2021 Virtual School on Electron-Phonon Physics and the EPW code

Temperature-dependent band structures and optical spectra

Hands-on Session (Friday, 18th June)

In this session we will learn how to use `ZG.x` for generating ZG supercell configurations and how to combine them with standard Density Functional Theory (DFT) calculations for evaluating temperature-dependent properties. You are advised to run the calculations in your scratch directory (`cd $SCRATCH`) and to prepare the following script file, e.g. `script.sh`:

```
#!/bin/bash
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=56
#SBATCH -A EPW-SCHOOL
#SBATCH --partition=small
#SBATCH --reservation=EPW-SCHOOL-06-18-2021
module purge
module load TACC
PATHQE=/work2/06868/giustino/EPW-SCHOOL/q-e/ # this is the path to q-e
cd $PWD
```

For each calculation, pass the `ibrun` 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:

`PATHQE=/work2/06868/giustino/EPW-SCHOOL/q-e/`

Some routines in PW, PH, and PP directories are required to run these exercises (e.g. `make pw ph pp`). To run ZG calculations go to `$PATHQE/EPW/ZG_displacement/src` and type `make`. To use local executables go to `$PATHQE/EPW/ZG_displacement/src/local` and type `./compile_gfortran.sh`.

Please copy the tutorial tarball, go to `exercise1`, and create directory `workdir`:

```
$ cp /work2/06868/giustino/EPW-SCHOOL/Fri.6.Zacharias.tar $SCRATCH
$ tar -xvf Fri.6.Zacharias.tar; cd tuto_Fri6/exercise1; mkdir workdir
```

Note: in this tutorial we will show how to obtain all input and output files. The directories `inputs` and `outputs` will be 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 of silicon are provided, but to speed up the process we will skip the first *four* steps.

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 negative frequencies (soft modes), the code sets them to positive.

▶ 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 $PATHQE/bin/pw.x -nk 4 < si.scf.in > si.scf.out
```

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

```
--                                                                      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      = 4, nq2 = 4, nq3 = 4
/
```

```
$ cp ../inputs/si.ph.in .
$ ibrun -np 4 $PATHQE/bin/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 **q**-point. The list of irreducible **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 $PATHQE/bin/q2r.x < q2r.in > q2r.out
```

This will generate **si.444.fc** which contains the IFCs. As we will see below, this is the only external input file necessary for running `ZG.x`.

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

```
--                                                                     matdyn.in
&input
   asr='simple', 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 .
$ ibrun -np 1 $PATHQE/bin/matdyn.x < matdyn.in > matdyn.out
$ $PATHQE/bin/plotband.x
```

```
Input file >  si.freq
Reading    6 bands at    702 k-points
Range:     0.0000  509.7631eV  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.eps
```
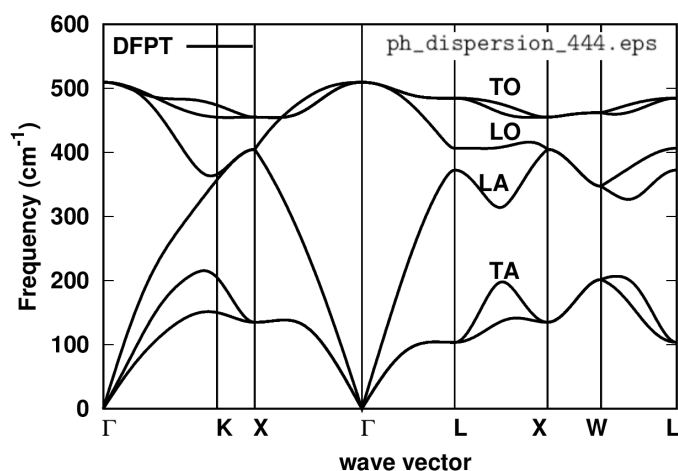

Is the phonon dispersion the one you would expect? One can verify the results with literature data.

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. 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='simple', amass(1)=28.0855, atm_zg(1) = 'Si',
    T = 0.00,
    dim1 = 3, dim2 = 3, dim3 = 3
    compute_error = .true., synch = .true., error_thresh = 0.02, niters = 30000
    incl_qA = .false.
/
```

Note: The 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**.

```
$ cp ../inputs/ZG_333.in .
$ ibrun -np 4 $PATHQE/bin/ZG.x -nk 4 < ZG_333.in > ZG_333.out
```

The input format of `ZG_333.in` is the same as matdyn.in. Lines 3 and 4 contain input variables similar to those in matdyn.in; In lines 5 and 6 we provide the temperature (T) in Kelvin and supercell dimensions (`dimX`) for which ZG displacements are generated; line 7 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`. More details about all input flags are provided in `tuto_Fri6_flags.pdf`.

The outputs from a `ZG.x` run are: **ZG-configuration_0.020.dat** and **equil_pos.dat** which contain the ZG (quantum nuclei at T K) and equilibrium (classical nuclei at 0 K) coordinates in angstroms.

▶ Run a self-consistent calculation for silicon with the nuclei clamped at their equilibrium coordinates in the supercell. To prepare the input file type:

```
$ cat ../inputs/si.scf.in equil_pos.dat > si.333_equil_scf.in
```

Then open the file `si.333_equil_scf.in` and apply the following modifications:

1. Set `prefix = 'si_333_equil'`

2. Set `celldm(1) = 30.60` (i.e. provide the correct lattice constants for the supercell),

3. Set `nat = 54`,

4. Set the automatic **k**-grid to `2 2 2 0 0 0`,

5. Add `(angstroms)` next to `ATOMIC_POSITIONS`,

6. Delete the lines right after `ATOMIC_POSITIONS` until you reach the atomic coordinates of the first atom (here you should delete four lines).

The input file `si.333_equil_scf.in` should look like this:

```
&control                                                    si.333_equil_scf.in
    calculation = 'scf'
    restart_mode = 'from_scratch',
    prefix = 'si_333_equil',
    pseudo_dir = './',
    outdir='./'
 /
 &system
    ibrav = 2,
    celldm(1) = 30.60,
    nat =  54,
    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
2 2 2 0 0 0
ATOMIC_POSITIONS (angstroms)
   Si    0.00000000    0.00000000    0.00000000
   Si    1.34940189    1.34940189    1.34940189
   Si   -2.69880378    0.00000000    2.69880378
...
```

```
$ cp ../inputs/si.333_equil_scf.in . # (if you didn't generate the input file)
$ ibrun -np 54 $PATHQE/bin/pw.x -nk 3 < si.333_equil_scf.in > si.333_equil_scf.out
```
Note: parallelization over 3 **k**-points.

▶ Run a self-consistent ZG calculation for silicon. Prepare the input file:

```
$ cat ../inputs/si.scf.in ZG-configuration_0.020.dat > si.333_ZG_scf.in
```

and repeat the previous modifications (1-6), but now set `prefix = 'si_333_ZG'`. The input file is:

```
&control                                                    si.333_ZG_scf.in
    calculation = 'scf'
    restart_mode = 'from_scratch',
    prefix = 'si_333_ZG',
    pseudo_dir = './',
    outdir='./'
 /
 &system
    ibrav = 2,
    celldm(1) = 30.60,
    nat =  54,
    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
2 2 2 0 0 0
ATOMIC_POSITIONS (angstroms)
   Si    0.07465546  -0.02027484   0.01183559
   Si    1.33329554   1.39151777   1.34662616
   Si   -2.60922970   0.02806964   2.69588796
...
```

Note: in general you should add `nosym = .true.`, since symmetries do not apply after ZG displacements.

```
$ cp ../inputs/si.333_ZG_scf.in . # (if you didn't generate the input file)
$ ibrun -np 56 $PATHQE/bin/pw.x -nk 8 < si.333_ZG_scf.in > si.333_ZG_scf.out
```
Note: parallelization over 8 **k**-points.

Let us now discuss some apsects about the generation of the **ZG-configuration** file. Type:

```
$ head -5 ZG-configuration_0.020.dat
```

```
Sum of diagonal terms per q-point:    0.548483
Error and niter index:    0.018511        25969
    Temperature is:   0.00 K
 Atomic positions          54
   Si    0.07465546  -0.02027484   0.01183559
```

These are the first 5 lines in the **ZG-configuration** file. Line 1 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 2 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 2 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. 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 $\mathbf{q}$-points, is less important as we approach the thermodynamic limit (see lecture notes). Lines 3 and 4 give the temperature for which ZG displacements are generated and the total number of atomic coordinates, respectively.

In line 5, 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, of course, should remain the same in all cases. The synchronization routine (setting `synch = .true.`) should apply a smooth Berry connection and align 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 displacement tensor with the exact values, both obtained at the end of the **ZG-configuration.dat**. To this aim type:

```
$ tail -12 ZG-configuration_0.020.dat
```

```
 Anisotropic displacement tensor vs exact values:
Atom: 1
   Si    0.187294    0.179045    0.191964
        Exact values
   Si    0.185657    0.185657    0.185657
Atom: 2
   Si    0.184972    0.181375    0.189294
        Exact values
   Si    0.185657    0.185657    0.185657
 off-diagonal terms
   Si    0.022325    0.006336    0.000134
   Si   -0.006360   -0.044372   -0.018081
```

Reducing `error_thresh` brings the anisotropic displacement tensor closer to the exact values. Now check whether the calculations have finished and type:

```
$ grep ! si.333_*.out
```

```
si.333_equil_scf.out:!    total energy              =    -427.83892821 Ry
si.333_ZG_scf.out:!    total energy              =    -427.72181311 Ry
```

These are the total energies (Kohn-Sham potential energy surfaces) 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 $\Delta E_{\mathrm{KS}} = 0.117$ Ry and is due to the vibrational potential energy. To check this type:

```
$ grep -3 "Potent" ZG_333.out
```

```
    Total vibrational energy:    0.24020955  Ry

            Potential energy:    0.12010477  Ry

              Kinetic energy:    0.12010477  Ry
```

Our computed ΔE_{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 deviates from 0.12, since we exclude from our calculations the q-points belonging in set $\mathcal{A}$. As we use larger supercells this small deviation reduces to zero and $\Delta E_{\mathrm{KS}} = $ `Total vibrational energy / 2`.

For the next exercises we will only need the charge-density files from `si_333_ZG.save` and `si_333_equil.save`. Thus remove:

```
$ rm -r *wfc* si.save _ph0/ si_333_ZG.save/*wfc* si_333_equil.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 $PATHQE/bin/pw.x -nk 4 < si.scf.in > si.scf.out
$ ibrun -np 4 $PATHQE/bin/pw.x -nk 4 < si.bands.in > si.bands.out
$ ibrun -np 4 $PATHQE/bin/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.eps
```
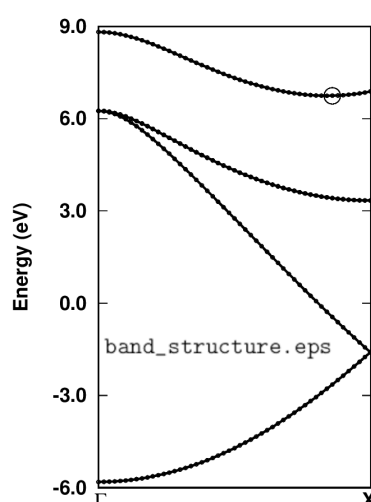

Can you find the k-coordinates of the valence band maximum (VBM) and conduction band minimum (CBM)? Inspecting file **bands.dat**, those are:

```
            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 ~ 0.84 Γ-X with energy $E_{\mathrm{VBM}} = 6.743$ 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: The selection rule here is to find the reciprocal lattice vector of the supercell **G** that maps **K** onto **k**, i.e. **K** = **k** − **G**. The trivial choice is **G** = Γ and thus set **K** = **k**.

To prepare the calculation proceed as follows:

```
$  cd ../workdir/
$ cp ../../exercise1/workdir/si.333_equil_scf.in si.333_equil_nscf.in
$ cp ../../exercise1/workdir/si.333_ZG_scf.in si.333_ZG_nscf.in
$ cp -r ../../exercise1/workdir/si_333_equil.save/ .
$ cp -r ../../exercise1/workdir/si_333_ZG.save/ .
```

In si.333_equil_nscf.in and si.333_ZG_nscf.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 tpiba
   2
   0.000000 0.000000 0.000000 1
   0.000000 2.520000 0.000000 1
   ```

   Note: 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 **k** = $[x\ y\ z]$ then **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 ZG input file add `nosym = .true.` below `nbnd = 135`.

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

```
$ cd ../workdir/; cp ../inputs/si.333_*in .; cp -r ../inputs/si_333_*.save/ .
```

For example the ZG input file should look like this:

```
&control                                                    si.333_ZG_nscf.in
    calculation = 'nscf'
    restart_mode = 'from_scratch',
    prefix = 'si_333_ZG',
    pseudo_dir = './',
```

```
    outdir='./'
 /
 &system
    ibrav = 2,
    celldm(1) = 30.60,
    nat =  54,
    ntyp = 1,
    ecutwfc = 20.0,
    nbnd = 135,
    nosum = .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
K_POINTS tpiba
2
0.000000 0.000000 0.000000 1
0.000000 2.520000 0.000000 1
ATOMIC_POSITIONS (angstroms)
...
```

```
$ ibrun -n 28 $PATHQE/bin/pw.x -nk 2 < si.333_equil_nscf.in > si.333_equil_nscf.out
$ ibrun -n 28 $PATHQE/bin/pw.x -nk 2 < si.333_ZG_nscf.in > si.333_ZG_nscf.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 si.333_equil_nscf.out
```

```
highest occupied, lowest unoccupied level (ev):    6.2534    6.7435
```

Indeed we obtain the correct VBM and CBM energies. To find the band gap renormalization from the ZG calculation type:

```
$ grep -20 "2.5200 0.0000" si.333_ZG_nscf.out
```

```
          k = 0.0000 0.0000 0.0000 ( 10849 PWs)   bands (ev):

   -5.8361  -4.5993  -4.5214  -4.4855  -4.4519  -4.3989  -4.3702  -4.3543
   -4.3259  -3.9599  -3.9345  -3.9007  -3.8625  -3.8262  -3.7843  -2.6373
   -2.6031  -2.5961  -2.5538  -2.5065  -2.4958  -2.4664  -2.4379  -2.4238
   -2.3878  -2.3609  -2.3180  -0.7144  -0.6732  -0.6366  -0.6186  -0.5833
   -0.5683  -0.5435  -0.4925  -0.4805  -0.4621  -0.4557  -0.3753   0.7632
    0.7805   0.7921   0.8091   0.8303   0.8865   0.9089   0.9363   1.1394
    1.2411   1.2465   1.2788   1.3025   1.3239   1.6239   1.6352   1.6696
    1.6774   1.6951   1.7128   1.7259   1.7532   1.7647   1.7932   1.8160
    1.8334   3.6567   3.6744   3.6843   3.7042   3.7369   3.7401   3.7606
    3.7938   3.8124   3.8199   3.8540   3.8849   4.0195   4.0238   4.0414
    4.0561   4.0829   4.0981   4.1093   4.1247   4.1602   4.1737   4.1873
    4.2367   5.0277   5.0774   5.1124   5.1781   5.1915   5.2172   5.2254
```

```
   5.2412    5.2611    5.2672    5.2923    5.3263    5.3399    5.3686    5.4193
   5.4217    6.2752    6.2792    6.2993    6.8042    6.8280    6.8919    6.9094
   6.9263    6.9677    7.6075    7.6294    7.6352    7.6701    7.6813    7.7004
   7.7411    7.7527    7.7653    7.8292    7.8424    7.8766    7.9676    7.9817
   7.9997    8.0110    8.0449    8.0545    8.0554    8.0587    8.0980

          k = 0.0000 2.5200 0.0000 ( 10874 PWs)   bands (ev):

  -5.7041   -4.8882   -4.7820   -4.7540   -4.7295   -4.6912   -4.0008   -3.9224
  -3.9091   -3.8537   -3.8188   -3.7520   -3.7368   -3.7016   -3.1325   -3.0720
  -3.0313   -3.0044   -2.8386   -2.7799   -2.7044   -2.6642   -2.6486   -1.9888
  -1.9341   -1.9027   -1.8561   -1.1309   -1.0645   -1.0469   -0.9998   -0.6386
  -0.6234   -0.5885   -0.5726   -0.2870   -0.0620   -0.0149    0.0123    0.0471
   0.0566    0.1038    0.1182    0.1543    0.9753    1.0533    1.0775    1.1071
   1.7249    1.7973    1.8214    1.8431    1.8717    1.8968    1.9093    1.9246
   1.9934    2.0154    2.0434    2.0672    2.2770    2.2962    2.3263    2.3704
   2.8278    3.2183    3.2580    3.2932    3.3111    3.3495    3.3706    3.3970
   3.4271    3.4452    3.4609    3.7507    3.7668    3.7925    3.8092    4.0534
   4.0792    4.0946    4.1193    4.1302    4.1892    4.2142    4.2458    4.3398
   4.3640    4.4322    4.4703    4.4947    4.5469    4.5988    4.6363    4.6650
   4.7037    4.7880    4.8214    4.8533    4.8868    5.4818    5.5178    5.5557
   5.6378    5.6880    5.8238    5.8456    6.7251    7.2578    7.3382    7.5455
   7.6026    7.6292    7.6557    7.8176    7.8502    7.8695    7.8974    7.9305
   7.9743    8.0155    8.0282    8.3419    8.3679    8.4031    8.4176    8.4775
   8.6171    8.6339    8.6496    8.6836    8.7102    8.7234    8.7447
```

Can you spot the energy of the VBM and CBM? We can see that there is a splitting of the originally threefold degenerated VBM of the order: $\Delta E = 6.2993 - 6.2752 = 25$ 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 to minimize the effect of band degeneracy splitting we exclude 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 $\Delta E > 150$ 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 renormalization by taking the average of the three bands: $E_{\mathrm{VBM}} = (6.2752 + 6.2792 + 6.2993)/3 = 6.2846$ eV. Thus the band gap at 0K is $E(0\,\mathrm{K}) = 6.7251 - 6.2846 = 0.4405$ eV, and we can determine a zero-point renormalization $E_{\mathrm{ZPR}} = 0.49 - 0.4329 \simeq 50$ meV. This value is in fair agreement with literature data (check, for example, Refs. [J. Chem. Phys. 143, 102813 (2015)] and [New J. Phys. 20, 123008 (2018)]).

Note: in general, for obtaining 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 below).

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:
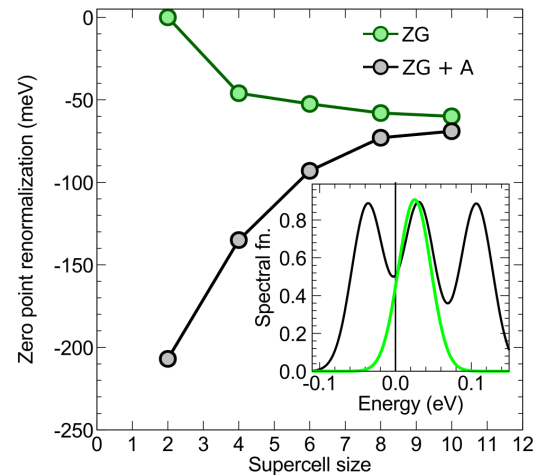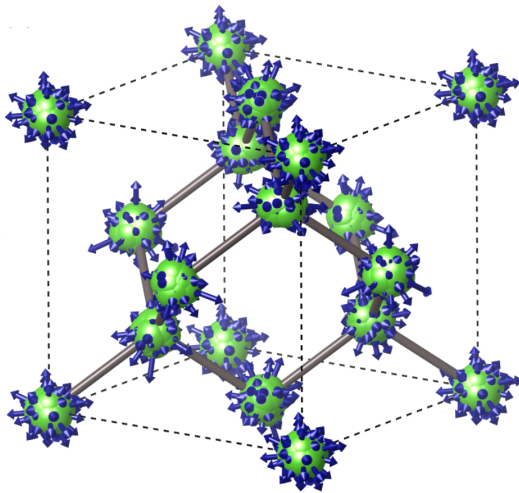
```
$ mkdir band_structure_unfolding/; cd band_structure_unfolding/
$ cp ../../inputs/bands_unfold.in bands.in; cp ../../inputs/Si.pz-vbc.UPF .
$ cp -r ../../../exercise1/workdir/si_333_ZG.save/ .
$ cp ../si.333_ZG_nscf.in si.333_ZG_bands
```

In si.333_ZG_bands apply the following modifications:

1. Set calculation = 'bands'

2. Delete the following lines containing information about the **K**-points:

   ```
   K_POINTS tpiba
   2
   0.000000 0.000000 0.000000 1
   0.000000 2.520000 0.000000 1
   ```

3. Delete the last 12 lines of the file and add the following (after ZG atomic positions):

   ```
   K_POINTS tpiba_b
   1
   ```

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

```
$ cp ../../inputs/si.333_ZG_bands .
```

The file si.333_ZG_bands should look like this:

```
&control                                                    si.333_ZG_bands
    calculation = 'bands'
    restart_mode = 'from_scratch',
    prefix = 'si_333_ZG',
    pseudo_dir = './',
    outdir='./'
/
&system
    ibrav = 2,
    celldm(1) = 30.60,
    nat =  54,
    ntyp = 1,
    ecutwfc = 20.0
    nbnd = 135
```

```
      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
ATOMIC_POSITIONS (angstroms)
   Si    0.07465546   -0.02027484    0.01183559
   Si    1.33329554    1.39151777    1.34662616
   Si   -2.60922970    0.02806964    2.69588796
...
K_POINTS tpiba_b
1
```

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

```
--                                                                    bands.in
&bands
 prefix  = 'si_333_ZG'
 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.

If we need to speed up the process we can skip the following step and go to the next one.

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

`$ $PATHQE/EPW/ZG_displacement/src/local/kpoints_band_str_unfold.x`

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 **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 1.0 0.0
 Write x-positions of high-sym kpts
0.0
1.0
 Step is (default):    2.85714287E-02
 Supercell size ?
3
 kpts to use for Supercell calculation:
    0.000000    0.000000    0.000000   35
    0.000000    3.000000    0.000000    1
```

```
 Write every single kpt? (0=no, 1=yes)
1
   0.000000    0.000000    0.000000    1
   0.000000    0.085714    0.000000    1
...
```

Note: for generating more **K**-points along the high-symmetry paths, you can increase the `step` value from the script:
`$PATHQE/EPW/ZG_displacement/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_displacement/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=si_333_ZG
i=1  # initial kpt index to calculate
f=35 # final kpt index to calculate
#
while [ $i -le $f ];do
    mkdir kpt_$i
    cd kpt_$i
#
    EXE=$PATHQE/bin/pw.x
    JNAME=si.333_ZG_bands
#
    cp -r ../"$prefix".save .
    mv ../"$JNAME"_"$i".in .
    ibrun -n 14 $EXE  < "$JNAME"_"$i".in > "$JNAME"_"$i".out
#
    EXE=$PATHQE/bin/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 read the outputs of **bands01.dat** and **spectral_weights01.dat**. Can you guess what's the meaning of the spectral weights for each band?

```
$ cat kpt_1/bands01.dat
&plot nbnd= 135, nks=      1 /
         0.000000  0.000000  0.000000
     -5.836093       -4.599302       -4.521405       -4.485488       -4.451935   ...
     -3.934487       -3.900722       -3.862462       -3.826158       -3.784317   ...
     -2.495809       -2.466409       -2.437924       -2.423761       -2.387836   ...
     -0.618566       -0.583327       -0.568251       -0.543510       -0.492492   ...
      0.780544        0.792145        0.809073        0.830272        0.886545   ...
      1.278808        1.302530        1.323948        1.623936        1.635247   ...
      1.753173        1.764726        1.793182        1.815957        1.833412   ...
      3.740133        3.760640        3.793828        3.812404        3.819916   ...
      4.056148        4.082880        4.098110        4.109307        4.124730   ...
      5.077351        5.112391        5.178059        5.191474        5.217247   ...
      5.326298        5.339917        5.368629        5.419329        5.421734   ...
      6.891922        6.909379        6.926339        6.967660        7.607521   ...
      7.741056        7.752693        7.765327        7.829213        7.842377   ...
      8.044942        8.054502        8.055361        8.058695        8.098015
```

```
$ cat kpt_1/spectral_weights01.dat
 &plot nbnd= 135, nks=      1 /
          0.000000   0.000000   0.000000
    0.988198     0.003257     0.001842     0.000097     0.000081   ...
    0.001300     0.000118     0.000163     0.000229     0.000226   ...
    0.000247     0.000198     0.000242     0.000215     0.000115   ...
    0.000204     0.000096     0.000144     0.000163     0.000189   ...
    0.000172     0.000179     0.000073     0.000262     0.000096   ...
    0.000124     0.000187     0.000260     0.000174     0.000269   ...
    0.000304     0.000267     0.000093     0.000200     0.000271   ...
    0.000590     0.001314     0.000212     0.000641     0.000694   ...
    0.001097     0.000963     0.000526     0.001391     0.000944   ...
    0.002419     0.007162     0.001575     0.002939     0.003744   ...
    0.001136     0.007393     0.003963     0.008301     0.003408   ...
    0.006582     0.003639     0.003098     0.002384     0.003624   ...
    0.004435     0.003644     0.002502     0.002284     0.005629   ...
    0.001350     0.006065     0.001782     0.003933     0.005753
```

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. If the calculation did not finish yet, we can use the time for some questions. After the calculations for all **K**-points are completed you can merge them all together in a new directory all_kpts using:

```
$ 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 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 plot_bands.x of QE:

```
$ cp ../../../inputs/energies.in .
$ cp ../../../inputs/spectral_weights.in .
$ $PATHQE/bin/plotband.x spectral_weights01.dat < spectral_weights.in > spw.out
$ $PATHQE/bin/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:

```
$ cp ../../../inputs/energies_weights.dat . # if you missed the step above
$ cp ../../../inputs/pp_spctrlfn.in .
```

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

```
--                                                                   pp_spctrlfn.in
&input
  flin   = 'energies_weights.dat'
  steps  = 4725,
  ksteps = 200,
  esteps = 200,
  kmin   = 0,
  kmax   = 1.0,
  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. More details about the input flags are provided in tuto_Fri6_flags.pdf. Now calculate the spectral function using pp_spctrlfn.x and plot:

```
$ ibrun -n 4 $PATHQE/bin/pp_spctrlfn.x -nk 4 < pp_spctrlfn.in > pp_spctrlfn.out
$ cp spectral_function.dat ../../../gnuplot/. ; cd ../../../gnuplot/
$ cp ../inputs/bands.dat.gnu .
$ gnuplot gp_pm3d.p; evince bsu_si_0K.eps
```

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). 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.

Exercise 3

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)].
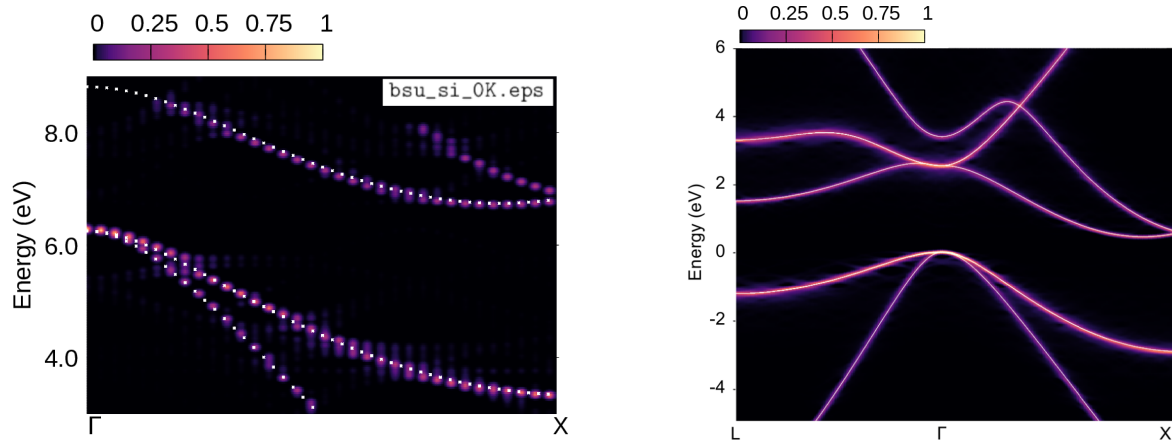
Note: The calculation of phonon-assisted optical spectra using ZG configurations has some drawbacks and advantages when compared to the perturbative methodology you have seen in the tutorial Thur.5 (see also lecture notes).

First go to the directory `exercise3`, create your working directories and copy the input files:

```
$ cd ../../exercise3/; 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/si_333_equil.save/ .
$ cp ../../../exercise2/workdir/si.333_equil_nscf.in si.333_equil_nscf
```

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, in order 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 tpiba
   2
   0.000000 0.000000 0.000000 1
   0.000000 2.520000 0.000000 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.

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

```
$ cp ../../inputs/equil/si.333_equil_nscf .
$ cp -r ../../inputs/equil/si_333_equil.save/ .
```

The file `si.333_equil_nscf` should look like this:

```
&control                                                      si.333_equil_nscf
   calculation = 'nscf'
   restart_mode = 'from_scratch',
   prefix = 'si_333_equil',
```

```
   pseudo_dir = './',
   outdir='./'
/
&system
   ibrav = 2,
   celldm(1) = 30.60,
   nat =  54,
   ntyp = 1,
   ecutwfc = 20.0
   nbnd = 135
   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
ATOMIC_POSITIONS (angstroms)
   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
```

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

```
--                                                                   epsilon.in
&inputpp
outdir       = './',
prefix       = 'si_333_equil',
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:

$$\epsilon_2(\omega) = \frac{2\pi}{m_e N_e} \frac{1}{N_\mathbf{K}} \frac{\omega_p^2}{\omega^2} \sum_{cv\mathbf{K}} |p_{cv\mathbf{K}}|^2 \delta(\varepsilon_{c\mathbf{K}} - \varepsilon_{v\mathbf{K}} - \hbar\omega), \tag{2}$$

where m_e is the electron mass, N_e is the number of electrons in the crystal unit cell, ω_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. In the same spirit with the BSU calculation we calculate $\epsilon_2(\omega)$ for every **K**-point using the following script:

```
prefix=si_333_equil
i=1  # initial kpt index to calculate
f=30 # final kpt index to calculate

while [ $i -le $f ];do
#
    JNAME=si.333_equil_nscf
    awk 'NR == '$i'' K_list.in > K_point.txt
    cat $JNAME K_point.txt > "$JNAME"_"$i".in
#
    mkdir kpt_$i
    cd kpt_$i
#
    EXE=$PATHQE/bin/pw.x
#
    cp -r ../"$prefix".save .
    mv ../"$JNAME"_"$i".in .
    ibrun -n 14 $EXE  < "$JNAME"_"$i".in > "$JNAME"_"$i".out
#
    EXE=$PATHQE/bin/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_si_333_equil.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 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 ../../inputs/ZG/epsi_av.sh .
$ cp -r ../../../exercise1/workdir/si_333_ZG.save/ .
$ cp ../../../exercise2/workdir/si.333_ZG_nscf.in si.333_ZG_nscf
```

In si.333_ZG_nscf apply the following change:

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

   ```
   K_POINTS tpiba
   2
   0.000000 0.000000 0.000000 1
   0.000000 2.520000 0.000000 1
   ```

2. Delete the last 12 lines of the file and add the following (**after** ZG atomic positions):

   ```
   K_POINTS crystal
   1
   ```

If you did not have time to complete Exercises 1 and 2, you can copy si_333_ZG.save/ and si.333_ZG_nscf from ../../inputs, i.e.:

```
$ cp ../../inputs/ZG/si.333_ZG_nscf .; cp -r ../../inputs/ZG/si_333_ZG.save/ .
```

The file si.333_ZG_nscf should look like this:

```
&control                                                          si.333_ZG_nscf
   calculation = 'nscf'
   restart_mode = 'from_scratch',
   prefix = 'si_333_ZG',
   pseudo_dir = './',
   outdir='./'
/
&system
   ibrav = 2,
   celldm(1) = 30.60,
   nat =  54,
   ntyp = 1,
   ecutwfc = 20.0
   nbnd = 135
   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
ATOMIC_POSITIONS (angstroms)
   Si    0.07465546  -0.02027484   0.01183559
   Si    1.33329554   1.39151777   1.34662616
   Si   -2.60922970   0.02806964   2.69588796
...
K_POINTS crystal
1
```

epsilon.in is the same with the one used before, but with `prefix = 'si_333_ZG'`.
Now run the following script that performs a loop over the **K**-points to calculate $\epsilon_2(\omega)$:

```
prefix=si_333_ZG
i=1  # initial kpt index to calculate
f=30 # final kpt index to calculate

while [ $i -le $f ];do
#
    JNAME=si.333_ZG_nscf
    awk 'NR == '$i'' K_list.in > K_point.txt
    cat $JNAME K_point.txt > "$JNAME"_"$i".in
#
    mkdir kpt_$i
    cd kpt_$i
#
    EXE=$PATHQE/bin/pw.x
#
    cp -r ../"$prefix".save .
    mv ../"$JNAME"_"$i".in .
    ibrun -n 14 $EXE  < "$JNAME"_"$i".in > "$JNAME"_"$i".out
#
    EXE=$PATHQE/bin/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/ZG/script_ZG.sh .`

This script will generate the directories **kpt_X**, containing $\epsilon_2(\omega)$ in the file **epsi_si_333_ZG.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.

▶ Once the calculation is finished, run the script:

`$ ./epsi_av.sh`

which does the same job as before. The output file containing the data is **epsi_si_333_ZG_30.dat**.

Now copy the output file in `gnuplot` directory and plot ZG and equilibrium spectra using:

```
$  cp epsi_si_333_ZG_30.dat ../../gnuplot/.
$  cd ../../gnuplot/; gnuplot gp_coms.p; evince ZG_spectra.eps
$  gnuplot gp_coms_logscale.p; evince ZG_spectra_logscale.eps
```

You can also plot $\varepsilon_2(\omega)$ as calculated with more **K**-points (outputs are provided):

```
$ cp ../outputs/ZG/epsi_si_333_ZG_*dat .
$ cp ../outputs/equil/epsi_si_333_equil_*dat .
$ gnuplot gp_coms_logscale_kpts.p; evince ZG_spectra_logscale_kpts.eps
$ gnuplot gp_coms_logscale_200kpts.p; evince ZG_spectra_logscale_200kpts.eps
```
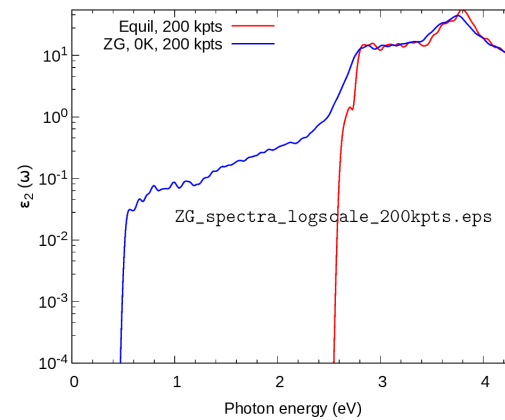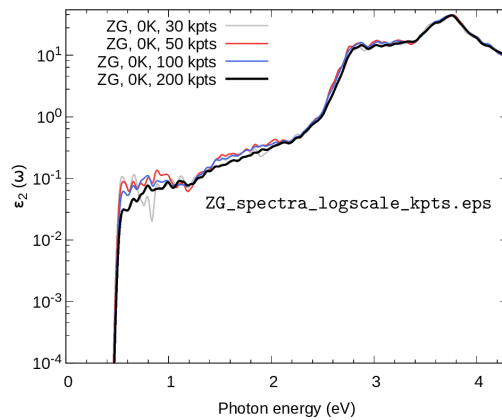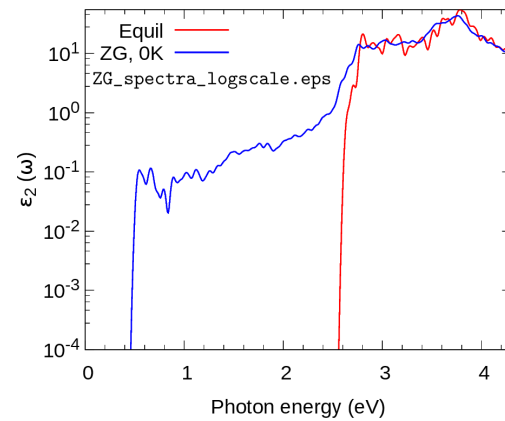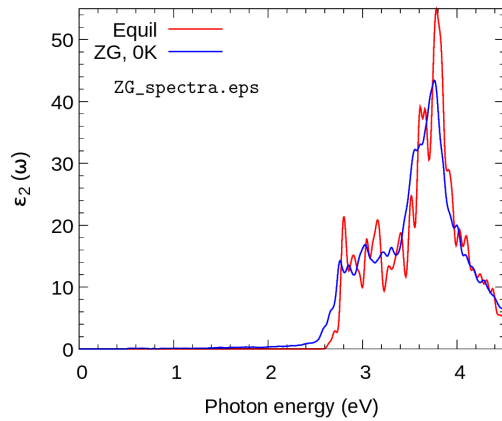

What are the differences between the ZG and equilibrium spectra?
Are phonon-assisted transitions captured as expected?
Ref. [Phys. Rev. B 94, 075125 (2016)] reports the convergence of the spectra with respect to the
ZG supercell size and the agreement with experimental data for the absorption coefficient.

Exercise 4

This is an advanced exercise; it is left as homework

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. [https://arxiv.org/abs/2104.07900].

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 `exercise4`, create your working directories, and copy the input files:

```
$ cd ../../exercise4/; mkdir workdir; cd workdir; mkdir exact; mkdir ZG
$  cd exact
$ cp ../../inputs/exact/graphene.881.fc .
$ cp ../../inputs/exact/disca.in .
$ cp ../../inputs/exact/pp_disca.in .
```

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

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

```
--                                                                    disca.in
&input
    asr = 'crystal', amass(1) = 12.011, flfrc = 'graphene.881.fc', atm_zg(1) = 'C',
    dim1 = 40, dim2 = 40, dim3 = 1,
    T = 300,
    atmsf_zg_a(1,1) = 0.1361,
    atmsf_zg_a(1,2) = 0.5482,
    atmsf_zg_a(1,3) = 1.2266,
    atmsf_zg_a(1,4) = 0.5971,
    atmsf_zg_a(1,5) = 0.0,
    atmsf_zg_b(1,1) = 0.3731,
    atmsf_zg_b(1,2) = 3.2814,
    atmsf_zg_b(1,3) = 13.0456,
    atmsf_zg_b(1,4) = 41.0202,
    atmsf_zg_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,
/
```

```
$ ibrun -np 56 $PATHQE/bin/disca.x -nk 56 < disca.in > disca.out
```

The input format of `disca.in` is the same as `ZG_333.in`. Line 3 contains input variables similar to those in matdyn.in; in line 4 we provide the dimensions `dimX` of the **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_zg_a` and `atmsf_zg_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) and 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.

More details about all input flags are provided in `tuto_Fri6_flags.pdf`.

The outputs from a `disca.x` run are: **Bragg_scattering.dat**, **structure_factor_one-phonon.dat**, **structure_factor_q_nu_one-phonon.dat**, and **structure_factor_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. `disca.x` also prints the **Q** vectors in crystal coordinates in **qpts_strf.dat** (to be used for computing the ZG scattering intensity).

▶ Rotate the coordinates of the all-phonon scattering intensity to obtain the complete map. The number of rotations should be based on the space group of each system. This step is important in order 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).

`$ $PATHQE/EPW/ZG_displacement/src/local/rotate.x`

```
 Write number of rotations based on symmetry (integer)
6
 Write number of entries (integer)
40000
```

Note: we apply a six-fold rotation (hexagonal symmetry of graphene) of the scattering map containing 40000 **Q**-points.

`rotate.x` will generate a new file **structure_factor_all-phonon_rot.dat** which contains the original map rotated 6 times.

▶ Run `pp_disca.x` calculation to apply a broadening to your raw data using the input:

```
--                                                                          pp_disca.in
&input
  flstrfin  = 'structure_factor_all-phonon_rot.dat'
  steps   = 240000,
  ksteps1 = 250,
  ksteps2 = 250,
  dim1    = 1,
  dim2    = 2,
  kmin    = -10,
  kmax    = 10,
  Np      = 1600,
  flstrfout = 'structure_factor_all-phonon_broad.dat'
/
```

`$ ibrun -np 56 $PATHQE/bin/pp_disca.x -nk 56 < pp_disca.in > pp_disca.out`

Here, (i) `steps` is the number of rows in the input file `flstrfin`, (ii) `ksteps1` and `ksteps2` define the resolution of the scattering intensity along the chosen Cartesian axes, (iii) `dim1` and `dim2` are integers defining two of the Cartesian directions of the scattering vectors (where 1 --> x, 2 --> y, and 3 --> z) (i.e used to pick two columns of `flstrfin`) (iv) (`kmin - kmax`) define the 2D momentum window in reciprocal space, (v) `Np` is the number of reduced wavevectors (**q**-points) used to sample each Brillouin zone, and (vi) in `flstrfout` we provide the name of the output file. More details about the input flags are provided in `tuto_Fri6_flags.pdf`. Now copy the output file `structure_factor_all-phonon_broad.dat` in the gnuplot directory:

`$ cp structure_factor_all-phonon_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 ../../inputs/ZG/pp_disca.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='crystal', amass(1)= 12.011, flfrc= 'graphene.881.fc', atm_zg(1) = 'C'
    dim1 = 40, dim2 = 40, dim3 = 1,
    T = 300,
    atmsf_zg_a(1,1) = 0.1361,
    atmsf_zg_a(1,2) = 0.5482,
    atmsf_zg_a(1,3) = 1.2266,
    atmsf_zg_a(1,4) = 0.5971,
    atmsf_zg_a(1,5) = 0.0,
    atmsf_zg_b(1,1) = 0.3731,
    atmsf_zg_b(1,2) = 3.2814,
    atmsf_zg_b(1,3) = 13.0456,
    atmsf_zg_b(1,4) = 41.0202,
    atmsf_zg_b(1,5) = 0.0,
    synch = .true., compute_error = .false.
    incl_qA = .true.,
    ZG_strf = .true., qpts_strf = 40000
/
```

```
$ ibrun -np 56 $PATHQE/bin/ZG.x -nk 56 < ZG_strf.in > ZG_strf.out
```

The input variables in the first 15 lines of `ZG.in` are the same as in `disca.in` (but strictly speaking `dimX`, here, is for the supercell size dimensions); in line 16 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 17 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 18 we ask the code to compute the structure factor (`ZG_strf = .true.`) for 40000 **Q**-points (`qpts_strf = 40000`) from file `qpts_strf.dat`. More details about all input flags are provided in `tuto_Fri6_flags.pdf`.

The calculations should take around 10 secs. The important outputs from this `ZG.x` run are: **ZG-configuration_0.050.dat** and **structure_factor_ZG.dat**, containing the collection of ZG atomic coordinates (as before) and the associated scattering intensity, respectively. File **structure_factor_ZG.dat** has 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.

▶ Rotate the coordinates of the ZG scattering intensity to obtain the complete map:

```
$ cp structure_factor_ZG.dat structure_factor_all-phonon.dat
$ $PATHQE/EPW/ZG_displacement/src/local/rotate.x
```

```
 Write number of rotations based on symmetry (integer)
6
 Write number of entries (integer)
40000
```

Note: we apply a six-fold rotation (hexagonal symmetry of graphene) of the scattering map containing 40000 **Q**-points.

`rotate.x` will generate a new file **structure_factor_all-phonon_rot.dat** which contains the original map rotated 6 times.

▶ Run `pp_disca.x` calculation to apply a broadening to your raw data as before.

```
$ ibrun -np 56 $PATHQE/bin/pp_disca.x -nk 56 < pp_disca.in > pp_disca.out
```

Now copy the output file `structure_factor_all-phonon_broad.dat` in the gnuplot directory and plot your ZG and exact data:

```
$ cp structure_factor_all-phonon_broad.dat ../../gnuplot/ZG
$ cd ../../gnuplot/ZG; gnuplot gp_pm3d.p; evince strf_ZG_40_40.eps
$ cd ../exact; gnuplot gp_pm3d.p; evince strf_exact_40_40.eps
```

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 (data are provided in the `outputs` directory). 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.