   ```
3. Copy and paste the first 30 **K**-points from file `K_list.in`.

If you did not complete Exercises 1 and 2, you can copy:

```
$  cp ../../inputs/equil/equil-nscf_333.in .
$  cp -r ../../../exercise2/inputs/equil-si.save/ .
```

The file `equil-nscf_333.in` should look like this:

```
&control                                                    equil-nscf_333.in
  calculation = 'nscf'
  restart_mode = 'from_scratch'
  prefix = 'equil-si'
  pseudo_dir = './'
  outdir = './'
/
&system
  ibrav = 0
  nat =   54
  ntyp =    1
```

```
  ecutwfc =  20.00
  nbnd = 135
  nosym = .true.
  occupations = 'fixed'
  smearing = 'gaussian'
  degauss =   0.0D+00
/
&electrons
  diagonalization = 'david'
  mixing_mode= 'plain'
  mixing_beta = 0.70
  conv_thr =   0.1D-06
/
ATOMIC_SPECIES
   Si   28.085 Si.pz-vbc.UPF
CELL_PARAMETERS {angstrom}
     -8.09641133      0.00000000      8.09641133
      0.00000000      8.09641133      8.09641133
     -8.09641133      8.09641133      0.00000000
ATOMIC_POSITIONS {angstrom}
   Si    0.00000000   0.00000000   0.00000000
   Si   -2.69880378   0.00000000   2.69880378
...
K_POINTS crystal
30
0.236650 0.022946 0.917156 1.0
0.392728 0.655175 0.471114 1.0
...
```

We also show here the file `epsilon.in`:

```
--                                                                    epsilon.in
&inputpp
outdir      = './',
prefix      = 'equil-si',
calculation = 'eps'
/
&energy_grid
smeartype   = 'gauss',
intersmear  = 0.03d0,
wmax        = 4.5d0,
wmin        = 0.2d0,
nw          = 600,
shift       = 0.0d0,
/
```

Note: The input variables in `epsilon.in` are the same as in a standard `epsilon.x` calculation.

▶ Run an optical spectrum calculation for silicon using the $3 \times 3 \times 3$ equilibrium supercell and 30 random **K**-points. The aim is to calculate the imaginary part of the dielectric function as:

$$\epsilon_2(\omega) = \frac{2\pi}{m_{\rm e} N_{\rm e}} \frac{1}{N_{\mathbf{K}}} \frac{\omega_{\rm p}^2}{\omega^2} \sum_{cv\mathbf{K}} |p_{cv\mathbf{K}}|^2 \delta(\varepsilon_{c\mathbf{K}} - \varepsilon_{v\mathbf{K}} - \hbar\omega), \tag{4}$$

where $m_{\rm e}$ is the electron mass, $N_{\rm e}$ is the number of electrons in the crystal unit cell, $\omega_{\rm p}$ is the plasma frequency, $N_{\mathbf{K}}$ is the number of **K**-points, and ω the photon frequency. The sum extends over the **K**-points, the occupied states of energy $\varepsilon_{v\mathbf{K}}$, and the unoccupied states of energy $\varepsilon_{c\mathbf{K}}$. $p_{cv\mathbf{K}}$ denotes the matrix elements of the momentum operator along a particular polarization direction. To calculate the dielectric function we use `epsilon_Gaus.x`, which is a slightly modified routine of `epsilon.x`, that replaces the delta function with a Gaussian instead of a Lorentzian function. We choose to apply Gaussian broadening in order to minimize numerical artefacts at the absorption onset.

```
$  ibrun -n 48 $QE/pw.x -nk 24 < equil-nscf_333.in > equil-nscf_333.out
$  ibrun -n 24 $QE/epsilon_Gaus.x < epsilon.in > epsilon.out
$  rm *wfc* *save/*wfc*
```

The full calculation should take around 50 secs. The output files are (i) `epsi_equil-si.dat` containing $\epsilon_2(\omega)$, (ii) `epsr_equil-si.dat` containing the real part of the dielectric function $\epsilon_1(\omega)$, and (iii) `eels_equil-si.dat` containing the electron energy loss spectrum; all calculated for the equilibrium structure. In each file the first column is the energy grid, and the rest three represent the observable along the three Cartesian directions.

▶ Take the isotropic average of the dielectric function and copy the output file in the `gnuplot` directory using:

```
$  awk '{print $1,($2+$3+$4)/3}' epsi_equil-si.dat > epsi_si_333_equil_30_av.dat
$  cp epsi_si_333_equil_30_av.dat ../../gnuplot/.
```

▶ Run an optical spectra calculation for silicon using the $3 \times 3 \times 3$ ZG supercell and 30 random **K**-points. We will essentially repeat all steps performed for calculating the optical spectra of the $3 \times 3 \times 3$ equilibrium supercell. First go to the `ZG` directory and copy the input files:

```
$  cd ../ZG/
$ cp ../../inputs/ZG/K_list.in .
$ cp ../../inputs/ZG/epsilon.in .
$ cp -r ../../../exercise1/workdir/ZG-0.00K-si.save/ .
$ cp ../../../exercise2/workdir/ZG-nscf_333_0.00K.in .
```

In `ZG-nscf_333_0.00K.in` apply the following change:

1. Delete the information for the **K**-points:

   ```
   K_POINTS crystal
   2
   0.000000 0.000000 0.000000 1
   0.000000 1.260000 1.260000 1
   ```

   and at the **bottom** of the file add the following:

   ```
   K_POINTS crystal
   30
   ```

2. Copy and paste the first 30 **K**-points from file `K_list.in`.

If you did not complete Exercises 1 and 2, copy `ZG-0.00K-si.save` and `ZG-nscf_333_0.00K.in` from:

```
$  cp ../../inputs/ZG/ZG-nscf_333_0.00K.in .
$  cp -r ../../../exercise2/inputs/ZG-0.00K-si.save/ .
```

The file `ZG-nscf_333_0.00K.in` should look like this:

```
&control                                                    ZG-nscf_333_0.00K.in
  calculation = 'nscf'
  restart_mode = 'from_scratch'
  prefix = 'ZG-0.00K-si'
  pseudo_dir = './'
  outdir = './'
/
&system
  ibrav = 0
  nat =   54
```

```
  ntyp =    1
  ecutwfc =  20.00
  nbnd = 135
  nosym = .true.
  occupations = 'fixed'
  smearing = 'gaussian'
  degauss =   0.0D+00
/
&electrons
  diagonalization = 'david'
  mixing_mode= 'plain'
  mixing_beta = 0.70
  conv_thr =   0.1D-06
/
ATOMIC_SPECIES
   Si   28.085 Si.pz-vbc.UPF
CELL_PARAMETERS {angstrom}
     -8.09641133      0.00000000      8.09641133
      0.00000000      8.09641133      8.09641133
     -8.09641133      8.09641133      0.00000000
ATOMIC_POSITIONS {angstrom}
...
K_POINTS crystal
30
  0.236650 0.022946 0.917156 1.0
  0.392728 0.655175 0.471114 1.0
...
```

epsilon.in is the same with the one used before, but with `prefix = 'ZG-0.00K-si'`.

▶ Now run the following to calculate $\epsilon_2(\omega)$ at 0.00 K:

```
$  ibrun -n 48 $QE/pw.x -nk 24 < ZG-nscf_333_0.00K.in > ZG-nscf_333_0.00K.out
$  ibrun -n 24 $QE/epsilon_Gaus.x < epsilon.in > epsilon.out
```

The full calculation should take around 50 secs. The output files are (i) `epsi_ZG-0.00K-si.dat` containing $\epsilon_2(\omega)$, (ii) `epsr_ZG-0.00K-si.dat` containing the real part of the dielectric function $\epsilon_1(\omega)$, and (iii) containing the electron energy loss spectrum `eels_ZG-0.00K-si.dat`; all calculated for $T = 0.00$ K. In each file the first column is the energy grid, and the rest three represent the temperature-dependent observable along the three Cartesian directions.

▶ Take the isotropic average of the dielectric function and copy the output file in the `gnuplot` directory using:

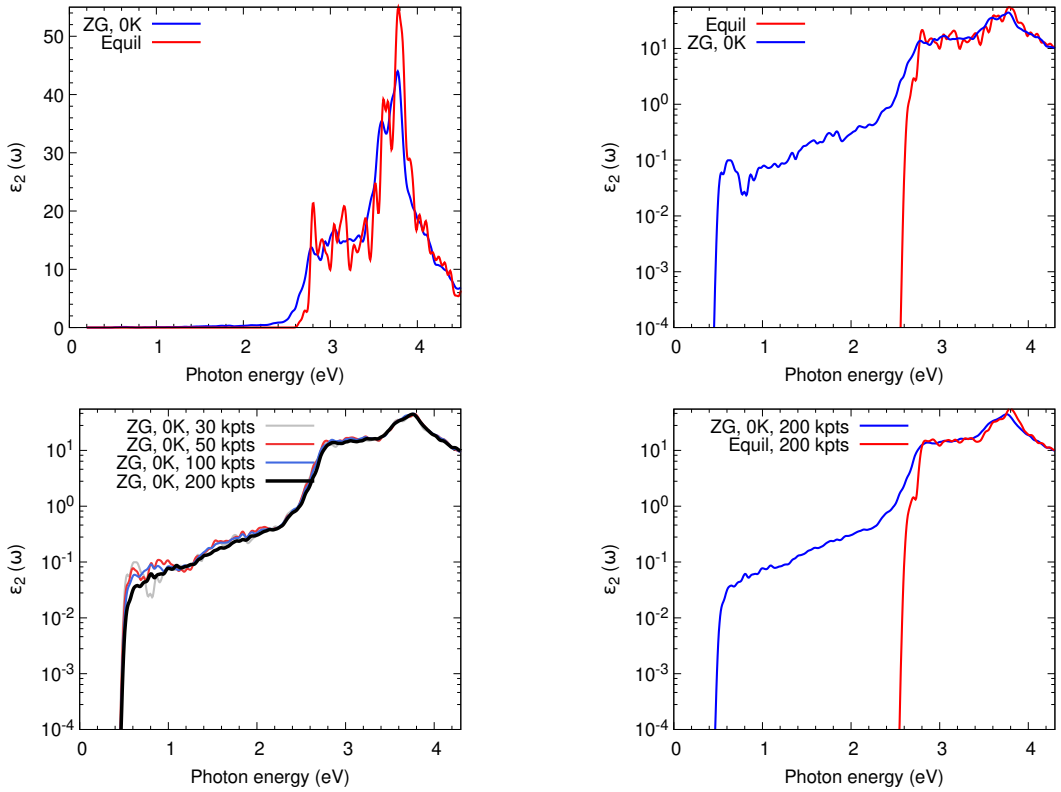
```
$  awk '{print $1,($2+$3+$4)/3}' epsi_ZG-0.00K-si.dat > epsi_si_333_ZG_30_av.dat
$  cp epsi_si_333_ZG_30_av.dat ../../gnuplot/.
$  rm *wfc* *save/*wfc*
```

The output file containing the data is **epsi_si_333_ZG_30.dat**.

Now go to the `gnuplot` directory and plot ZG and equilibrium spectra using:

```
$  cd ../../gnuplot/; gnuplot gp_coms.p; evince ZG_spectra.pdf
$  gnuplot gp_coms_logscale.p; evince ZG_spectra_logscale.pdf
```

We also plot $\varepsilon_2(\omega)$ as calculated with more **K**-points to obtain convergence.



What are the differences between the ZG and equilibrium spectra?
Are phonon-assisted transitions captured as expected?

As a homework, repeat the calculation with more random **K**-points using those provided in the file `K_list.in`. The result should look like as in the bottom panel of the figure above.

Ref. [Phys. Rev. B 94, 075125 (2016)] reports the convergence of the spectra with respect to the ZG supercell size and the agreement with experimental data for the absorption coefficient.

In **APPENDIX / Exercise 4** we provide a recipe for calculating optical spectra for each **K**-point separately. This approach is most effective when larger supercells are employed and `pw.x` cannot handle all **K**-points at once.

Exercise 5

In this exercise we will discuss the physical meaning of the ZG configuration and learn how to calculate phonon-induced diffuse (or inelastic) scattering patterns using graphene. This is an example of an observable which allows us to computationally reach the thermodynamic limit and assess multi-phonon processes. More details about the theory of phonon-induced inelastic scattering and its connection to the special displacement method can be found in Ref. [Phys. Rev. B 104, 205109 (2021)].

We will evaluate the exact expression for the all-phonon inelastic scattering intensity and compare it with the ZG scattering intensity, i.e. when the scatterers (nuclei) are defined by the ZG displacements. Go to the directory `exercise5`, create your working directories, and copy the input files:

```
$  cd ../../exercise5/; mkdir workdir; cd workdir; mkdir exact; mkdir ZG
$  cd  exact
$  cp ../../inputs/exact/graphene.881.fc .
$  cp ../../inputs/exact/disca.in .
```

Note: The file `graphene.881.fc` contains the IFC of graphene calculated using an $8 \times 8 \times 1$ $\mathbf{q}$-grid.

▶ Run a diffuse scattering calculation using `disca.x` and the input:

```
--                                                                                    disca.in
&input
    asr = 'all', amass(1) = 12.011, flfrc = 'graphene.881.fc', atm_zg(1) = 'C',
    dim1 = 40, dim2 = 40, dim3 = 1,
    T = 300,
    atmsf_a(1,1) = 0.1361,
    atmsf_a(1,2) = 0.5482,
    atmsf_a(1,3) = 1.2266,
    atmsf_a(1,4) = 0.5971,
    atmsf_a(1,5) = 0.0,
    atmsf_b(1,1) = 0.3731,
    atmsf_b(1,2) = 3.2814,
    atmsf_b(1,3) = 13.0456,
    atmsf_b(1,4) = 41.0202,
    atmsf_b(1,5) = 0.0,
    zero_one_phonon = .true., full_phonon = .true.
    nks1 = 0,  nks2 = 0,  nks3 = 0, nksf1 = 5, nksf2 = 5, nksf3 = 1,
    plane_val = 0.d0, plane_dir = 3
    qstart = 1, qfinal = 40000,
/
&pp_disca
    nrots = 6,
    kres1 = 250,
    kres2 = 250,
    kmin  = -10,
    kmax  = 10,
    col1  = 1,
    col2  = 2,
    Np    = 1600
/
```

```
$ ibrun -np 48 $QE/disca.x -nk 48 < disca.in > disca.out
```

The input format of `disca.in` is similar to `ZG_333.in`. Line 3 contains input variables similar to those in matdyn.in; in line 4 we provide the dimensions `dimX` of the $\mathbf{q}$-grid used to sample the Brillouin zone; in line 5 we provide the temperature `T` in Kelvin; lines 6-15 contain the parameters of the atomic scattering factor (`atmsf_a` and `atmsf_b`) from Ref. [Micron 30, 625–648, (1999)]; in line 16 we specify whether we want to compute the one-phonon (`zero_one_phonon`) and/or all-phonon

(full_phonon) structure factor; in line 17 nksX and nksfX define the range of reciprocal lattice vectors (in crystal coordinates) which are used to determine the extent of the calculated scattering vectors (**Q**-points); in line 18 we provide plane_val that defines the plane along which the structure factor is calculated (in units of 2pi/alat) and plane_dir defines the Cartesian direction perpendicular to the plane (where 1 --> x, 2 --> y, and 3 --> z); and in line 19 we specify the range of **Q**-points (from qstart to qfinal) for which the structure factor is calculated.

In the namelist &pp_disca we have: (i) nrots which is the number of rotations to be applied on the all-phonon scattering intensity to obtain the complete map. nrots is based on the space group of each system. This step is important to avoid the extra effort of calculating the scattering intensity for wavevectors connected by rotation symmetry (see also plots at the end of the exercise). (ii) kres1 and kres2 define the resolution of the scattering intensity along the chosen Cartesian axes, (iii) col1 and col2 are integers defining two of the Cartesian directions of the scattering vectors (where 1 --> x, 2 --> y, and 3 --> z), (iv) (kmin - kmax) define the 2D momentum window in reciprocal space, and (v) Np is the number of reduced wavevectors (**q**-points) used to sample each Brillouin zone.

The standard outputs from a disca.x run are: **strf_one-ph_broad.dat** and **strf_all-ph_broad.dat** containing the zero+one-phonon, and all-phonon scattering intensities, respectively, after convolution is applied. These files have three columns: the first two represent the Cartesian coordinates of the scattering vectors and the third column is the scattering intensity.

One can also print for each case the mode resolved and atom resolved scattering intensities by setting mode_resolved = .true. and/or atom_resolved = .true..

For printing the raw data, i.e. before convolution is applied, one can use print_raw_data = .true. to obtain the following data: **Bragg_scattering.dat**, **strf_one-phonon.dat**, **strf_q_nu_one-phonon.dat**, and **strf_all-phonon.dat**, containing the Bragg, mode (or branch) resolved, one-phonon, and all-phonon scattering intensities, respectively. These files have four columns: the first three represent the Cartesian coordinates of the scattering vectors (Q_x, Q_y, and Q_z) and the fourth column is the scattering intensity. The code pp_disca.x in EPW/ZG/src can be used separately to convolute the raw data. disca.x also prints the **Q** vectors in crystal coordinates in **qpts_strf.dat** (to be used for computing the ZG scattering intensity).

Now copy the output file strf_all-ph_broad.dat in the gnuplot directory:

```
$ cp strf_all-ph_broad.dat ../../gnuplot/exact/
```

▶ Run a ZG.x calculation that allows to compute the structure factor. Go to the ZG directory and copy the input files:

```
$ cd ../ZG; cp ../../inputs/ZG/graphene.881.fc .; cp ../../inputs/ZG/ZG_strf.in .
$ cp ../exact/qpts_strf.dat .
```

Note: we also copied qpts_strf.dat containing the **Q**-points generated by disca.x.

The ZG_strf.in should look like this:

```
--                                                                                ZG_strf.in
&input
    asr='all', amass(1)= 12.011, flfrc= 'graphene.881.fc', atm_zg(1) = 'C'
    dim1 = 40, dim2 = 40, dim3 = 1,
    T = 300,
    synch = .true., compute_error = .false.
    incl_qA = .true.,
    ZG_strf = .true.
/
&strf_ZG
    qpts_strf = 40000
    atmsf_a(1,1) = 0.1361, atmsf_a(1,2) = 0.5482, atmsf_a(1,3) = 1.2266,
    atmsf_a(1,4) = 0.5971, atmsf_a(1,5) = 0.0,
    atmsf_b(1,1) = 0.3731, atmsf_b(1,2) = 3.2814, atmsf_b(1,3) = 13.0456,
    atmsf_b(1,4) = 41.0202, atmsf_b(1,5) = 0.0,
    nrots = 6,
    kres1 = 250,
    kres2 = 250,
    kmin  = -10,
    kmax  = 10,
    col1  = 1,
    col2  = 2,
    Np    = 1600
/
```

```
$ ibrun -np 48 $QE/ZG.x -nk 48 < ZG_strf.in > ZG_strf.out
```

The input variables in the first 3 lines of ZG_strf.in are the same as in disca.in (but strictly speaking dimX, here, is for the supercell size dimensions); in line 4 we ask the code to synchronize the modes (synch = .true.), but not to compute the minimization function $E(\{S_{\mathbf{q},\nu}\})$, as we have seen in **Exercise 1** (compute_error = .false.). We made this choice to demonstrate that the order of choosing the unique set of signs is not important as we approach the thermodynamic limit; in line 5 we make the choice to include $\mathbf{q}$-points in set $\mathcal{A}$ by setting incl_qA = .true. (at the thermodynamic limit should make no difference); in line 6 we ask the code to compute the structure factor (ZG_strf = .true.).

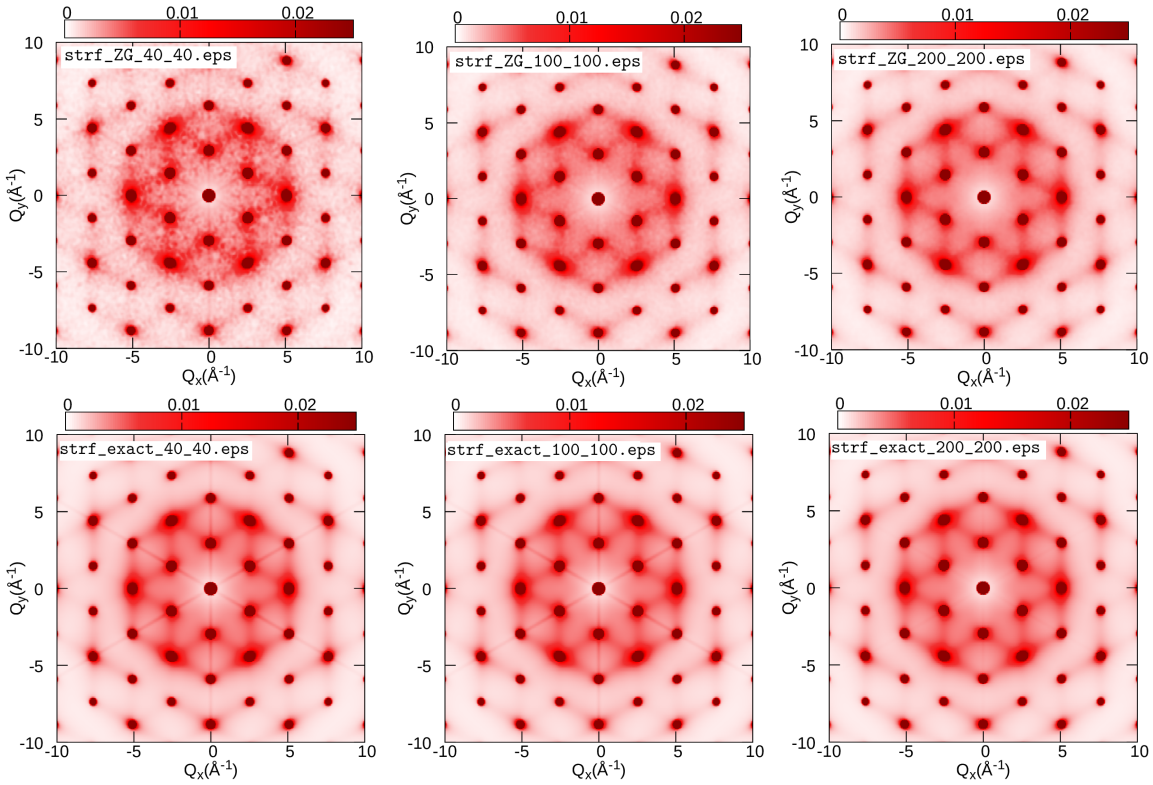
The namelist &strf_ZG is as &pp_disca seen before, but here we include the parameters of the atomic scattering factor (atmsf_a and atmsf_b). We also ask the code to compute the map for 40000 **Q**-points (qpts_strf = 40000) from file qpts_strf.dat.

The calculation should take around 10 secs. The important outputs from this ZG.x run are: **ZG-configuration_300.00K.dat** and **strf_ZG_broad.dat**, containing the collection of ZG atomic coordinates (as before) and the associated scattering intensity, respectively. File **structure_factor_ZG_raw.dat** contains the raw data and has four columns: the first three represent the Cartesian coordinates of the scattering vectors and the fourth column is the scattering intensity.

Now copy the output file strf_ZG_broad.dat in the gnuplot directory and plot your ZG and exact data:

```
$  cp strf_ZG_broad.dat ../../gnuplot/ZG
$  cd ../../gnuplot/ZG; gnuplot gp_pm3d.p; evince strf_ZG_40_40.pdf
$  cd ../exact; gnuplot gp_pm3d.p; evince strf_exact_40_40.pdf
```

Your results should look like the plots below (left panel). We also add plots of the exact and ZG all-phonon structure maps using a 100×100 and 200×200 **q**-grid for Brillouin sampling What happens to the ZG structure factor map as we approach the thermodynamic limit? Indeed, the choice of the order of signs is no longer important and the function $E(\{S_{\mathbf{q},\nu}\})$ is minimized at the thermodynamic limit. Our calculations here also demonstrate the physical significance of the ZG configuration as the best collection of scatterers that reproduce the phonon-induced inelastic scattering.



Exercise 6

In this exercise we will learn how to perform phonon unfolding. This technique is useful to identify the effect on the phonons in the fundamental Brillouin zone when supercells are employed to accommodate disorder, defects, or any other perturbation that breaks the translational symmetry of the lattice. First, as a proof of concept, we will perform the perfect unfolding of the phonons calculated for a 2×2×2 silicon supercell; then we will show how to calculate the effect on the phonons when a 2×2×2 silicon supercell is doped by a phosphorous (P) atom.

▶ Go to `exercise6`, create `workdir`, and calculate the phonon dispersion of silicon in the unit cell using a 2×2×2 homogeneous $\mathbf{q}$-grid:

```
$  cd workdir; cp -r ../inputs/unit_cell_222_grid/ .; cd unit_cell_222_grid/
$ ibrun -np 4 $QE/pw.x -nk 4 < si.scf.in > si.scf.out
$ ibrun -np 4 $QE/ph.x -nk 4 < si.ph.in > si.ph.out
$ ibrun -np 1 $QE/q2r.x < q2r.in > q2r.out
$ ibrun -np 1 $QE/matdyn.x < matdyn.in > matdyn.out
```

Input files are provided here:

```
--                                                              si.scf.in
 &control
    calculation = 'scf'
    restart_mode = 'from_scratch',
    prefix = 'si',
    pseudo_dir = './',
    outdir='./'
 /
 &system
    ibrav = 2,
    celldm(1) = 10.20,
    nat =  2,
    ntyp = 1,
    ecutwfc = 30.0
 /
 &electrons
    diagonalization='david'
    mixing_mode = 'plain'
    mixing_beta = 0.7
    conv_thr = 1.0d-7
 /
ATOMIC_SPECIES
 Si  28.086  Si.pz-vbc.UPF
K_POINTS automatic
6 6 6 0 0 0
ATOMIC_POSITIONS {crystal}
 Si 0.00 0.00 0.00
 Si 0.25 0.25 0.25
/
```

```
--                                                              si.ph.in
Phonons of Silicon
&inputph
   amass(1)= 28.0855,
   prefix = 'si'
   outdir = './'
   ldisp = .true.
   fildyn = 'si.dyn'
```

```
   tr2_ph = 1.0d-12
   nq1 = 2
   nq2 = 2
   nq3 = 2
/
```

```
--                                                                  q2r.in
&input
 fildyn='si.dyn', flfrc = 'si.222.fc'
/
```

```
--                                                                matdyn.in
&input
    asr='simple', amass(1)=28.0855,
    flfrc='si.222.fc', fldyn='si.dyn.mat',
    flfrq='si.freq', fleig='si.dyn.eig', q_in_cryst_coord = .false.
    q_in_band_form = .true.
/
 9
 0         0         0      100
 0.75      0.75      0.0    1
 0.2500    1.0000    0.2500 100
 0.000     1         0.000  100
 0         0         0      100
 0.5000    0.5000    0.5000 100
 0         1         0      100
 0.5000    1.0000    0      100
 0.5000    0.5000    0.5000 100
/
```

▶ After the calculation is completed use `plotband.x` of Quantum Espresso to obtain the phonon dispersion data as an a xmgr file.

```
$  $QE/plotband.x
```

```
    Input file > si.freq
Reading    6 bands at    702 k-points
Range:    0.0000  510.0113eV  Emin, Emax, [firstk, lastk] > 0 600
high-symmetry point:  0.0000 0.0000 0.0000   x coordinate   0.0000
high-symmetry point:  0.7500 0.7500 0.0000   x coordinate   1.0607
...
high-symmetry point:  0.5000 0.5000 0.5000   x coordinate   5.3534
output file (gnuplot/xmgr) > si_ph_dispersion.xmgr
bands in gnuplot/xmgr format written to file si_ph_dispersion.xmgr
output file (ps) >
stopping ...
```

▶ Calculate the interatomic force constants of silicon using a $2\times2\times2$ supercell and the Γ-point for the $\mathbf{q}$-grid.

```
$  cd ../; cp -r ../inputs/222_super/ .; cd 222_super
$ ibrun -np 48 $QE/pw.x -nk 4 < si.scf_222.in > si.scf_222.out
$ ibrun -np 48 $QE/ph.x -nk 4 < si.ph_222.in > si.ph_222.out
$ ibrun -np 1 $QE/q2r.x < q2r.in > q2r.out
```

Input files are provided here:

```
--                                                                    si.scf_222.in
 &control
    calculation = 'scf'
    restart_mode = 'from_scratch',
    prefix = 'si',
    pseudo_dir = './',
    outdir='./'
 /
 &system
    ibrav = 2,
    celldm(1) = 20.40,
    nat =  16,
    ntyp = 1,
    ecutwfc = 20.0
 /
 &electrons
    diagonalization='david'
    mixing_mode = 'plain'
    mixing_beta = 0.7
    conv_thr = 1.0d-7
 /
ATOMIC_SPECIES
 Si  28.086  Si.pz-vbc.UPF
K_POINTS automatic
3 3 3 0 0 0
ATOMIC_POSITIONS (crystal)
Si   0.00000000   0.00000000   0.00000000
Si  -0.12500000   0.37500000  -0.12500000
Si   0.50000000   0.00000000   0.00000000
Si   0.37500000   0.37500000  -0.12500000
Si   0.00000000   0.50000000   0.00000000
Si  -0.12500000   0.87500000  -0.12500000
Si   0.50000000   0.50000000   0.00000000
Si   0.37500000   0.87500000  -0.12500000
Si   0.00000000   0.00000000   0.50000000
Si  -0.12500000   0.37500000   0.37500000
Si   0.50000000   0.00000000   0.50000000
Si   0.37500000   0.37500000   0.37500000
Si   0.00000000   0.50000000   0.50000000
Si  -0.12500000   0.87500000   0.37500000
Si   0.50000000   0.50000000   0.50000000
Si   0.37500000   0.87500000   0.37500000
```

```
--                                                                       si.ph.in
Phonons of Silicon
&inputph
   amass(1)= 28.0855,
   prefix = 'si'
   outdir = './'
   ldisp = .true.
   fildyn = 'si.dyn'
   tr2_ph = 1.0d-12
   nq1 = 1, nq2 = 1, nq3 = 1
/
```

```
--                                                                       q2r.in
&input
 fildyn='si.dyn', flfrc = 'si.222.fc'
/
```

▶ Perform phonon unfolding of the phonons of silicon calculated for a 2×2×2 supercell using `ZG.x`

```
$ ibrun -np 48 $QE/ZG.x -nk 48 < ZG_ph_unfold.in > ZG_ph_unfold.out
```

where the input file should look like:

```
--                                                              ZG_ph_unfold.in
&input
    flfrc='si.222.fc',
    asr='simple', amass(1)=28.0855
    q_in_cryst_coord = .false., q_in_band_form = .true.
    ZG_conf = .false., ph_unfold = .true.
/
&phonon_unfold
     dim1 = 2, dim2 = 2, dim3 = 2
     ng1 = 14, ng2 = 14, ng3 = 14
     flfrq='frequencies.dat', flweights='unfold_weights.dat'
/
 9
   0.000000   0.000000   0.000000  74
   1.500000   1.500000   0.000000   1
   0.500000   2.000000   0.500000  24
   0.000000   2.000000   0.000000  70
   0.000000   0.000000   0.000000  60
   1.000000   1.000000   1.000000  60
   0.000000   2.000000   0.000000  34
   1.000000   2.000000   0.000000  50
   1.000000   1.000000   1.000000   1
/
```

Here, lines 3-5 contain standard inputs for a phonon dispersion calculation (as for `matdyn.x`). In line 4, we specify that we would like to use `ZG.x` for phonon unfolding (`ph_unfold = .true.`) and not to generate special displacements (`ZG_conf = .false.`). In the namelist `&phonon_unfold` we have: (i) `dimX` which are essentially the supercell dimensions, (ii) `ngX` define the range of the reciprocal lattice vectors (and thus plane waves) that need to be employed for calculating the spectral weights. Check for convergence by increasing this number; already 14 is a tight value, (iii) `flfrq` is the name of the output file containing the frequencies for each band and $\mathbf{Q}$-point and (iv) `flweights` is the name of the output file containing the spectral weights for each band and $\mathbf{Q}$-point. At the end of the file we provide the high-symmetry path for which we would like to perform phonon unfolding. The input structure of path is in the same spirit with a `matdyn.x` calculation. As for the electron band structure unfolding, the coordinates of the $\mathbf{Q}$-points are obtained by multiplying the $\mathbf{q}$ wavevectors (defining the fundamental Brillouin zone) with the dimensions of the supercell, i.e. if $\mathbf{q} = [x \;\; y \;\; z]$ then $\mathbf{Q} = [\mathrm{dim1} \times x \quad \mathrm{dim2} \times y \quad \mathrm{dim3} \times z]$.

▶ Now we have the phonon frequencies and their spectral weights, we can calculate the phonon spectral function using `pp_spctrlfn.x`. Convert first the raw data in **frequencies.dat** and **unfold_weights.dat** into a different format using `plotband.x` of QE (commands provided in `bands.sh`):

```
$ $QE/plotband.x unfold_weights.dat < spectral_weights.in > spectral_weights.out
$ $QE/plotband.x frequencies.dat < energies.in > energies.out
$ sed -i '/^\s*$/d' spectral_weights.dat # remove empty lines
$ sed -i '/^\s*$/d' energies.dat
$ paste energies.dat spectral_weights.dat > tmp
$ awk '{print $1,$2,$4}' tmp > energies_weights.dat; rm tmp
```

You have just generated **energies_weights.dat** file in a similar format to a ".gnu" file where the first column has the momentum-axis values, the second the phonon frequencies, and the third the spectral weights. Now proceed with the calculation of the spectral function:

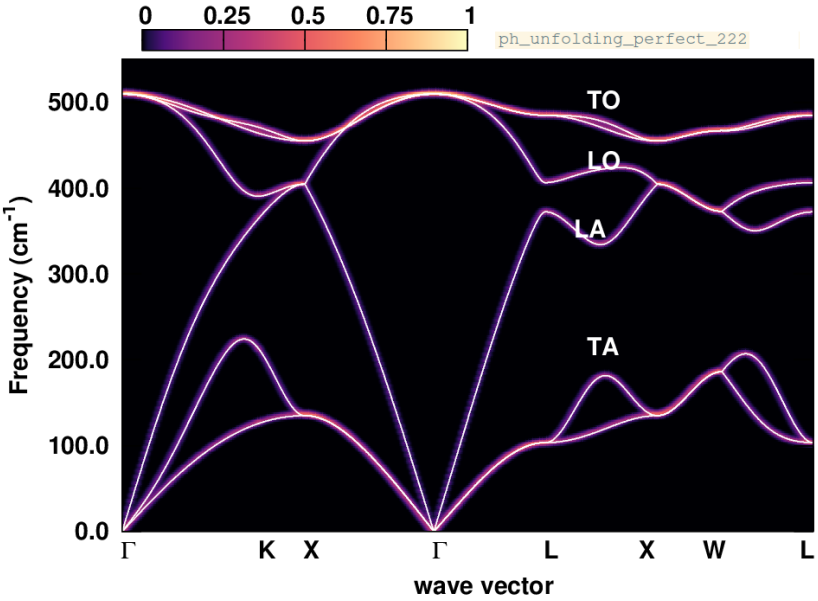
```
$ ibrun -n 28 $QE/pp_spctrlfn.x -nk 28 < pp_spctrlfn.in > pp_spctrlfn.out
$ cp spectral_function.dat ../../gnuplot/spectral_function_222.dat
$ cd ../../gnuplot/; gnuplot gp_pm3d_222.p; evince ph_unfolding_perfect_222.pdf
```

where the input file should look like:

pp_spctrlfn.in

```
&input
  flin = 'energies_weights.dat'
  steps = 17952,
  ksteps = 300,
  esteps = 300,
  kmin = 0,
  kmax = 5.3534,
  emin =  -10
  emax = 550
  flspfn = 'spectral_function.dat'
/
```

Note: The description of the input flags is available in Exercise 2.



In this plot, the silicon phonon spectral function (color map) overlays with the phonon dispersion calculated in the unit cell, since the supercell is an exact periodic replica of the unit cell. One can also check the raw data in `unfold_weights.dat`, and observe that the weights of the phonon frequencies are either 1 or 0, making an one-to-one correspondence with the frequencies calculated in the unit cell.

▶ Now go to `workdir`, calculate the phonons of P-doped silicon, and perform phonon unfolding:

```
$ cd ../workdir/;cp -r ../inputs/222_super_P_doped .; cd 222_super_P_doped/
$ ibrun -np 48 $QE/pw.x -nk 4 < si.scf_222.in > si.scf_222.out
$ ibrun -np 48 $QE/ph.x -nk 4 < si.ph_222.in > si.ph_222.out
$ ibrun -np 1 $QE/q2r.x < q2r.in > q2r.out
$ ibrun -np 48 $QE/ZG.x -nk 48 < ZG_ph_unfold.in > ZG_ph_unfold.out
```

Note: Here we replace one Si atom with a P atom and provide the relaxed structure. Due to the excess electron the system becomes metallic.

Here we provide all input files:

```
--                                                                    si.scf_222.in
 &control
    calculation = 'scf'
    restart_mode = 'from_scratch',
    prefix = 'si',
    pseudo_dir = './',
    outdir='./'
 /
 &system
    ibrav = 2,
    celldm(1) = 20.40,
    nat =  16,
    ntyp = 2,
    ecutwfc = 30.0
    occupations='smearing'
    degauss = 0.01
    nosym =.true.
 /
 &electrons
    diagonalization='david'
    mixing_mode = 'plain'
    mixing_beta = 0.7
    conv_thr = 1.0d-7
 /
ATOMIC_SPECIES
 Si  28.086  Si.pz-vbc.UPF
 P  30.974   P.pz-hgh.UPF
K_POINTS automatic
3 3 3 0 0 0
ATOMIC_POSITIONS (crystal)
P             0.0000000002       -0.0000000243        0.0000000412
Si           -0.1261235427        0.3783706126       -0.1261235364
Si            0.4983582216        0.0016417759        0.0016417825
Si            0.3751699490        0.3751699540       -0.1255098692
Si            0.0016417706        0.4983582252        0.0016417800
Si           -0.1261235463        0.8738764598       -0.1261235454
Si            0.4983582134        0.4983582133        0.0016417907
Si            0.3783706477        0.8738764517       -0.1261235465
Si            0.0016417985        0.0016418052        0.4983581945
Si           -0.1255098925        0.3751699883        0.3751699378
Si            0.4983582056        0.0016417944        0.4983582063
Si            0.3751699659        0.3751699770        0.3751699562
Si            0.0016417902        0.4983582115        0.4983582106
Si           -0.1261235587        0.8738764426        0.3783706794
Si            0.4999999969        0.5000000190        0.4999999746
Si            0.3751699807        0.8744900937        0.3751699437
/
```

```
--                                                                    si.ph_222.in
&inputph
   amass(1)= 28.0855,
   amass(2)= 30.974,
   prefix = 'si'
   outdir = './'
   ldisp = .true.
   fildyn = 'si.dyn'
   tr2_ph = 1.0d-12
```

```
   epsil = .false.
   nq1 = 1, nq2 = 1, nq3 = 1
/
```

```
--                                                                    q2r.in
&input
 fildyn='si.dyn', flfrc = 'si.222.fc'
/
```

```
--                                                                    ZG_ph_unfold.in
&input
    flfrc='si.222.fc',
    asr='simple', amass(1)=28.0855, amass(2)= 30.974
    q_in_cryst_coord = .false., q_in_band_form = .true.
    ZG_conf = .false.
    ph_unfold = .true.
/
&phonon_unfold
     dim1 = 2, dim2 = 2, dim3 = 2
     ng1 = 14, ng2 = 14, ng3 = 14
     flfrq='frequencies.dat', flweights='unfold_weights.dat'
/
 9
   0.000000   0.000000   0.000000  74
   1.500000   1.500000   0.000000   1
   0.500000   2.000000   0.500000  24
   0.000000   2.000000   0.000000  70
   0.000000   0.000000   0.000000  60
   1.000000   1.000000   1.000000  60
   0.000000   2.000000   0.000000  34
   1.000000   2.000000   0.000000  50
   1.000000   1.000000   1.000000   1
```

▶ The calculation should take around 4.5 minutes. Now we have the phonon frequencies and their spectral weights, we can calculate the phonon spectral function using `pp_spctrlfn.x` as before. Convert first the raw data in **frequencies.dat** and **unfold_weights.dat** into a different format using `plotband.x` of QE (commands provided in `bands.sh`):

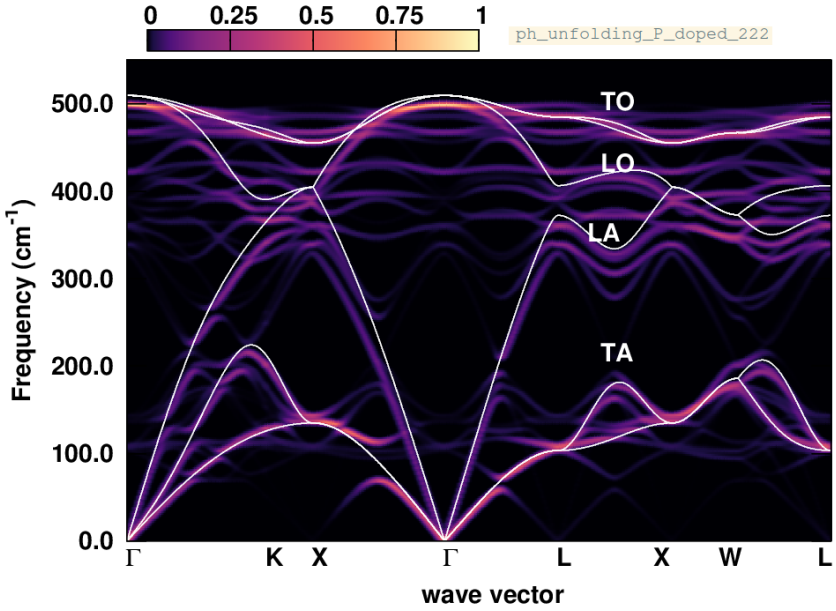
```
$ $QE/plotband.x unfold_weights.dat < spectral_weights.in > spectral_weights.out
$ $QE/plotband.x frequencies.dat < energies.in > energies.out
$ sed -i '/^\s*$/d' spectral_weights.dat # remove empty lines
$ sed -i '/^\s*$/d' energies.dat
$ paste energies.dat spectral_weights.dat > tmp
$ awk '{print $1,$2,$4}' tmp > energies_weights.dat; rm tmp
```

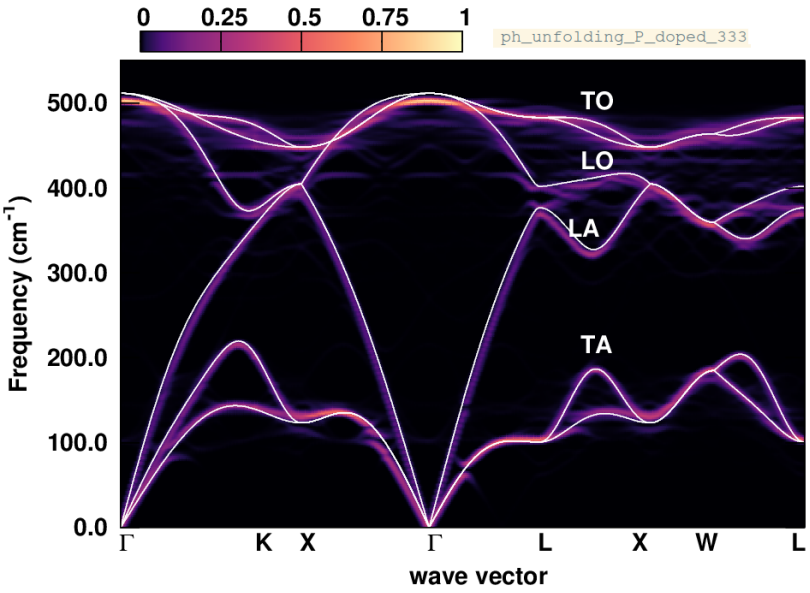
▶ Now proceed with the calculation of the spectral function:

```
$ ibrun -n 28 $QE/pp_spctrlfn.x -nk 28 < pp_spctrlfn.in > pp_spctrlfn.out
$ cp spectral_function.dat ../../gnuplot/spectral_function_222_P_doped.dat
$ cd ../../gnuplot/; gnuplot gp_pm3d_222_P_doped.p
$ evince ph_unfolding_P_doped_222.pdf
```

where the input file `pp_spctrlfn.in` is the same as before.

What are the changes in the phonon dispersion? Below we also provide the phonon spectral function calculated after replacing one Si atom with a P atom in a 3×3×3 supercell. Why are the changes less pronounced now?


0 0.25 0.5 0.75 1
ph_unfolding_P_doped_222
500.0
400.0
300.0
200.0
100.0
0.0
Frequency (cm-1)
TO
LO
LA
TA
Γ K X Γ L X W L
wave vector


0 0.25 0.5 0.75 1
ph_unfolding_P_doped_333
500.0
400.0
300.0
200.0
100.0
0.0
Frequency (cm-1)
TO
LO
LA
TA
Γ K X Γ L X W L
wave vector

Exercise Diamond

In this exercise we will learn how to calculate temperature-dependent gap renormalizations in the harmonic approximation with SDM, using the example of the direct gap of diamond. We will employ 4×4×4 ZG supercells. We will also show how to calculate the free energy of the system as a function of temperature. To handle multiple temperatures, we introduce the flag `T_array(X)` in the ZG code. For consistency reasons, the code assigns the first choice of signs generated without minimizing the error function $E(\{S_{\mathbf{q},\nu}\})$, i.e. sets `compute_error = .false.`. A prerequisite for understanding the workflow in this exercise is to complete Exercise 1 for silicon.

▶ Go to the directory `exerciseC`, create `workdir`, copy `script.sh` from `inputs`, and submit:

```
$  cd exerciseC/; mkdir workdir; cd workdir
$  cp ../inputs/script.sh .
$  sbatch script.sh
```

The script will (i) run scf and DFPT phonon calculations to obtain the harmonic phonons of diamond within the unit cell, (ii) generate ZG configurations for multiple temperatures with `ZG.x`, (iii) run scf calculations for the equilibrium and ZG structures, (iv) prepare and run nscf calculations for the equilibrium and ZG structures, and (v) extract the temperature dependent direct gap. The commands and workflow in the submission script **script.sh** are provided in **APPENDIX / Exercise C**. The script should take around 18 minutes to finish, so while it's running proceed to the following.

The files **scf.in**, **ph.in**, **q2r.in**, and **matdyn.in** are provided here:

```
----                                                    scf.in
 &control
    calculation = 'scf'
    restart_mode = 'from_scratch',
    prefix = 'scf',
    pseudo_dir = './',
    outdir='./'
 /
 &system
    ibrav = 2,
    celldm(1) = 6.75,
    nat =  2,
    ntyp = 1,
    ecutwfc = 50.0
 /
 &electrons
    diagonalization='david'
    mixing_mode = 'plain'
    mixing_beta = 0.7
    conv_thr = 1.0d-7
 /
ATOMIC_SPECIES
 C  12.011  C.upf
K_POINTS automatic
6 6 6 0 0 0
ATOMIC_POSITIONS {crystal}
 C 0.00 0.00 0.00
 C 0.25 0.25 0.25
```

```
----                                                    ph.in
&inputph
   amass(1)= 12.011,
   prefix = 'scf'
   outdir = './'
   fildyn = 'C.dyn'
   tr2_ph = 1.0d-12, ldisp = .true.
```

```
   nq1 = 2,   nq2 = 2,   nq3 = 2
/
```

```
--                                                  q2r.in
&input
 fildyn='C.dyn', flfrc = 'C.fc'
/
```

```
--                                                 matdyn.in
&input
    asr='all', amass(1)= 12.011,
    flfrc='C.fc', fldyn='C.dyn.mat', flfrq='C.freq', fleig='C.dyn.eig',
    q_in_cryst_coord = .false., q_in_band_form = .true.
/
 9
 0       0       0      100
 0.75    0.75    0.0    1
 0.25    1.00    0.25   100
 0.00    1       0.00   100
 0       0       0      100
 0.50    0.50    0.50   100
 0       1       0      100
 0.50    1.00    0      100
 0.50    0.50    0.50   100
```

We also note that the calculations are performed with unconverged settings (e.g. cutoff, k-point and q-point grids). Results for converged settings are provided in Ref. [Phys. Rev. B 94, 075125 (2016)].

The ZG file used (`ZG_444_multi_T.in`) looks like:

```
--                                                  ZG_444_multi_T.in
&input
    flfrc='C.fc', flscf = 'scf.in'
    asr='all', amass(1)=12.011, atm_zg(1) = 'C',
    T_array(1) = 0, T_array(2) = 50, T_array(3) = 100, T_array(4) = 150
    T_array(5) = 200, T_array(6) = 250, T_array(7) = 300,  T_array(8) = 350,
    T_array(9) = 400, T_array(10) = 600, T_array(11) = 800, T_array(12) = 1000
    dim1 = 3, dim2= 3, dim3 = 3, incl_qA = .false.
    compute_error = .true., synch =.true., error_thresh = 0.1, niters = 100
/
```

Note: **C.fc** contains the harmonic IFCs. `T_array(X)` allows ZG configurations for multiple temperatures.

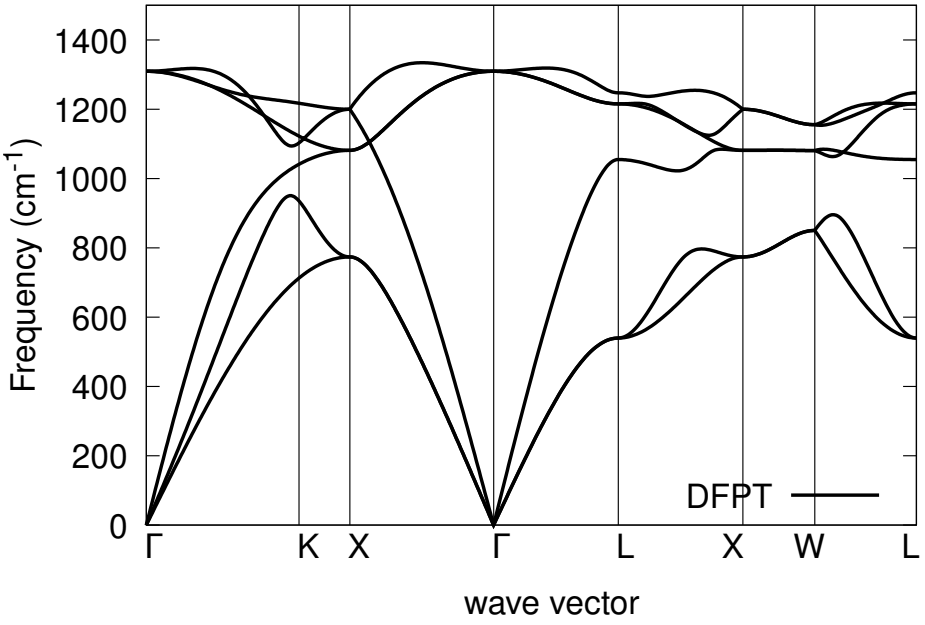
▶ Use `plotband.x` to obtain the data for the phonon dispersion:

`$ $QE/plotband.x`

```
     Input file > C.freq
Reading    6 bands at    702 k-points
Range:    0.0000 1334.1534eV  Emin, Emax, [firstk, lastk] > 0 1400
high-symmetry point:  0.0000 0.0000 0.0000   x coordinate   0.0000
high-symmetry point:  0.7500 0.7500 0.0000   x coordinate   1.0607
high-symmetry point:  0.2500 1.0000 0.2500   x coordinate   1.0607
high-symmetry point:  0.0000 1.0000 0.0000   x coordinate   1.4142
high-symmetry point:  0.0000 0.0000 0.0000   x coordinate   2.4142
high-symmetry point:  0.5000 0.5000 0.5000   x coordinate   3.2802
high-symmetry point:  0.0000 1.0000 0.0000   x coordinate   4.1463
high-symmetry point:  0.5000 1.0000 0.0000   x coordinate   4.6463
high-symmetry point:  0.5000 0.5000 0.5000   x coordinate   5.3534
output file (gnuplot/xmgr) > C_harm_ph_dispersion.xmgr
bands in gnuplot/xmgr format written to file C_harm_ph_dispersion.xmgr
output file (ps) >
stopping ...
```

▶ Now plot the phonon dispersion using:

```
$ gnuplot gp_coms.p
$ evince C_ph_dispersion.pdf
```



▶ Plot the free energy of diamond ZG configurations given by

$$\begin{aligned} F &= \langle U \rangle_T - T\,S \\ &= U^{\tau_{\rm ZG}} - \frac{M_0}{2}\sum_{\mathbf{q}\nu}\omega^2_{\mathbf{q}\nu}\sigma^2_{\mathbf{q}\nu}(T) + \sum_{\mathbf{q}\nu}\left[\frac{\hbar\omega_{\mathbf{q}\nu}}{2} - k_{\rm B}T\ln[1+n_{\mathbf{q}\nu}(T)]\right], \end{aligned} \tag{5}$$

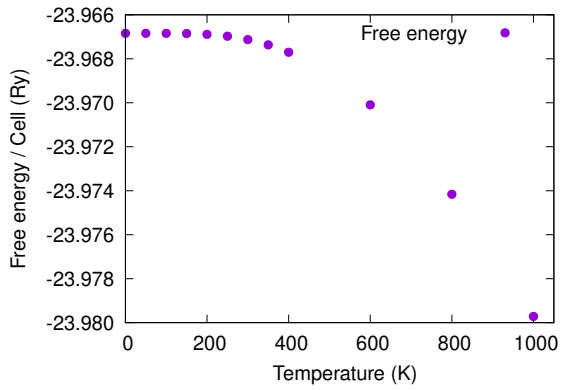
where U is the total energy of the system and S is the entropy. The last line is essentially Eq. (10) of Ref. [Phys. Rev. B **108**, 035155 (2023)]; for definitions please check therein. The first term on the right hand side is the total energy of the system at thermal equilibrium evaluated with the nuclei clamped at their ZG coordinates, the second term (without the sign) is the potential energy and the third term is the vibrational free energy. The first term is obtained after a DFT calculation using the ZG configuration and the last two terms are printed by the code in **ZG_444_multi_T.out**.

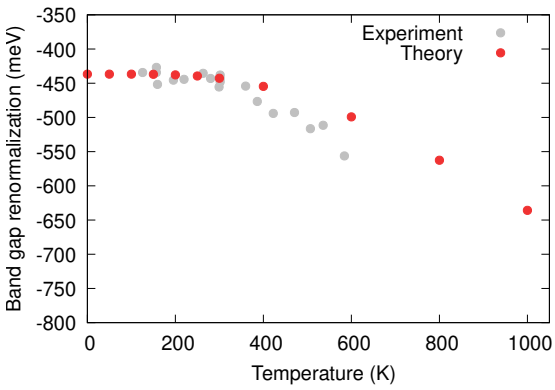
▶ Run the commands below one by one and spot each term in the outputs:

```
$  grep "Potential" ZG_444_multi_T.out | awk '{print $3}' > Pot_ene.dat
$  grep "free ene" ZG_444_multi_T.out | awk '{print $4}' > Vib_free_ene.dat
$  grep ! ZG-scf_444_* | awk '{print $1,$5}' > tmp
$  sed 's/ZG-scf_444_//g' tmp | sed 's/K.out:!//g' > tmp2 # remove strings
$  sort -k1n tmp2 > Tot_ene.dat
$  paste Tot_ene.dat Pot_ene.dat Vib_free_ene.dat > data.dat
$  gnuplot gp_coms_free.p; evince C_free_energy.pdf
```

▶ If the calculation is completed, plot the direct gap renormalization of diamond and compare with experiment.

```
$  gnuplot gp_coms_gap.p; evince C_Renormalization_T.pdf
```





The DFT direct gap of diamond at static-equilibrium is

```
$  cat eq_direct_gap.dat
   5.5464 # eV
```

The DFT-ZG direct gap of diamond at finite temperatures is:

```
$  cat direct_gap_T_av.dat
# T (K)  Dir. gap (eV)
   0      5.1097
   50     5.1097
   100    5.1097
   150    5.1095
   200    5.1086
   250    5.107
   300    5.1037
   400    5.0918
   600    5.0472
   800    4.9838
   1000   4.9106
```

Here our calculations yield a zero-point renormalization of $\Delta E_{\rm g}^{\rm dir}(0\,{\rm K}) = 5.5464 - 5.1097 = 437$ meV for the direct gap of diamond, in good agreement with literature data reported in Table 1 of Ref. [Phys. Rev. B 94, 075125 (2016)]. To obtain the direct gap we (i) identify that the first direct gap of diamond lies at the Γ point of the unit cell's Brillouin zone, this can be seen by inspecting the file **bands.out**; both the VBM and CBM defining the direct gap are triply degenerate with values `13.3502` and `18.8956` eV, respectively, (ii) identify the same states calculated for the supercell with atoms at static-equilibrium in file **equil-nscf_444.out**, (iii) identify the corresponding states calculated for the supercell with atoms at ZG coordinates in files **ZG-nscf_444_T.00K.out**, and (iv) take the average of the corresponding three valence states and the three conduction states in files **ZG-nscf_444_T.00K.out** (see script **com_av_gap.sh** provided in **APPENDIX / Exercise C**). Note that states near the CBM defining the indirect gap of diamond (lying at $\sim$ 0.8 Γ-X of the unit cell's Brillouin zone) are folded at the Γ-point of the 4×4×4 supercell and should be excluded from the analysis (e.g. value `17.6921` eV in **equil-nscf_444.out**).

Our calculations for the temperature-dependent direct gap of diamond compare well with experimental data from Ref. [Phys. Rev. B 46, 4483 (1992)]; note that experimental data are shifted in `gp_coms_gap.p` to ease comparison with theory.

Exercise PbTe

In this exercise we will learn how to calculate temperature-dependent anharmonic phonons of PbTe using 2×2×2 ZG supercells and the A-SDM. Unlike Zr, seen in Exercise 3, PbTe is a polar material and a correction to the dynamical matrix arising from long-range dipole-dipole interactions is essential. Since the A-SDM relies on the computation of IFCs by finite differences, we employ the mixed space approach described in J. Phys. Condens. Matter **22** 202201 (2010), as implemented in `matdyn.x`. The key flags here to be used are `fd = .true.`, `na_ifc = .true.`, and `incl_epsil = .true.`. In this example, the harmonic IFCs matrix is positive definite and therefore the computation of the polymorphous structure is not required to initialize the A-SDM procedure. Furthermore, here we will employ the anharmonic phonons to evaluate the band gap renormalization of PbTe.

▶ Go to the directory `exercisePbTe`, create `workdir`, copy `script.sh`, and submit:

```
$  cd exercisePbTe/; mkdir workdir; cd workdir
$  cp ../inputs/script.sh .
$  sbatch script.sh
```

The script will run scf and DFPT phonon calculations to obtain the (stable) harmonic phonons, and then start the A-SDM procedure to evaluate anharmonic phonons at 300 K. The A-SDM is performed for three iterations. The commands and workflow in the submission script **script.sh** are provided in **APPENDIX / Exercise PbTe**.

The files **scf.in**, **ph.in**, **q2r.in**, and **matdyn.in** are provided here:

```
----                                                scf.in
&control
   calculation='scf'
   restart_mode='from_scratch'
   prefix = 'scf'
   outdir='./'
   pseudo_dir='./'
   tstress=.true.
   tprnfor=.true.
/
&system
   ibrav=2
   space_group = 225
   nat=2
   ntyp=2
   ecutwfc= 50
   A = 6.4440
/
&electrons
   diagonalization='david'
   mixing_mode='plain'
   mixing_beta=0.7
   conv_thr=1e-7
/
ATOMIC_SPECIES
  Pb 207.2 Pb.upf
  Te 127.60 Te.upf
K_POINTS {automatic}
  6 6 6 0 0 0
ATOMIC_POSITIONS (crystal_sg)
    Pb  4b
    Te  4a
```

Note: Here we provide the space group of the crystal, as given in the International Tables of Crystallography A (ITA), and the atomic coordinates in Wyckoff positions. This is an alternative way to specify the geometry of the system in Quantum Espresso which ensures that the symmetry of the crystal is respected.

```
----                                                ph.in
 &inputph
 prefix = 'scf',
 amass(1) = 207.2
 amass(2) = 127.60
 outdir = './'
 verbosity = 'high'
 ldisp = .true.
 fildyn = 'PbTe.dyn'
 tr2_ph = 1.0d-14
 nq1 = 2
 nq2 = 2
 nq3 = 2
/
```

```
--                                               q2r.in
&input
 fildyn='PbTe.dyn', zasr='crystal', flfrc = 'PbTe.222.fc'
/
```

```
--                                             matdyn.in
&input
 asr='all',
 flfrc='PbTe.222.fc', fldyn=' ', flfrq='PbTe_harm.freq',
 fleig=' ', q_in_cryst_coord = .true., q_in_band_form = .true.
/
3
0 0.5 0.5 100
0 0.0 0.0 100
0 0.5 0.5 100
```

We also note that the calculations are performed with unconverged settings (e.g. cutoff and k-point grid). Results for converged settings are provided in https://docs.epw-code.org/doc/TutorialZG.html To spead up the A-SDM process, the script uses calculated output scf files given in folder **fd_forces**. `ZG.x` applies mixing of IFCs between iterations (`mixing = .true.`) and includes information of the Born effective charges and dielectric constant (`incl_epsil = .true.`) computed by DFPT. See for example files **ZG_4.in** and **ZG_6.in**.

▶ Type the following command to see the important files generated.

```
$ ls -lrt *freq *fc
```

```
PbTe.222.fc
PbTe_harm.freq
PbTe_300.00K_01.freq
300.00K_iter_01.fc
PbTe_300.00K_02.freq
300.00K_iter_02.fc
PbTe_300.00K_03.freq
300.00K_iter_03.fc
```

Those are the files containing the harmonic IFCs, A-SDM IFCs, and the corresponding phonon frequencies. The indices `01, 02, 03` indicate the A-SDM iteration.

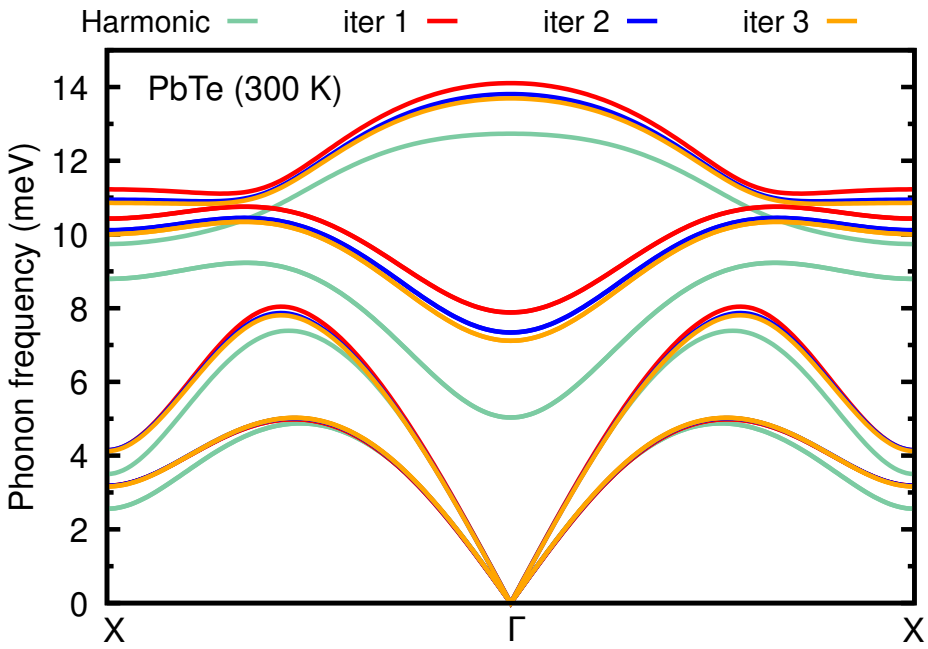
▶ Now use `plotband.x` to obtain the data for the phonon dispersions as xmgr files, i.e.:

```
$ $QE/plotband.x
     Input file > PbTe_harm.freq
Reading    6 bands at    201 k-points
Range:    0.0000  102.6916eV  Emin, Emax, [firstk, lastk] > 0 110
high-symmetry point:  0.0000 1.0000 0.0000   x coordinate   0.0000
high-symmetry point:  0.0000 0.0000 0.0000   x coordinate   1.0000
high-symmetry point:  0.0000 1.0000 0.0000   x coordinate   2.0000
output file (gnuplot/xmgr) > PbTe_harm.xmgr
bands in gnuplot/xmgr format written to file PbTe_harm.xmgr
output file (ps) >
stopping ...
```

Note: We use the name `PbTe_harm.xmgr` for the harmonic case, and the names `iter_01.xmgr`, `iter_02.xmgr`, and `iter_03.xmgr` for anharmonic phonon dispersions.

▶ Now plot the phonon dispersions using:

```
$  gnuplot gp_coms_iters.p
$  evince PbTe_300K_iters_ph_dispersion.pdf
```



The A-SDM is almost converged at the third iteration. If higher cutoff and denser k-grids are used, convergence is achieved even at the first iteration as in https://docs.epw-code.org/doc/TutorialZG.html

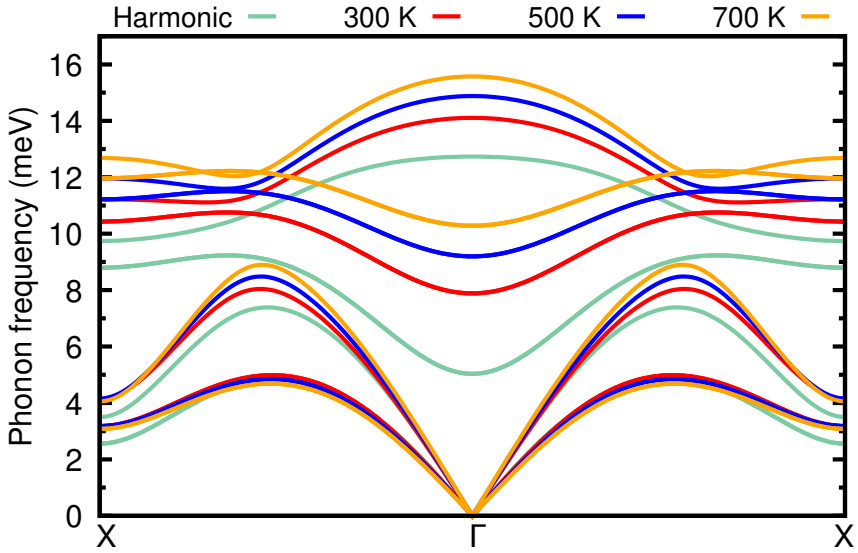
▶ Now calculate the anharmonic phonon dispersion at 500 K using the script `script_ASDM.sh`. The submission script is provided in **APPENDIX / Exercise PbTe**.

```
$ sbatch script_ASDM.sh
```

▶ Repeat the calculation but now for 700 K by changing `T` in the submission script `script_ASDM.sh`.

▶ Now obtain `500K_iter_01.xmgr` and `700K_iter_01.xmgr` from `PbTe_500.00K_01.freq` and `PbTe_700.00K_01.freq` using `plotband.x` (as above) and plot the phonon dispersions:

```
$  gnuplot gp_coms_T.p
$  evince PbTe_T_ph_dispersion.pdf
```



It is evident that anharmonicity is important to describe lattice dynamics in PbTe at elevated temperatures, particularly for the optical modes. Therefore, anharmonicity cannot be neglected in the description of electron-phonon properties in this system. In the following, we are going to calculate the phonon-induced band gap renormalization using anharmonic phonons and compare the result with the one obtained within the harmonic approximation.

▶ Submit the script `script_harm.sh` to evaluate the direct band gap of PbTe within the harmonic approximation at 300, 500, and 700 K using $2 \times 2 \times 2$ ZG configurations. The script runs (i) a `ZG.x` calculation to generate the scf files, (ii) scf calculations for the equilibrium and T-dependent structures, and (iii) nscf calculations with empty bands for the equilibrium and T-dependent structures. The submission script is provided in **APPENDIX / Exercise PbTe**.

```
$ sbatch script_harm.sh
```

The ZG file used (`ZG_harm.in`) is as shown below:

```
--                                                    ZG_harm.in
&input
   asr='all',
   flscf = 'scf.in'
   flfrc='PbTe.222.fc'
   atm_zg(1) = 'Pb',
   atm_zg(2) = 'Te',
   T_array(1) = 300,
   T_array(2) = 500,
   T_array(3) = 700,
   dim1 = 2, dim2 = 2, dim3 = 2, incl_qA = .true.,
   synch = .true.
/
```

Note: **PbTe.222.fc** contains the harmonic IFCs. `T_array(X)` allows ZG configurations for multiple temperatures using the same set of signs generated by `ZG.x`.

The calculations should take around 3 minutes. While the calculations are running, let us discuss some aspects about the choice of the **K**-point (`0.00 0.00 0.00`) made in **Kpoints.txt** for the nscf calculation. The direct gap of cubic PbTe lies at the L-point (`1/2, 1/2, 1/2`) of the fundamental

Brillouin zone (see Materials Project). Therefore, using a $2\times2\times2$ supercell, the bands at the L-point are folded at the Γ-point of the supercell's Brillouin zone. Note that this is not the case for other supercell sizes, e.g. $3\times3\times3$, and therefore a different **K**-point should be chosen.

▶ Extract the band gap of the structure at static equilibrium and at finite temperatures using:

```
$  grep highest equil-nscf_222.out > tmp
$  awk '{print $8-$7}' tmp > equil_gap.dat
$  cat equil_gap.dat
   0.6543
$  ./gap_222.sh # script to obtain T-dependent gaps
$  cp gap_222_T.dat gap_222_T_harm.dat
$  cat gap_222_T_harm.dat
#  T (K)  E (eV)
   300    0.8999
   500    0.99453
   700    1.06219
```

The script `gap_222.sh` employs basic Linux commands to find the direct band gap of PbTe using the bands energy information printed in `ZG-nscf_222_T.00K.out` files. It takes the average of the four valence bands higher in energy and the four conduction bands lower in energy which are degenerate for the equilibrium structure. The degeneracy splitting is an artifact due to finite supercell size effects, as discussed in Exercise 1. The script `gap_222.sh` is provided in **APPENDIX / Exercise PbTe**.

▶ Now submit the script `script_anh.sh` to evaluate the direct band gap of PbTe at 300, 500, and 700 K using $2\times2\times2$ ZG supercells generated using anharmonic phonons. The script runs three `ZG.x` calculations to generate the scf files for each temperature using the corresponding **T.00K_iter_01.fc** files. Then it performs the scf calculations for the equilibrium and T-dependent structures, and nscf calculations with empty bands for the equilibrium and T-dependent structures. The submission script is provided in **APPENDIX / Exercise PbTe**.

```
$ sbatch script_anh.sh
```

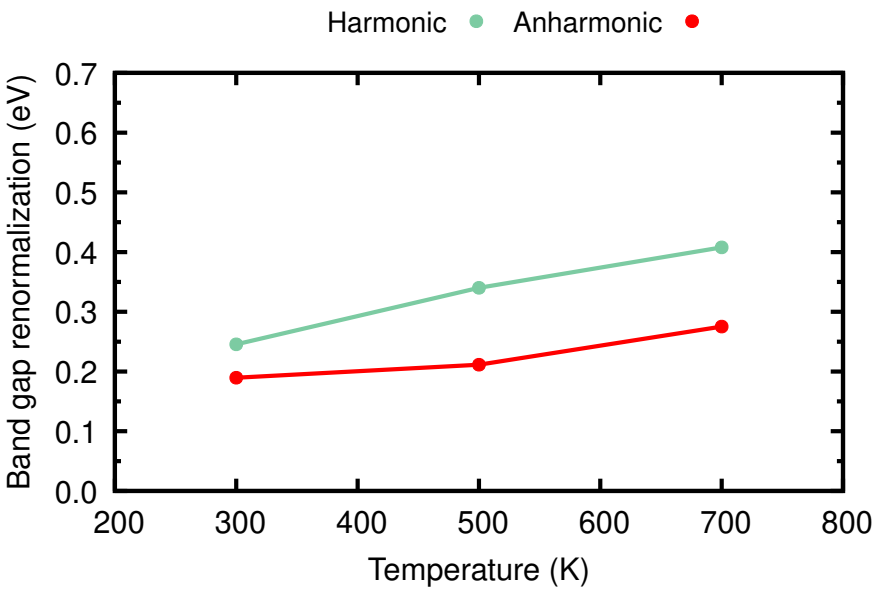
The calculations should take around 3 minutes. Here we provide the **ZG_300K.in**:

```
--                                                    ZG_300K.in
&input
   asr='all',
   flscf = 'scf.in'
   flfrc='300.00K_iter_01.fc'
   atm_zg(1) = 'Pb',
   atm_zg(2) = 'Te',
   T = 300,
   dim1 = 2, dim2 = 2, dim3 = 2, incl_qA = .true.
   compute_error = .true., synch = .true., error_thresh = 0.4
   na_ifc = .true., fd =.true.
/
```

Note: Unlike file **ZG_harm.in** , here we provide only one temperature per ZG run since **300.00K_iter_01.fc** corresponds to anharmonic phonons computed for 300 K. In addition, we add the flags `fd = .true.`, `na_ifc = .true.` to account for long range effects since the IFCs in **300.00K_iter_01.fc** are computed by finite differences. For the same reason, the flags `fd = .true.` and `na_ifc = .true.` are used in **matdyn_0X.in** (X = 01, 02, 03), **ZG_X.in** (X = 3, 4, 5, 6), and **ZG_XK.in** (X = 300, 400, 500). This is not the case for file **ZG_harm.in**, since initial harmonic IFCs (**PbTe.222.fc**) are computed by DFPT.

▶ Extract the direct band gap of PbTe at finite temperatures and compare it with the harmonic case using:

```
$  ./gap_222.sh # script to obtain T-dependent gaps
$  cat gap_222_T.dat
#  T (K)  E (eV)
   300    0.84392
   500    0.86571
   700    0.92958
$  gnuplot gp_coms_gap.p; evince PbTe_Renormalization_T.pdf
```



Here we show the band gap renormalization, i.e. we subtract the band gap at static equilibrium (0.654 eV) from the temperature-dependent band gap. At variance with conventional semiconductors, e.g. silicon and diamond, the direct band gap of PbTe increases with temperature which is consistent with exprimental measurements (see Ref. [Appl. Phys. Lett. 103, 262109 (2013)]). With these tutorial settings, anharmonicity leads to a correction of 30%-50% to the band gap renormalization. More accurate values can be obtained using larger ZG supercells, larger cutoff, denser grids, and inclusion of spin-orbit coupling.

Exercise $CsPbBr_3$

In this exercise we will explore anharmonic electron-phonon coupling in cubic $CsPbBr_3$ (inorganic metal halide perovskite) by calculating the band gap renormalization, following Ref. [npj Comput Mater 9, 153 (2023)]. Here, anharmonic phonons are calculated by the A-SDM and used to generate ZG displacements for the monomorphous (high-symmetry) and ground state polymorphous (low-symmetry) structures in 2×2×2 supercells. The difference in the computed band gap renormalization for the two cases is strikking, showing that polymorphism (or local disorder) plays a key role in predicting accurately electron-phonon properties in cubic perovskites.

▶ Go to `exerciseCsPbBr3`, create `workdir`, copy `script.sh` from `inputs`, and submit:

```
$  cd exerciseCsPbBr3/; mkdir workdir; cd workdir
$  cp ../inputs/script.sh .
$  sbatch script.sh
```

Note: `ZG.x` applies mixing of IFCs between iterations (`mixing = .true.`) and includes information of the Born effective charges and dielectric constant (`incl_epsil = .true.`) computed by DFPT at the initial step. In addition, we use the flags `fd = .true.`, `na_ifc = .true.` to account for long range effects since the IFCs at each iteration are computed by finite differences. See for example files **ZG_5.in** and **ZG_7.in**.

The script will run scf and DFPT phonon calculations to obtain the harmonic (unstable) phonons, and then start the A-SDM approach to evaluate anharmonic phonons at 450 K. The A-SDM is performed for (i) the zeroth iteration to obtain the polymorphous structure and its stable phonons and (ii) for the first and second iterations to obtain self-consistent anharmonic phonons. The commands and workflow in the submission script **script.sh** are provided in **APPENDIX / Exercise $CsPbBr_3$**. To spead up the A-SDM, the script uses calculated output scf files given in folder **fd_forces**. The script should take around 4 minutes to finish. The files **scf.in**, **ph.in**, **q2r.in**, **matdyn.in** are provided here:

```
----                                                    scf.in
&control
        calculation='scf'
        restart_mode='from_scratch'
        prefix = 'scf'
        outdir='./'
        pseudo_dir='./'
        tstress=.true., tprnfor=.true.
/
&system
        ibrav=1, A = 5.874, nat=5, ntyp=3,
        ecutwfc=  40
/
&electrons
        diagonalization='david'
        mixing_mode='plain'
        mixing_beta=0.7, conv_thr=1e-6
/
ATOMIC_SPECIES
Cs 132.90545 Cs.upf
Pb 207.2  Pb.upf
Br 79.904 Br.upf
K_POINTS {automatic}
4 4 4 0 0 0
ATOMIC_POSITIONS (crystal)
 Cs  0.500000 0.500000 0.500000
 Pb  0.000000 0.000000 0.000000
 Br  0.000000 0.000000 0.500000
 Br  0.000000 0.500000 0.000000
 Br  0.500000 0.000000 0.000000
```

```
----                                                ph.in
&inputph
   prefix = 'scf'
   outdir = './'
   fildyn = 'CsPbBr3.dyn'
   tr2_ph = 1.0d-12
   ldisp = .true.
   nq1 = 2, nq2 = 2, nq3 = 2
/
```

```
--                                                q2r.in
&input
 fildyn='CsPbBr3.dyn', flfrc = 'CsPbBr3.222.fc'
/
```

```
--                                              matdyn.in
&input
    asr='all',
    flfrc='CsPbBr3.222.fc', fldyn='CsPbBr3.dyn.mat', flfrq='CsPbBr3.freq',
    fleig='CsPbBr3.dyn.eig', q_in_cryst_coord = .true., q_in_band_form = .true.
/
5
0.0000    0.5000    0.000000  100
0.5000    0.5000    0.500000  100
0.5000    0.5000    0.000000  100
0.0000    0.0000    0.000000  100
0.5000    0.5000    0.500000  1
```

▶ Type the following command to see the important files generated.

```
$ ls -lrt *freq *fc
```

```
   CsPbBr3.222.fc # harmonic
   CsPbBr3.freq   # harmonic
   poly_iter_00.fc
   CsPbBr3_poly.freq
   450.00K_iter_01.fc
   CsPbBr3_01.freq
   450.00K_iter_02.fc
   CsPbBr3_02.freq
```

Those are the files containing the harmonic IFCs, A-SDM IFCs, and corresponding phonon frequencies. The indices `00, 01, 02` indicate the A-SDM iteration, with index `00` representing the 0th iteration for the polymorphous structure.

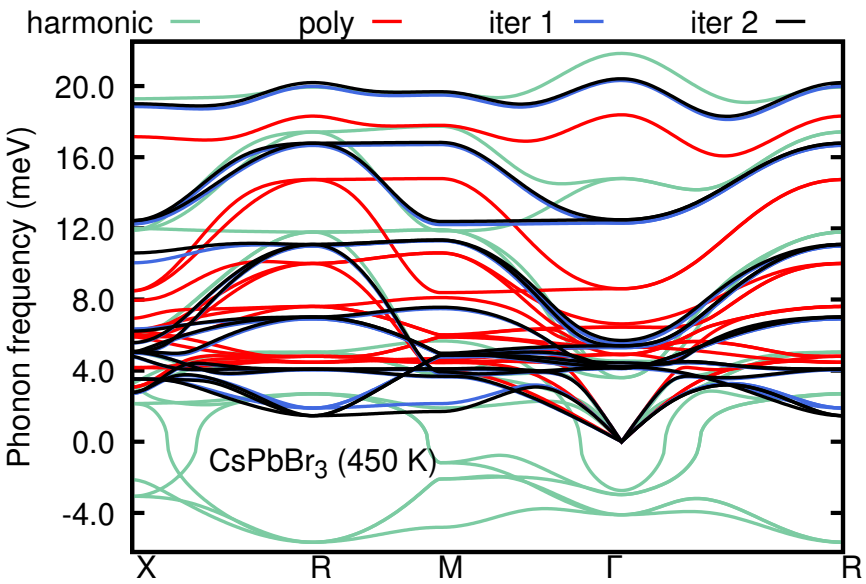
▶ Now use `plotband.x` to obtain the data for the phonon dispersions as xmgr files, e.g.:

```
$ $QE/plotband.x
     Input file > CsPbBr3.freq
Reading   15 bands at    401 k-points
Range:  -45.5651  176.0997eV  Emin, Emax, [firstk, lastk] > -50 180
high-symmetry point:  0.0000 0.5000 0.0000   x coordinate   0.0000
high-symmetry point:  0.5000 0.5000 0.5000   x coordinate   0.7071
...
high-symmetry point:  0.5000 0.5000 0.5000   x coordinate   2.7802
output file (gnuplot/xmgr) > CsPbBr3_harm.xmgr
bands in gnuplot/xmgr format written to file CsPbBr3_harm.xmgr
```

Note: Use the name `CsPbBr3_harm.xmgr` for the harmonic monomorphous case, `CsPbBr3_poly.xmgr` for the polymorphous case, and `iter_01.xmgr` and `iter_02.xmgr` for the A-SDM phonon dispersions up to iteration 2.

▶ Plot the phonon dispersions using:

```
$  cp ../inputs/gp_coms_iters.p .
$  gnuplot gp_coms_iters.p
$  evince CsPbBr3_450K_iters_ph_dispersion.pdf
```



The harmonic phonons computed for the monomorphous structure (green) exhibit large instabilities. The computed symmetrized phonons for the polymorphous structure are stable (red), and corrections to the real part of the phonon self-energy are included via the A-SDM (blue and black). The A-SDM is almost converged at the second iteration. Please refer to Ref. [Phys. Rev. B **108**, 035155 (2023)] for converged settings and results.

▶ Calculate the band gap renormalization of the monomorphous structure at 430, 450, 470, and 500 K using the A-SDM phonons.

```
$  cp ../inputs/script_mono_gap.sh .
$  sbatch script_mono_gap.sh
```

The commands in **script_mono_gap.sh** are provided in **APPENDIX / Exercise CsPbBr$_3$**. The ZG files are generated using the anharmonic IFCs in `450.00K_iter_02.fc` via the input **ZG_multi_T.in**:

```
--                                                    ZG_multi_T.in
&input
   asr='all', flfrc='450.00K_iter_02.fc'
   flscf = 'scf.in'
   T_array(1) = 430, T_array(2) = 450, T_array(3) = 470, T_array(4) = 500
   atm_zg(1) = 'Cs', atm_zg(2) = 'Pb',  atm_zg(3) = 'Br'
   dim1 = 2, dim2 =2, dim3 =2, incl_qA = .true.
   synch = .true.,
   na_ifc = .true., fd = .true.
/
```

Note: For maximum accuracy, one should evaluate the A-SDM phonons for each temperature separately. Here, we assume that phonons from 430 to 500 K do not change significantly.

The script **script_mono_gap.sh** (i) runs scf calculations for the static-equilibrium and ZG configurations of the monomorphous structure, (ii) prepares nscf files via script **mono.sh** provided in **APPENDIX / Exercise CsPbBr$_3$**, and (iii) runs nscf calculations at Γ point to obtain information of the band energies for filled and empty bands. We choose Γ point since the band gap of $CsPbBr_3$ is at the R point (1/2 1/2 1/2) of the fundamental Brillouin zone, and thus folded at the center of the Brillouin zone of a 2×2×2 supercell. All calculations should take around 7 minutes and 30 seconds.

▶ Extract the temperature-dependent band gap and renormalization from the nscf files using:

```
$  mono_gap=$(grep highest equil-nscf_222.out | awk '{print $8-$7}')
$  echo $mono_gap
   1.4377
$  grep highest ZG-nscf_222_*.out > tmp
$  awk '{print $1, $8, $9, $9-$8, $9-$8-'$mono_gap'}' tmp > Renorm_mono_T.dat
$  sed -i 's/ZG-nscf_222_//g' Renorm_mono_T.dat
$  sed -i 's/K.out://g' Renorm_mono_T.dat
$  cat Renorm_mono_T.dat
#  T(K)    VBM(eV) CBM(eV)  Gap(eV)  Renorm. (eV)
  430.00   3.3770  5.2349   1.8579   0.4202
  450.00   3.3623  5.2357   1.8734   0.4357
  470.00   3.3479  5.2365   1.8886   0.4509
  500.00   3.3266  5.2374   1.9108   0.4731
```

▶ Calculate the band gap renormalization at 430, 450, 470, and 500 K using the A-SDM phonons but now displace the atoms of the polymorphous network.

```
$  cp ../inputs/script_poly_gap.sh .
$  sbatch script_poly_gap.sh
```

The commands in **script_poly_gap.sh** are provided in **APPENDIX / Exercise CsPbBr$_3$**. The script (i) prepares scf and nscf files via the script **poly.sh** which is also provided in **APPENDIX / Exercise CsPbBr$_3$**. The displacements (see files **displ_TK.txt**) are obtained by taking the difference between the ZG and high-symmetry coordinates. Then, the displacements are added on the coordinates of the polymorphous structure. The script also (ii) runs scf calculations for the static-equilibrium and ZG configurations of the polymorphous structure, and (iii) runs the correspondng nscf calculations to obtain information of the band energies for filled and empty bands. All calculations should take around 7 minutes and 30 seconds.

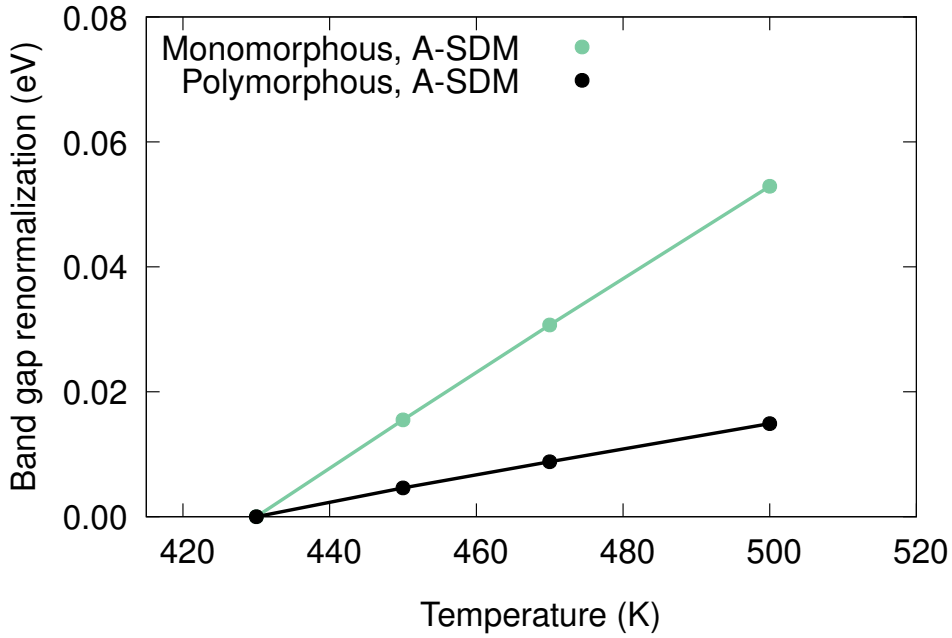
▶ Extract the temperature-dependent band gap and renormalization from the nscf files using:

```
$  poly_gap=$(grep highest ZG-nscf_poly_iter_00.out | awk '{print $8-$7}')
$  echo $poly_gap
    2.0189
$  grep highest ZG-nscf_poly_*K.out > tmp
$  awk '{print $1, $8, $9, $9-$8, $9-$8-'$poly_gap'}' tmp > Renorm_poly_T.dat
$  sed -i 's/ZG-nscf_poly_//g' Renorm_poly_T.dat
$  sed -i 's/K.out://g' Renorm_poly_T.dat
$  cat Renorm_poly_T.dat
#  T(K)    VBM(eV) CBM(eV)  Gap(eV)  Renorm. (eV)
  430.00   2.9337  5.1302   2.1965   0.1776
  450.00   2.9274  5.1285   2.2011   0.1822
  470.00   2.9213  5.1266   2.2053   0.1864
  500.00   2.9121  5.1235   2.2114   0.1925
```

Can you compare first the band gap of the monomorphous and polymorphous structures without the effect of electron-phonon coupling? Polymorphism induces a band gap opening of 0.58 eV in good agreement with the value reported in Table 1 of Ref. [npj Comput Mater 9, 153 (2023)]. Furthermore, the band gap renormalization calculated for the monomorphous and polymorphous structures at 430 K is 420 meV and 178 meV, respectively. These values are in fair agreement with those reported in Ref. [npj Comput Mater 9, 153 (2023)], showing the significant effect of polymorphism on electron-phonon coupling in halide perovskites. Note that here we use a small supercell size and do not include the effect of spin-orbit coupling, which are important for obtaining more accurate results.

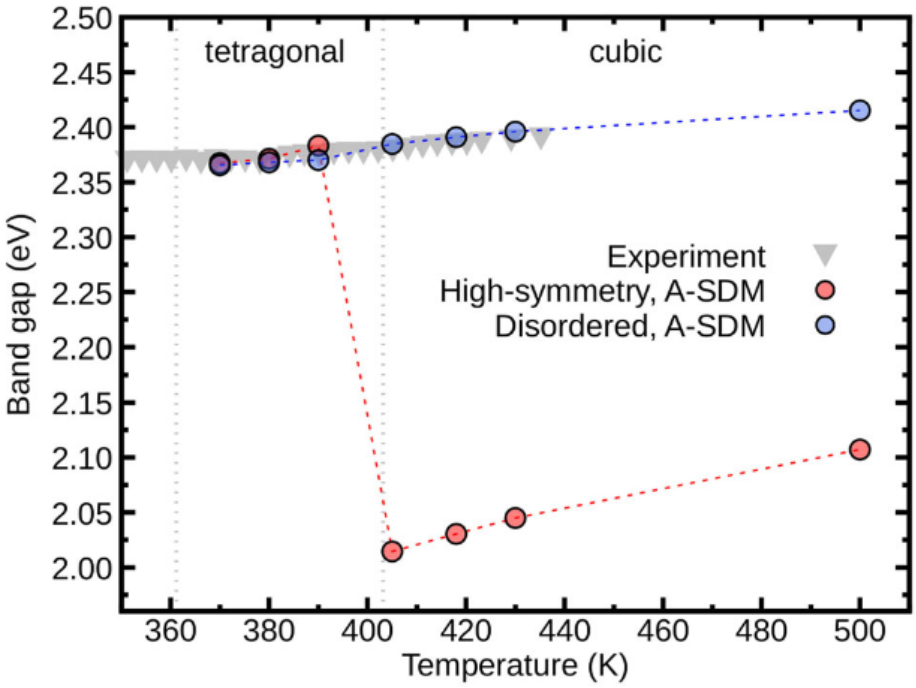
▶ Plot the band gap renormalization for the monomorphous and polymorphous networks:

```
$  cp ../inputs/gp_coms_gap.p .
$  gnuplot gp_coms_gap.p
$  evince CsPbBr3_renormalization.pdf
```



To facilitate comparison, we show the band gap renormalization with respect to the gap at 430 K. The band gap of the polymorphous structure increases much slower with temperature than the one calculated for the monomorphous structure. Evaluating electron-phonon renormalization using the polymorphous structure is crucial for explaining experimental data, especially across phase transitions, as shown in Figure 6 of Ref. [npj Comput Mater 9, 153 (2023)], which is reproduced below.

The success of the polymorphous network is related to the fact that electrons will mostly see the nuclei in the infinitely more stable polymorphous configurations that exist energetically. Can you extract the energy lowering of the polymorphous structure (**ZG-scf_poly_iter_00.out**) with respect to the monomorphous structure (**equil-scf_222.out**)? As homework, you can restart the calculations by setting a different initial temperature in files **ZG_X.in** (X = 1, 2, 3). This will generate different coordinates for the polymorphous structure (but very similar ground state energy). Moreover, the final results obtained in this tutorial should not be affected significantly.



Acknowledgements

Funded by the European Union

This work was also funded by the European Union (project ULTRA-2DPK / HORIZON-MSCA-2022-PF-01 / Grant Agreement No. 101106654). Views and opinions expressed are however those of the author only and do not necessarily reflect those of the European Union or the European Commission. Neither the European Union nor the granting authority can be held responsible for them.

APPENDIX / Exercise 2

In this section of the appendix we show how the band structure unfolding calculation can be broken down to one **K**-point at a time. This strategy is useful when the supercell size is large (larger than $4 \times 4 \times 4$) and the calculation requires a large amount of memory and storage space for the wavefunctions.

▶ Go to `workdir/band_structure_unfolding` of Exercise 2 and copy:

```
$ cp -r ../../../exercise1/workdir/ZG-0.00K-si.save/ .
$ cp ../../inputs/ZG-bands_333_0.00K.in ZG-bands_333_0.00K
$ cp ../../inputs/bands_unfold.in bands.in
```

In `ZG-bands_333_0.00K` apply the following modifications:

1. Set `calculation = 'bands'`
2. Delete the following lines containing information about the **K**-points:

```
K_POINTS crystal_b
2
0.000000 0.000000 0.000000 35
0.000000 1.500000 1.500000 1
```

3. At the bottom of the file add the following (after ZG atomic positions):

```
K_POINTS crystal_b
1
```

The file `ZG-bands_333_0.00K` should look like this:

```
&control                                                    ZG-bands_333_0.00K
  calculation = 'bands'
  restart_mode = 'from_scratch'
  prefix = 'ZG-0.00K-si'
  pseudo_dir = './'
  outdir = './'
/
&system
  ibrav = 0
  nat =   54
  ntyp =    1
  nbnd = 135, nosym = .true., ecutwfc =  20.00
  occupations = 'smearing'
  smearing='gaussian',
  degauss=0.005d0
/
&electrons
  diagonalization = 'david'
  mixing_mode= 'plain'
  mixing_beta = 0.70
  conv_thr =   0.1D-06
/
ATOMIC_SPECIES
   Si   28.085 Si.pz-vbc.UPF
CELL_PARAMETERS {angstrom}
     -8.09641133      0.00000000      8.09641133
      0.00000000      8.09641133      8.09641133
     -8.09641133      8.09641133      0.00000000
```

```
ATOMIC_POSITIONS {angstrom}
...
K_POINTS crystal_b
1
```

▶ Prepare the **K**-point grid for band structure unfolding. To this aim run:

```
$ $PATHQE/EPW/ZG/src/local/kpoints_band_str_unfold.x ! replace PATHQE with path to QE dir
```

This script is designed to generate a list of **K**-points that will unfold back to the fundamental Brillouin zone. You will be asked to provide: (i) the number of high-symmetry **k**-points in the Brillouin zone of the unit-cell, (ii) their coordinates, (iii) their position along the $\mathbf{k}$-axis, (iv) the supercell size, and (v) whether you want to print every single **K**-point. For example:

```
 Write number of high-symmetry kpts
2
 Write high-symmetry kpts (3 columns per row)
0.0 0.0 0.0
0.0 0.5 0.5
 Write x-positions of high-sym kpts
0.0
1.0
 Step is (default):  2.8571429E-02
 Supercell size (n * m * p) ?
3 3 3
 kpts to use for Supercell calculation:
   0.000000   0.000000   0.000000  35
   0.000000   1.500000   1.500000   1
 Write every single kpt? (0=no, 1=yes)
1
...
```

Note: for generating more **K**-points along the high-symmetry paths, you can increase the step value from the script: `$PATHQE/EPW/ZG/src/local/kpoints_band_str_unfold.f90`.

Pass all 35 **K**-points with their weights into a new file `Kpoints.dat`. Otherwise you can run:

```
$  cp ../../inputs/input_Kpoints.in .
$  $PATHQE/EPW/ZG/src/local/kpoints_band_str_unfold.x \
   < input_Kpoints.in > Kpoints.dat
$  sed -i '1,9d' Kpoints.dat # deletes first 9 info lines from Kpoints.dat
```

▶ Run a band structure unfolding calculation.

```
$ cp ../../inputs/Kpoints.dat . # if you did not complete the previous action
$ cp ../../inputs/merge_files.sh .; ./merge_files.sh
```

Note: The script `merge_files.sh` will generate 35 input files `si.333_ZG_bands_X.in` to be used for a bands calculation for every single **K**-point. This strategy is the most efficient way to perform BSU in large supercells.

Pass the following lines into a script and run:

```
# This script runs a bands calculation with "pw.x" and then performs
# band structure unfolding with "bands_unfold.x".
prefix=ZG-0.00K-si
i=1  # initial kpt index to calculate
f=35 # final kpt index to calculate
```

```
#
while [ $i -le $f ];do
    mkdir kpt_$i
    cd kpt_$i
#
    EXE=$QE/pw.x
    JNAME=ZG-bands_333_0.00K
#
    cp -r ../"$prefix".save .
    mv ../"$JNAME"_"$i".in .
    ibrun -n 14 $EXE  < "$JNAME"_"$i".in > "$JNAME"_"$i".out
#
    EXE=$QE/bands_unfold.x
    JNAME=bands
#
    cp ../"$JNAME".in .
    sed -i 's/tmp/'$i'/g' $JNAME.in
    ibrun -n 14 $EXE  < "$JNAME".in > "$JNAME".out
#
    rm -r *wfc* "$prefix".save
    cd ../
    i=$((i+1))
done
```

Note: You can also copy the script from `inputs`, i.e. `cp ../../inputs/script_bands.sh .`
This script will generate the output directories **kpt_X**, containing the band energies files (**bands01.dat**) and spectral weights (**spectral_weights01.dat**) for each single **K**-point. The full calculation should take around 5 mins. Meanwhile you can check the progress of the calculation with `ls -lrt` and see how many **kpt_X** files have been generated so far. If **kpt_1** exist, then check the outputs of **bands01.dat** and **spectral_weights01.dat**. The should contain the energies and spectral weights for **K**-point 1 (i.e. Γ here).

▶ Once energies and spectral weights for all **K**-points are calculated type:

```
$ cp ../../inputs/merge_kpts.sh .; ./merge_kpts.sh; cd all_kpts; ls
```

This will generate `all_kpts` directory and the new files **bands01.dat** and **spectral_weights01.dat**, containing the band energies, $\varepsilon_{m\mathbf{K}}(T)$, and spectral weights, $P_{m\mathbf{K},\mathbf{k}}(T)$, for all **K**-points, respectively. Now you have all the ingredients to evaluate the spectral function as in Exercise 2.

▶ Now repeat all steps after Equation 1 in Exercise 2. (Note that you are one directory down, so modify accordingly; you might also need to modify `steps` in `pp_spctrlfn.in` to 4725)

APPENDIX / Exercise 4

In this section of the appendix we show how the calculation of temperature-dependent optical spectra can be broken down to one **K**-point at a time. This strategy is useful when the supercell size is large (larger than $4 \times 4 \times 4$) and the calculation requires a large amount of memory and storage space for the wavefunctions.

▶ First go to the directory `exercise4`, create your working directories and copy the input files:

```
$ cd ../../exercise4/; mkdir workdir; cd workdir; mkdir equil; mkdir ZG
$  cd equil
$ cp ../../inputs/equil/K_list.in .
$ cp ../../inputs/equil/epsilon.in .
$ cp ../../inputs/equil/epsi_av.sh .
$ cp -r ../../../exercise1/workdir/equil-si.save/ .
$ cp ../../../exercise2/workdir/equil-nscf_333.in equil-nscf_333
```

The file `K_list.in` contains the crystal coordinates of 200 randomly generated **K**-points with all weights set to `1.0` (we assign this weight, since we will perform calculations for each **K**-point separately, in a similar spirit with the BSU calculation). We use random **K**-points, instead of a uniform grid, to speed up convergence. In `si.333_equil_nscf` apply the following changes:

1. Add `nosym = .true.` below `nbnd = 135`.
2. Delete the information for the **K**-points:

   ```
   K_POINTS crystal
   2
   0.000000 0.000000 0.000000 1
   0.000000 1.260000 1.260000 1
   ```

   and at the **bottom** of the file add the following:

   ```
   K_POINTS crystal
   1
   ```

   Note: although we use the equilibrium structure we set `nosym = .true.` since random **K**-points are employed.

The file `equil-nscf_333` should look like this:

```
&control                                                    equil-nscf_333
  calculation = 'nscf'
  restart_mode = 'from_scratch'
  prefix = 'equil-si'
  pseudo_dir = './'
  outdir = './'
/
&system
  ibrav = 0
  nat =   54
  ntyp =    1
  ecutwfc =  20.00
  nbnd = 135
  nosym = .true.
  occupations = 'fixed'
  smearing = 'gaussian'
  degauss =   0.0D+00
/
&electrons
  diagonalization = 'david'
  mixing_mode= 'plain'
```

```
  mixing_beta = 0.70
  conv_thr =   0.1D-06
/
ATOMIC_SPECIES
   Si   28.085 Si.pz-vbc.UPF
CELL_PARAMETERS {angstrom}
     -8.09641133      0.00000000      8.09641133
      0.00000000      8.09641133      8.09641133
     -8.09641133      8.09641133      0.00000000
ATOMIC_POSITIONS {angstrom}
   Si     0.00000000   0.00000000   0.00000000
   Si    -1.34940189   1.34940189   1.34940189
   Si    -2.69880378   0.00000000   2.69880378
...
K_POINTS crystal
1
```

▶ Run an optical spectrum calculation for silicon using the $3 \times 3 \times 3$ equilibrium supercell and 30 random **K**-points. In the same spirit with the BSU calculation we calculate $\epsilon_2(\omega)$ for every **K**-point using the following script:

```
prefix=equil-si
i=1  # initial kpt index to calculate
f=30 # final kpt index to calculate

while [ $i -le $f ];do
#
    JNAME=equil-nscf_333
    awk 'NR == '$i'' K_list.in > K_point.txt
    cat $JNAME K_point.txt > "$JNAME"_"$i".in
#
    mkdir kpt_$i
    cd kpt_$i
#
    EXE=$QE/pw.x
#
    cp -r ../"$prefix".save .
    mv ../"$JNAME"_"$i".in .
    ibrun -n 14 $EXE  < "$JNAME"_"$i".in > "$JNAME"_"$i".out
#
    EXE=$QE/epsilon_Gaus.x
    JNAME=epsilon
#
    cp ../"$JNAME".in .
    sed -i 's/tmp/'$i'/g' $JNAME.in
    ibrun -n 14 $EXE  < "$JNAME".in > "$JNAME".out
#
    rm -r *wfc* "$prefix".save
    cd ../
    i=$((i+1))
done
```

Note: You can also copy the script from `inputs`, i.e. `cp ../../inputs/equil/script_equil.sh .`

This script will generate the output directories **kpt_X**, containing $\epsilon_2(\omega)$ in the file **epsi_equil-si.dat** for each single **K**-point. The full calculation should take around 3 mins. Meanwhile you can check the progress of the calculation with `ls -lrt` and see how many **kpt_X** directories have been generated. If **kpt_1** exist, then check the format of **kpt_1/epsi_si_333_equil.dat**. The first column is the energy grid, and the rest three represent $\epsilon_2(\omega)$ along the three Cartesian directions.

▶ Once the calculation is completed run the script:

```
$ ./epsi_av.sh
```

This script takes first the isotropic average of $\epsilon_2(\omega)$ calculated for each **K**-point and then takes the average of $\epsilon_2(\omega)$ over all **K**-points. The output file containing this data is **epsi_si_333_equil_30.dat**. **Note:** `epsi_av.sh` takes the average over 30 **K**-points. You can modify this number using the variable `m` in the script. Now copy the output file in the `gnuplot` directory using:

```
$ cp epsi_si_333_equil_30.dat ../../gnuplot/.
```

▶ Run an optical spectra calculation for silicon using the $3 \times 3 \times 3$ ZG supercell following exactly the same procedure used for the equilibrium structure (as above); make sure now you use the file `ZG-nscf_333_0.00K.in`.

APPENDIX / Exercise C

```
# script.sh for computing phonons and band gap renormalization in diamond
cd $PWD

QE=path_to_qe_bin

wget http://www.pseudo-dojo.org/pseudos/nc-sr-04_pbesol_standard/C.upf.gz
gunzip C.upf.gz
 cp ../inputs/* .
# calculate the phonons
 ibrun -np 4 $QE/pw.x -nk 4 < scf.in > scf.out
 cp ../inputs/ph.in .
 ibrun -np 4 $QE/ph.x -nk 4 < ph.in > ph.out
 cp ../inputs/q2r.in .
 ibrun -np 1 $QE/q2r.x < q2r.in > q2r.out
 ibrun -np 1 $QE/matdyn.x < matdyn.in > matdyn.out
 rm -rf _ph0
#
# band strcture for unit cell
 ibrun -np 4 $QE/pw.x -nk 4 < bands.in > bands.out
#
# generate ZG configurations for multiple temperatures
 ibrun -np 4 $QE/ZG.x -nk 4 < ZG_444_multi_T.in > ZG_444_multi_T.out
#
# run scf calculations for equilibrium and ZG structures
 ibrun -np 48 $QE/pw.x -nk 3 < equil-scf_444.in > equil-scf_444.out
 declare -i j
 array=(0 50 100 150 200 250 300 350 400 600 800 1000)
 for j in ${array[@]}
 do
   ibrun -np 48 $QE/pw.x -nk 8 < ZG-scf_444_"$j".00K.in > ZG-scf_444_"$j".00K.out
 done

# prepare nscf files
 ./com.sh # script provided below

# run nscf calculations for equilibrium and ZG structures
 ibrun -np 48 $QE/pw.x < equil-nscf_444.in > equil-nscf_444.out
#
 declare -i j
 array=(0 50 100 150 200 250 300 350 400 600 800 1000)
 for j in ${array[@]}
 do
   ibrun -np 48 $QE/pw.x < ZG-nscf_444_"$j".00K.in > ZG-nscf_444_"$j".00K.out
   rm *wfc* *.save/*wfc*
 done

# calculate the band gap
./com_av_gap.sh  # script provided below
```

```
# com.sh to prepare nscf files
#
#!/bin/bash
#
cp equil-scf_444.in equil-nscf_444.in
sed -i 's/scf/nscf/g' equil-nscf_444.in
sed -i 's/-nscf/-scf/g' equil-nscf_444.in
sed -i 's/ecutwfc/nosym = .true., nbnd = 320, ecutwfc/g' equil-nscf_444.in
sed -i '/K_P/d' equil-nscf_444.in
sed -i '/2   2   2/d' equil-nscf_444.in
```

```
#
cat equil-nscf_444.in Kpoints.txt > tmp
mv tmp equil-nscf_444.in
#
declare -i i
array=(0 50 100 150 200 250 300 400 600 800 1000)
for i in ${array[@]}
do
  cp ZG-scf_444_"$i".00K.in ZG-nscf_444_"$i".00K.in
done
#
sed -i 's/scf/nscf/g' ZG-nscf_444_*
sed -i 's/-nscf/-scf/g' ZG-nscf_444_*
sed -i 's/ecutwfc/nosym = .true., nbnd = 320, ecutwfc/g' ZG-nscf_444_*
sed -i '/K_P/d' ZG-nscf_444_*
sed -i '/2   2   2/d' ZG-nscf_444_*
#
# Add k-point for the direct gap of diamond
for i in ${array[@]}
do
  cat ZG-nscf_444_"$i".00K.in Kpoints.txt > tmp
  mv tmp ZG-nscf_444_"$i".00K.in
done
```

```
# com_av_gap.sh to extract the band gap from the ZG nscf outputs
#
#!/bin/bash
#
rm tmp tmp2 tmp3
# find the gap at equilibrium
  grep highest equil-nscf_444.out > tmp
  a=$(awk '{print $8}' tmp) # cbm
  awk '/'$a'/ { getline; print }' equil-nscf_444.out > tmp # print line below
  sed -i '$d' tmp # remove last empty line
  awk '{print ($5+$6+$7)/3}' tmp >> tmp3 # three deg. bands
  grep highest equil-nscf_444.out > tmp
  a=$(awk '{print $7}' tmp) # vbm
  grep $a equil-nscf_444.out |head -1 > tmp
  awk '{print ($6+$7+$8)/3}' tmp >> tmp4 # three deg. bands
  paste tmp3 tmp4 > tmp5
  awk '{print ($1-$2)}' tmp5 > eq_direct_gap.dat
#
rm tmp*
rm direct_gap_T_av.dat
touch direct_gap_T_av.dat
declare -i i
array=(0 50 100 150 200 250 300 400 600 800 1000)
for i in ${array[@]}
do
  grep highest ZG-nscf_444_"$i".00K.out > tmp
  a=$(awk '{print $8}' tmp) # cbm
  awk '/'$a'/ { getline; print }' ZG-nscf_444_"$i".00K.out > tmp # print line below
  sed -i '$d' tmp # remove last empty line
  awk '{print ($5+$6+$7)/3}' tmp >> tmp3
  grep highest ZG-nscf_444_"$i".00K.out > tmp
  a=$(awk '{print $7}' tmp) # vbm
  grep $a ZG-nscf_444_"$i".00K.out |head -1 > tmp
  awk '{print ($6+$7+$8)/3}' tmp >> tmp4
  paste tmp3 tmp4 > tmp5
  awk '{print ($1-$2)}' tmp5 > tmp6
  echo $i >> tmp2
```

```
done
paste tmp2 tmp6 > direct_gap_T_av.dat
rm tmp*
```

APPENDIX / Exercise PbTe

```
# script.sh for computing anharmonic phonons in PbTe
cd $PWD

QE=path_to_qe_bin

# download pseudos from pseudo-dojo library
wget http://www.pseudo-dojo.org/pseudos/nc-sr-04_pbesol_standard/Pb.upf.gz
wget http://www.pseudo-dojo.org/pseudos/nc-sr-04_pbesol_standard/Te.upf.gz
gunzip Pb.upf.gz
gunzip Te.upf.gz

# copy inputs
cp -r ../inputs/* .
# run sch and ph to obtain harmonic phonons
ibrun -np 48 $QE/pw.x -nk 12 < scf.in > scf.out
ibrun -np 48 $QE/ph.x -nk 12 < ph.in > ph.out
ibrun -np 1 $QE/q2r.x < q2r.in > q2r.out
ibrun -np 1 $QE/matdyn.x < matdyn.in > matdyn.out
rm -rf _ph0 *wfc* *save/*wfc*
#
# set temperature in ZG files by replacing string AAA
T=300
sed -i 's/AAA/'$T'/g' ZG*in
sed -i 's/AAA/'$T'/g' matdyn_*in

# first iteration starting from harmonic phonons
ibrun -np 1 $QE/ZG.x < ZG_1.in > ZG_1.out

file=ZG-scf_300.00K_iter_01
mkdir fd_forces/
for i in {0001..0096}; do
  mkdir $i
  cd $i
  cp ../Pb.upf .
  cp ../Te.upf .
  mv ../"$file"_"$i".in .
## We have copied the "fd_forces" folder from inputs which includes all ASDM calculations.
## This is done for spead up purposes. Otherwise, calculate and copy forces files by
## uncommenting the two lines below.
#  ibrun -np 48 $QE/pw.x -nk 12 < "$file"_"$i".in > "$file"_"$i".out
#  mv "$file"_"$i".out ../fd_forces/.
  rm Pb.upf Te.upf
  cd ../
done

ibrun -np 1 $QE/ZG.x < ZG_2.in > ZG_2.out
ibrun -np 1 $QE/matdyn.x < matdyn_01.in > matdyn_01.out
#
# second iteration
ibrun -np 1 $QE/ZG.x < ZG_3.in > ZG_3.out

file=ZG-scf_300.00K_iter_02
# mkdir fd_forces/
for i in {0001..0096}; do
 # mkdir $i
  cd $i
  cp ../Pb.upf .
  cp ../Te.upf .
  mv ../"$file"_"$i".in .
```

```
## We have copied the "fd_forces" folder from inputs which includes all ASDM calculations.
## This is done for spead up purposes. Otherwise, calculate and copy forces files by
## uncommenting the two lines below.
#  ibrun -np 48 $QE/pw.x -nk 12 < "$file"_"$i".in > "$file"_"$i".out
#  mv "$file"_"$i".out ../fd_forces/.
  rm Pb.upf Te.upf
  cd ../
done

ibrun -np 1 $QE/ZG.x < ZG_4.in > ZG_4.out
ibrun -np 1 $QE/matdyn.x < matdyn_02.in > matdyn_02.out

# third iteration
ibrun -np 1 $QE/ZG.x < ZG_5.in > ZG_5.out

file=ZG-scf_300.00K_iter_03
# mkdir fd_forces/
for i in {0001..0096}; do
  # mkdir $i
  cd $i
  cp ../Pb.upf .
  cp ../Te.upf .
  mv ../"$file"_"$i".in .
## We have copied the "fd_forces" folder from inputs which includes all ASDM calculations.
## This is done for spead up purposes. Otherwise, calculate and copy forces files by
## uncommenting the two lines below.
#  ibrun -np 48 $QE/pw.x -nk 12 < "$file"_"$i".in > "$file"_"$i".out
#  mv "$file"_"$i".out ../fd_forces/.
  rm Pb.upf Te.upf
  cd ../
done

ibrun -np 1 $QE/ZG.x < ZG_6.in > ZG_6.out
ibrun -np 1 $QE/matdyn.x < matdyn_03.in > matdyn_03.out
```

```
# script_ASDM.sh for computing anharmonic phonons in PbTe
cd $PWD

QE=path_to_qe_bin

# Repeat the process for different temperature
# set temperature
T=500
cp ../inputs/ZG*in .
cp ../inputs/matdyn*in .
sed -i 's/AAA/'$T'/g' ZG*in
sed -i 's/AAA/'$T'/g' matdyn_*in

# first iteration only
ibrun -np 1 $QE/ZG.x < ZG_1.in > ZG_1.out

file=ZG-scf_"$T".00K_iter_01
# mkdir fd_forces/
for i in {0001..0096}; do
  # mkdir $i
  cd $i
  cp ../Pb.upf .
  cp ../Te.upf .
  mv ../"$file"_"$i".in .
## We have copied the "fd_forces" folder from inputs which includes all ASDM calculations.
## This is done for spead up purposes. Otherwise, calculate and copy forces files by
```

```
## uncommenting the two lines below.
#  ibrun -np 48 $QE/pw.x -nk 12 < "$file"_"$i".in > "$file"_"$i".out
#  mv "$file"_"$i".out ../fd_forces/.
  rm Pb.upf Te.upf
  cd ../
done

ibrun -np 1 $QE/ZG.x < ZG_2.in > ZG_2.out
ibrun -np 1 $QE/matdyn.x < matdyn_01.in > matdyn_01.out
```

```
# script_harm.sh for computing the band gap at 300 K, 500 K and 700 K using harmonic phonons
cd $PWD
QE=path_to_qe_bin

# copy inputs
cp ../inputs/ZG_harm.in .
cp ../inputs/Kpoints.txt .
ibrun -np 1 $QE/ZG.x < ZG_harm.in > ZG_harm.out

# run scf for equilibrium structure (no el-ph coupling)
ibrun -np 48 $QE/pw.x -nk 3 < equil-scf_222.in > equil-scf_222.out

# prepare file to run nscf by adding empty bands
cp equil-scf_222.in equil-nscf_222.in
sed -i 's/scf/nscf/g' equil-nscf_222.in
sed -i 's/-nscf/-scf/g' equil-nscf_222.in
sed -i 's/ecutwfc/nosym = .true., nbnd = 140, ecutwfc/g' equil-nscf_222.in
sed -i '/K_P/d' equil-nscf_222.in
sed -i '/3   3   3/d' equil-nscf_222.in
# K point as specified in Kpoints.txt. We use Gamma point which captures the direct gap
# of PbTe in a 2x2x2 supercell. Note that for, e.g., a 3x3x3 supercell this is not the
# case and one needs to specify a different K point.
cat equil-nscf_222.in Kpoints.txt > tmp
mv tmp equil-nscf_222.in
# run nscf
ibrun -np 48 $QE/pw.x < equil-nscf_222.in > equil-nscf_222.out
rm *wfc* *.save/*wfc*
grep highest equil-nscf_222.out > tmp
awk '{print $8-$7}' tmp > equil_gap.dat

# repeat the step above but now for T-dependent ZG files
# Those are generated relying on the harmonic phonons --> see ZG_harm.in
declare -i j
array=(300 500 700)
for j in ${array[@]}
do
  file=ZG-nscf_222_"$j".00K
  # run scf
  ibrun -np 48 $QE/pw.x -nk 8 < ZG-scf_222_"$j".00K.in > ZG-scf_222_"$j".00K.out
  cp ZG-scf_222_"$j".00K.in "$file".in
  # prepare nscf files
  sed -i 's/scf/nscf/g' "$file".in
  sed -i 's/-nscf/-scf/g' "$file".in
  sed -i 's/ecutwfc/nosym = .true., nbnd = 140, ecutwfc/g' "$file".in
  sed -i '/K_P/d' "$file".in
  sed -i '/3   3   3/d' "$file".in
  cat "$file".in Kpoints.txt > tmp
  mv tmp "$file".in
  # run nscf
  ibrun -np 48 $QE/pw.x < "$file".in > "$file".out
  rm *wfc* *.save/*wfc*
```

```
done
```

```
# gap_222.sh for finding the direct band gap at 300 K, 500 K and 700 K of PbTe
rm gap_222_T.dat
touch gap_222_T.dat
declare -i j
array=(300 500 700)
for j in ${array[@]}
do
  a=$(grep highest ZG-nscf_222_"$j".00K.out | awk '{print $7}')
  grep $a ZG-nscf_222_"$j".00K.out | head -1 > tmp
  awk '{print ($5+$6+$7+$8)/4}' tmp > VBM # average of 4 bands
  a=$(grep highest ZG-nscf_222_"$j".00K.out | awk '{print $8}')
  grep $a ZG-nscf_222_"$j".00K.out | head -1 > tmp
  awk '{print ($1+$2+$3+$4)/4}' tmp > CBM # average of 4 bands
  echo $j > T
  paste T VBM CBM > tmp2
  awk '{print $1, ($3-$2)}' tmp2 >> gap_222_T.dat
  rm T VBM CBM tmp*
done
```

```
cd $PWD

QE=/home1/07369/mzach/codes/q-e_dev_2024/bin


# set array with temperatures
array=(300 500 700)
for j in ${array[@]}
do
  # run ZG for each temperature using the corresponding T.00K_iter_01.fc file
  ibrun -n 1 $QE/ZG.x < ZG_"$j"K.in > ZG_"$j"K.out
  # run scf
  ibrun -np 48 $QE/pw.x -nk 8 < ZG-scf_222_"$j".00K.in > ZG-scf_222_"$j".00K.out
  # prepare nscf file
  file=ZG-nscf_222_"$j".00K
  cp ZG-scf_222_"$j".00K.in "$file".in
  sed -i 's/scf/nscf/g' "$file".in
  sed -i 's/-nscf/-scf/g' "$file".in
  sed -i 's/ecutwfc/nosym = .true., nbnd = 140, ecutwfc/g' "$file".in
  sed -i '/K_P/d' "$file".in
  sed -i '/3   3   3/d' "$file".in
  cat "$file".in Kpoints.txt > tmp
  mv tmp "$file".in
  # run nscf
  ibrun -np 48 $QE/pw.x < "$file".in > "$file".out
  rm *wfc* *.save/*wfc*
done
```

APPENDIX / Exercise $CsPbBr_3$

```
# script.sh for computing anharmonic phonons of CsPbBr3 at 450 K

cd $PWD
QE=/work2/05193/sabyadk/stampede3/EPWSchool2024/q-e/bin

# download pseudos from pseudo-dojo library
wget http://www.pseudo-dojo.org/pseudos/nc-sr-04_pbesol_standard/Cs.upf.gz
wget http://www.pseudo-dojo.org/pseudos/nc-sr-04_pbesol_standard/Pb.upf.gz
wget http://www.pseudo-dojo.org/pseudos/nc-sr-04_pbesol_standard/Br.upf.gz
gunzip Cs.upf.gz Pb.upf.gz Br.upf.gz

# calculate harmonic phonons
cp ../inputs/scf.in .
ibrun -np 48 $QE/pw.x -nk 8 < scf.in > scf.out
cp ../inputs/ph.in .
ibrun -np 48 $QE/ph.x -nk 8 < ph.in > ph.out
rm -r _ph0 *wfc* *save/*wfc*
cp ../inputs/q2r.in .
ibrun -np 1 $QE/q2r.x < q2r.in > q2r.out
cp ../inputs/matdyn.in .
ibrun -np 1 $QE/matdyn.x < matdyn.in > matdyn.out

# compute the polymorphous structure
rm ZG-relax.out
cp ../inputs/ZG_*in .
# generate ZG-relax.in
ibrun -np 1 $QE/ZG.x < ZG_1.in > ZG_1.out
cp ../inputs/ZG-relax.out .
## to speed up the process we copy ZG-relax.out from inputs.
## Uncomment the line below if you want to perform the relaxation.
#ibrun -np 40 $QE/pw.x -nk 10 < ZG-relax.in > ZG-relax.out
ibrun -np 1 $QE/ZG.x < ZG_2.in > ZG_2.out

# compute the phonons of the polymorphous structure
## to speed up the process we copy scf output files from inputs.
## Comment the line below if you want to perform the calculations.
cp -r ../inputs/fd_forces .
#
file=ZG-scf_poly_iter_00
mkdir fd_forces/
for i in {0001..0240}; do
  mkdir $i
  cd $i
  cp ../*.upf .
  mv ../"$file"_"$i".in .
## to speed up the process, outputs ZG-scf_poly_iter_00*out are given in fd_forces
## Uncomment the two lines below to perform the scf calculations.
#  ibrun -np 48 $QE/pw.x -nk 8 < "$file"_"$i".in > "$file"_"$i".out
#  mv "$file"_"$i".out ../fd_forces/.
  rm *.upf
  cd ../
done
ibrun -np 1 $QE/ZG.x < ZG_3.in > ZG_3.out
cp ../inputs/matdyn_poly.in .
ibrun -np 1 $QE/matdyn.x < matdyn_poly.in > matdyn_poly.out

# compute the phonons at 450 K: first iteration
ibrun -np 1 $QE/ZG.x < ZG_4.in > ZG_4.out
#
```

```
file=ZG-scf_450.00K_iter_01
for i in {0001..0240}; do
  mkdir $i
  cd $i
  cp ../*.upf .
  mv ../"$file"_"$i".in .
## to speed up the process, outputs ZG-scf_450.00K_iter_01*out are given in fd_forces
## Uncomment the two lines below to perform the scf calculations.
#  ibrun -np 48 $QE/pw.x -nk 8 < "$file"_"$i".in > "$file"_"$i".out
#  mv "$file"_"$i".out ../fd_forces/.
  rm *.upf
  cd ../
done
ibrun -np 1 $QE/ZG.x < ZG_5.in > ZG_5.out
cp ../inputs/matdyn_01.in .
ibrun -np 1 $QE/matdyn.x < matdyn_01.in > matdyn_01.out

# compute the phonons at 450 K: second iteration
ibrun -np 1 $QE/ZG.x < ZG_6.in > ZG_6.out
#
file=ZG-scf_450.00K_iter_02
for i in {0001..0240}; do
  mkdir $i
  cd $i
  cp ../*.upf .
  mv ../"$file"_"$i".in .
## to speed up the process, outputs ZG-scf_450.00K_iter_02*out are given in fd_forces
## Uncomment the two lines below to perform the scf calculations.
#  ibrun -np 48 $QE/pw.x -nk 8 < "$file"_"$i".in > "$file"_"$i".out
#  mv "$file"_"$i".out ../fd_forces/.
  rm *.upf
  cd ../
done
ibrun -np 1 $QE/ZG.x < ZG_7.in > ZG_7.out
cp ../inputs/matdyn_02.in .
ibrun -np 1 $QE/matdyn.x < matdyn_02.in > matdyn_02.out
```

```
# script_mono_gap.sh for computing band gap renormalization of monomorphous CsPbBr3

cd $PWD
QE=/home1/07369/mzach/codes/q-e_dev_2024

# generate ZG configurations of the monomorphous structure for multiple temperatures
# using anharmonic phonons at 450 K
ibrun -np 4 $QE/ZG.x -nk 4 < ZG_multi_T.in > ZG_multi_T.out

# static-equilibrium and ZG configurations of the monomorphous structure
# scf calculations
ibrun -np 48 $QE/pw.x -nk 8 < equil-scf_222.in > equil-scf_222.out
rm *save/*wfc*
array=(430 450 470 500)
for j in ${array[@]}
do
  ibrun -np 48 $QE/pw.x -nk 8 < ZG-scf_222_"$j".00K.in > ZG-scf_222_"$j".00K.out
  rm ZG-"$j".00K-scf.save/*wfc*
  rm ZG-"$j".00K-scf.save/*upf
done
#
# copy inputs and prepare ncsf files
cp ../inputs/Kpoints.txt .
cp ../inputs/mono.sh .
```

```
./mono.sh # see the script below

# nscf calculations
ibrun -np 48 $QE/pw.x < equil-nscf_222.in > equil-nscf_222.out
rm *save/*wfc*
mono_gap=$(grep highest equil-nscf_222.out | awk '{print $8-$7}')
echo $mono_gap > mono_gap.dat
#
array=(430 450 470 500)
for j in ${array[@]}
do
  ibrun -np 48 $QE/pw.x  < ZG-nscf_222_"$j".00K.in > ZG-nscf_222_"$j".00K.out
  rm ZG-"$j".00K-scf.save/*wfc*
  rm ZG-"$j".00K-scf.save/*upf
  rm *wfc*
done
# calculate the band gap renormalization from nscf files
grep highest ZG-nscf_222_*.out > tmp
awk '{print $1, $8, $9, $9-$8, $9-$8-'$mono_gap'}' tmp | sed 's/ZG-nscf_222_//g' \
| sed 's/K.out://g' > Renorm_mono_T.dat
```

```
#!/bash
#
# mono.sh to generate nscf files for the monomorphous structure
#
#!/bash
#
declare -i i
cp equil-scf_222.in equil-nscf_222.in
#
array=(430 450 470 500)
for i in ${array[@]}
do
  cp ZG-scf_222_"$i".00K.in ZG-nscf_222_"$i".00K.in
done

sed -i 's/scf/nscf/g' *-nscf*
sed -i 's/-nscf/-scf/g' *-nscf*
sed -i 's/ecutwfc/nosym = .true., nbnd = 200, ecutwfc/g' *-nscf*
sed -i '/K_P/d' *-nscf*
sed -i '/2   2   2/d' *-nscf*

cat equil-nscf_222.in Kpoints.txt > tmp
mv tmp equil-nscf_222.in

for i in ${array[@]}
do
  cat ZG-nscf_222_"$i".00K.in Kpoints.txt > tmp
  mv tmp ZG-nscf_222_"$i".00K.in
done
```

```
# script_poly_gap.sh for computing band gap renormalization of polymorphous CsPbBr3

cd $PWD
QE=/home1/07369/mzach/codes/q-e_dev_2024

# static-equilibrium and ZG configurations of the polymorphous structure
# scf calculation for static-equilibrium polymorphous structure
ibrun -np 48 $QE/pw.x -nk 8 < ZG-scf_poly_iter_00.in > ZG-scf_poly_iter_00.out
ibrun -np 48 $QE/pw.x < ZG-nscf_poly_iter_00.in > ZG-nscf_poly_iter_00.out
rm *wfc* *.save/*wfc*
```

```
poly_gap=$(grep highest ZG-nscf_poly_iter_00.out | awk '{print $8-$7}')
echo $poly_gap > poly_gap.dat
#
# apply ZG displacements and generate, scf, and nscf files
# for calculations on the polymorphous structure
cp ../inputs/poly.sh .
./poly.sh

# run scf and nscf calculations
array=(430 450 470 500)
for j in ${array[@]}
do
  ibrun -np 48 $QE/pw.x -nk 8  < ZG-scf_poly_"$j".00K.in > ZG-scf_poly_"$j".00K.out
  ibrun -np 48 $QE/pw.x < ZG-nscf_poly_"$j".00K.in > ZG-nscf_poly_"$j".00K.out
  rm ZG-"$j".00-poly-scf.save/*wfc*
  rm ZG-"$j".00-poly-scf.save/*upf
  rm *wfc*
done
# calculate the band gap renormalization
grep highest ZG-nscf_poly_*K.out > tmp
awk '{print $1, $8, $9, $9-$8, $9-$8-'$poly_gap'}' tmp | sed 's/ZG-nscf_poly_//g' \
| sed 's/K.out://g' > Renorm_poly_T.dat
```

```
#
# poly.sh to get positions, scf, and nscf files for the polymorphous structure
#
#!/bash
#
declare -i i
#
sed -i '/disk_io/d' ZG-scf_poly_iter_00.in
cp ZG-scf_poly_iter_00.in ZG-nscf_poly_iter_00.in
#
cat ZG-nscf_poly_iter_00.in Kpoints.txt > tmp
mv tmp ZG-nscf_poly_iter_00.in
#
head -n -40 ZG-scf_poly_iter_00.in > ZG-scf_poly_iter_00
tail -40 ZG-scf_poly_iter_00.in > poly_pos.dat
#
cp equil_pos.dat tmp
sed -i '1,2d' tmp
array=(430 450 470 500)
for i in ${array[@]}
do
  # capture displacements by taking the difference between
  # ZG coordinates and equil structure
  cp ZG-configuration_"$i".00K.dat tmpZG
  sed -i '1,2d' tmpZG
  paste tmpZG tmp > oo
  awk '{print $1, $2 - $6, $3 - $7, $4 - $8}' oo > displ_"$i".txt
  # add displacements on the polymorphous configuration
  paste poly_pos.dat displ_"$i".txt > oo
  awk '{print $1, $2 + $6, $3 + $7, $4 + $8}' oo > ZG_poly-configuration_"$i".00K.dat
  cat ZG-scf_poly_iter_00 ZG_poly-configuration_"$i".00K.dat > ZG-scf_poly_"$i".00K.in
  sed -i 's/ZG-scf/ZG-'$i'.00-poly-scf/' ZG-scf_poly_"$i".00K.in
  cp ZG-scf_poly_"$i".00K.in ZG-nscf_poly_"$i".00K.in
done

# prepare nscf files with empty bands and remove uniform grid info
sed -i 's/scf/nscf/g' *-nscf_poly*
```

```
sed -i 's/-nscf/-scf/g' *-nscf_poly*
sed -i 's/ecutwfc/nosym = .true., nbnd = 200, ecutwfc/g' *-nscf_poly*
sed -i '/K_P/d' *-nscf_poly*
sed -i '/2   2   2/d' *-nscf_poly*

# add correct K-point at the end of the nscf files
for i in ${array[@]}
do
  cat ZG-nscf_poly_"$i".00K.in Kpoints.txt > tmp
  mv tmp ZG-nscf_poly_"$i".00K.in
done
```