Electron-phonon matrix elements and coupling strength

Tutorial Tue.5

Hands-on session

Hands-on based on Quantum ESPRESSO development version and EPW v5.4Beta

In this tutorial session we will learn how to use the core capabilities of EPW. In particular, we will focus on how to check the quality of Wannier interpolation of physical quantities for a future production run.

You are advised to run the calculations in your scratch directory and follow the instructions carefully. Unless you are an experienced user of EPW and Quantum ESPRESSO, do not change the directory structure and the parameters in jobscript.

You can start by creating the working directory in your scratch folder and copying the tar file which contains all necessary files for this tutorial and extracting it. Then, go to the directory of the first exercise:

```
$ cd $SCRATCH
$ mkdir EPW-SCHOOL ; cd EPW-SCHOOL
$ cp /work2/06868/giustino/EPW-SCHOOL/Tue.5.Lee.tar .
$ tar -xvf Tue.5.Lee.tar ; cd Tue.5.Lee/exercise1
```

Exercise 1

In this exercise we will calculate physical properties of metallic Pb using Quantum ESPRESSO and repeat the same calculations through Wannier interpolation using EPW; then we will compare them to check the interpolation quality. After assessing interpolation quality, we will calculate the phonon linewidth for Pb.

▶ 1st step: Run a self-consistent calculation on a homogeneous $8 \times 8 \times 8$ $\mathbf{k}$-point grid and a phonon calculation on a homogeneous $3 \times 3 \times 3$ $\mathbf{q}$-point grid using the following inputs and jobscript:
Note: The energy cutoff ecutwfc needed for convergence should be 90 Ry.

```
$ cd phonon
$ sbatch job.ph
```

job.ph

```
#!/bin/bash
#SBATCH -J myjob            # Job name
#SBATCH -p small     # Queue (partition) name
#SBATCH -N 1                # Total # of nodes
#SBATCH --ntasks-per-node 8
#SBATCH -t 01:00:00         # Run time (hh:mm:ss)
#SBATCH -A EPW-SCHOOL
#SBATCH --reservation=EPW-SCHOOL-06-15-2021

module list
pwd
date

# Launch MPI code...
export PATHQE=/work2/06868/giustino/EPW-SCHOOL/q-e
```

```
ibrun $PATHQE/bin/pw.x -nk 4 -in pb.scf.in > pb.scf.out
ibrun $PATHQE/bin/ph.x -nk 4 -in pb.ph.in > pb.ph.out

date
```

```
--                                                                      pb.scf.in
 &control
    calculation='scf'
    restart_mode='from_scratch',
    prefix='pb',
    pseudo_dir = '../',
    outdir='./'
 /
 &system
    ibrav=  2,
    celldm(1) = 9.2225583816,
    nat= 1,
    ntyp= 1,
    ecutwfc = 30.0
    occupations='smearing',
    smearing='marzari-vanderbilt',
    degauss=0.05
 /
 &electrons
    conv_thr =  1.0d-10
    mixing_beta = 0.7
 /
ATOMIC_SPECIES
 Pb 207.2 pb_s.UPF
ATOMIC_POSITIONS
 Pb 0.00 0.00 0.00
K_POINTS {automatic}
 8 8 8 0 0 0
```

```
--                                                                       pb.ph.in
&inputph
prefix = 'pb',
fildyn = 'pb.dyn',
fildvscf = 'dvscf',
ldisp = .true.,
nq1 = 3,
nq2 = 3,
nq3 = 3,
tr2_ph = 1.0d-12
/
```

The keyword fildvscf in pb.ph.in tells the code to write on file the change of the self-consistent potential due to phonon perturbations, $\partial_{\mathbf{q}\nu}V^{\rm scf}$, that is needed to compute the electron-phonon matrix elements.
In the output file of pb.ph.out, locate the list of 4 irreducible $\mathbf{q}$ points in the Brillouin Zone (BZ):

```
Dynamical matrices for ( 3, 3, 3)  uniform grid of q-points
(   4 q-points):
  N         xq(1)         xq(2)         xq(3)
  1   0.000000000   0.000000000   0.000000000
  2  -0.333333333   0.333333333  -0.333333333
  3   0.000000000   0.666666667   0.000000000
  4   0.666666667  -0.000000000   0.666666667
```

The list of irreducible $\mathbf{q}$ points is also written in the pb.dyn0 file. If you type `ls`, you can see pb.dynX files containing the dynamical matrix for each irreducible $\mathbf{q}$ point. The dvscf files are all named `pb.dvscf1` and are located inside the `_ph0/pb.q_X/` folders, except for the one corresponding to the first $\mathbf{q}$ point (Γ) that is located in `_ph0/`.

▶ 2nd step: Calculate the phonon dispersion by using Fourier interpolation. This can be done in the following two steps:
(1) Bring the interatomic force constant to real space using a Fourier transformation with the program of `q2r.x`:

```
$ /work2/06868/giustino/EPW-SCHOOL/q-e/bin/q2r.x -in q2r.in > q2r.out
```

```
--                                                                    q2r.in
&input
  zasr='simple', fildyn='pb.dyn', flfrc='pb333.fc'
/
```

Note in the output if the Fast Fourier transform (FFT) was successful:

```
fft-check success (sum of imaginary terms < 10^-12)
```

The matrix of interatomic force constants in real space should have been created (file `pb333.fc`).

(2) Calculate the phonon dispersion along the high-symmetry lines using the program of `matdyn.x`:

```
$ /work2/06868/giustino/EPW-SCHOOL/q-e/bin/matdyn.x -in matdyn.in > matdyn.out
```

```
--                                                                 matdyn.in
&input
  asr              = 'simple',
  flfrc            = 'pb333.fc'
  flfrq            = 'pb.freq'
  q_in_band_form   = .true.
  q_in_cryst_coord = .true.
/
  7
  0.000 0.000 0.000  30
  0.500 0.000 0.500  30
  0.500 0.250 0.750  30
  0.500 0.500 0.500  30
  0.375 0.375 0.750  30
  0.000 0.000 0.000  30
  0.500 0.500 0.500  30
```

This produces the file named `pb.freq.gp` with the phonon frequencies along the path, expressed in cm^{-1} unit. This will be checked against the phonon frequencies along the same path from `EPW`.

▶ 3rd step: Gather the `.dyn` and `.dvscf` files into a new `save/` directory which `EPW` will read. The files in `_ph0/pb.phsave/` containing the displacement patterns are also needed. This can easily be done using the `pp.py` python script which is already included in the `EPW` distribution:

```
$ python3 /work2/06868/giustino/EPW-SCHOOL/q-e/EPW/bin/pp.py
```

The script will ask you to enter the `prefix` of your calculation (pb):

```
Enter the prefix used for PH calculations (e.g. diam)
pb
```

▶ 4th step: Run a non self-consistent calculation for electronic band structures using the charge density and other outputs from the previous self-consistent run:

```
$ cd ../band
$ sbatch job.band
```

job.band

```
#!/bin/bash
#SBATCH -J myjob           # Job name
#SBATCH -p small    # Queue (partition) name
#SBATCH -N 1               # Total # of nodes
#SBATCH --ntasks-per-node 8
#SBATCH -t 01:00:00        # Run time (hh:mm:ss)
#SBATCH -A EPW-SCHOOL
#SBATCH --reservation=EPW-SCHOOL-06-15-2021

module list
pwd
date

# Launch MPI code...
export PATHQE=/work2/06868/giustino/EPW-SCHOOL/q-e

mkdir pb.save
cp ../phonon/pb.save/charge-density.dat pb.save/
cp ../phonon/pb.save/data-file-schema.xml pb.save/

ibrun $PATHQE/bin/pw.x -nk 4 -in pb.band.in > pb.band.out

date
```

pb.band.in

```
--
 &control
    calculation='bands'
    restart_mode='from_scratch',
    prefix='pb',
    pseudo_dir = '../',
    outdir='./'
 /
 &system
    ibrav=  2,
    celldm(1) = 9.2225583816,
    nat= 1,
    ntyp= 1,
    ecutwfc = 30.0
    occupations='smearing',
    smearing='marzari-vanderbilt',
    degauss=0.05
    nbnd=10
 /
 &electrons
    conv_thr =  1.0d-10
    mixing_beta = 0.7
 /
ATOMIC_SPECIES
 Pb 207.2 pb_s.UPF
ATOMIC_POSITIONS
 Pb 0.00 0.00 0.00
K_POINTS crystal_b
  7
  0.000 0.000 0.000  30
  0.500 0.000 0.500  30
  0.500 0.250 0.750  30
  0.500 0.500 0.500  30
  0.375 0.375 0.750  30
  0.000 0.000 0.000  30
  0.500 0.500 0.500  30
```

Then, run the program of `bands.x` to obtain the band structure data:

```
$ /work2/06868/giustino/EPW-SCHOOL/q-e/bin/bands.x -in pb.bands.in > pb.bands.out
```

```
--                                                                    pb.bands.in
&BANDS
  prefix = 'pb'
  lsym   = .false.
/
```

This will produce the file named `bands.out.gnu` which will be compared with the interpolated electronic band structure from EPW.

▶ 5th step: Run a non self-consistent calculation on a homogeneous $6 \times 6 \times 6$ $\mathbf{k}$-point grid for generating the input of wave functions for EPW and an EPW calculation for the Wannier interpolation of electronic band structure and phonon dispersion along the high-symmetry lines:

```
$ cd ../epw
$ sbatch job.epw1
```

```
                                                                      job.epw1
#!/bin/bash
#SBATCH -J myjob           # Job name
#SBATCH -p small    # Queue (partition) name
#SBATCH -N 1               # Total # of nodes
#SBATCH --ntasks-per-node 8
#SBATCH -t 01:00:00        # Run time (hh:mm:ss)
#SBATCH -A EPW-SCHOOL
#SBATCH --reservation=EPW-SCHOOL-06-15-2021

module list
pwd
date

# Launch MPI code...
export PATHQE=/work2/06868/giustino/EPW-SCHOOL/q-e

mkdir pb.save
cp ../phonon/pb.save/charge-density.dat pb.save/
cp ../phonon/pb.save/data-file-schema.xml pb.save/

ibrun $PATHQE/bin/pw.x -nk 4 -in pb.nscf.in > pb.nscf.out
ibrun $PATHQE/bin/epw.x -nk 8 -in pb.epw1.in > pb.epw1.out

date
```

```
--                                                                    pb.nscf.in
 &control
    calculation='nscf'
    restart_mode='from_scratch',
    prefix='pb',
    pseudo_dir = '../',
    outdir='./'
 /
 &system
    ibrav=  2,
    celldm(1) = 9.2225583816,
    nat= 1,
    ntyp= 1,
    ecutwfc = 30.0
    occupations='smearing',
    smearing='marzari-vanderbilt',
    degauss=0.05
    nbnd=10
 /
 &electrons
    conv_thr =  1.0d-10
    mixing_beta = 0.7
 /
ATOMIC_SPECIES
 Pb 207.2 pb_s.UPF
ATOMIC_POSITIONS
 Pb 0.00 0.00 0.00
K_POINTS crystal
216
  0.00000000  0.00000000  0.00000000  4.629630e-03
  0.00000000  0.00000000  0.16666667  4.629630e-03
  0.00000000  0.00000000  0.33333333  4.629630e-03
...
```

```
--                                                                                                   pb.epw1.in
&inputepw
  prefix      = 'pb',
  amass(1)    = 207.2
  outdir      = './'
  dvscf_dir   = '../phonon/save'

  elph        = .true.
  epwwrite    = .true.
  epwread     = .false.

  wannierize  = .true.
  nbndsub     =  4
  bands_skipped = 'exclude_bands = 1-5'
  num_iter    = 300
  dis_win_max = 21
  dis_froz_max= 13.5
  proj(1)     = 'Pb:sp3'
  wannier_plot= .true.

  band_plot   = .true.

  filkf       = 'path2.dat'
  filqf       = 'path2.dat'

  nk1         = 6
  nk2         = 6
  nk3         = 6
  nq1         = 3
  nq2         = 3
  nq3         = 3
```

```
/
```

Note 1: In some cases, pw.x calculates additional $\mathbf{k}$ points which are not provided in the $\mathbf{k}$-point list of the input. If this happens, you need to use the keyword of `calculation='bands'` instead of `calculation='nscf'`. Also note that the $\mathbf{k}$ and $\mathbf{q}$ grids need to be **commensurate**, with the $\mathbf{k}$ grid at least of the size of the $\mathbf{q}$ grid. Since we chose a $6 \times 6 \times 6$ $\mathbf{k}$-point grid, the $3 \times 3 \times 3$ $\mathbf{q}$-point grid used in the phonon calculation is appropriate, however a $6 \times 6 \times 6$ $\mathbf{q}$-point grid would be needed in order to interpolate more accurately the dynamical matrix and the electron-phonon matrix elements.
Note 2: The positive-definite homogeneous $6 \times 6 \times 6$ $\mathbf{k}$-point grid between 0 and 1 in pb.nscf.in can be generated by using the script of kmesh.pl included in the Wannier90 package:

```
$ /work2/06868/giustino/EPW-SCHOOL/q-e/wannier90-3.1.0/utility/kmesh.pl 6 6 6
```

Note 3: In `dvscf_dir   = '../phonon/save'` we specify the directory where the .dyn, .dvscf and patterns files are stored.

With this input, EPW will perform the following main steps before interpolating the electronic band structure and the phonon dispersion along the selected path:

- Wannierization using Wannier90 as a library. In the output of pb.epw1.out you can find the Wannier function centers and spreads obtained:

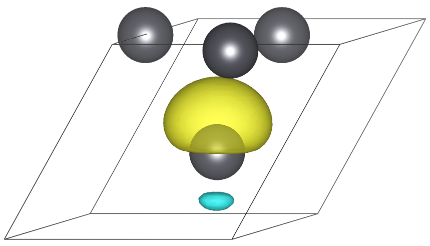
```
 Wannier Function centers (cartesian, alat) and spreads (ang):

 (    0.07779    0.07779    0.07779) :    2.22274
 (    0.07779   -0.07779   -0.07779) :    2.22274
 (   -0.07779    0.07779   -0.07779) :    2.22274
 (   -0.07779   -0.07779    0.07779) :    2.22274
```

  while the full output from the Wannier90 run is in the file of pb.wout.

  The input keywords for the Wannierization are in the block following wannierize = .true.:

  nbndsub corresponds to the number of Wannier functions (4, starting from Pb sp^3 orbitals as the initial guess), while `bands_skipped = ...` specifies the list of bands not wannierized (generally a set of bands lying at lower energies, such as semicore states in this example). For the other input keywords you can refer to the page at https://docs.epw-code.org/doc/Inputs.html. It is also possible to add extra keywords that are read by Wannier90 by using the input keyword of wdata(index) with increasing index number.

  In addition, with the input keyword of wannier_plot= .true. we can generate directly and efficiently in EPW the cube files containing real-space Wannier functions which can be plotted by using the plotting softwares such as VMD and VESTA; the plot of 3rd Wannier function should look like this:



- Calculate the electron-phonon matrix elements on the $(\mathbf{k}, \mathbf{k}+\mathbf{q})$ points for each irreducible $\mathbf{q}$-point in the Brillouin zone and unfold to the full Brillouin zone using symmetries. This is the most expensive part of the run. In the output you can see:

```
...
   =====================================================================
   irreducible q point #    1
   =====================================================================

   Symmetries of small group of q: 48
        in addition sym. q -> -q+G:

   Number of q in the star =    1
   List of q in the star:
        1   0.000000000   0.000000000   0.000000000
   Imposing acoustic sum rule on the dynamical matrix

      q(    1 ) = (   0.0000000   0.0000000   0.0000000 )
...
```

- Transform all quantities from reciprocal (Bloch) space to real (Wannier) space and store on file the resulting matrices and additional information needed for restarting a calculation (`pb.epmatwp, crystal.fmt, vmedata.fmt (or dmedata.fmt), and epwdata.fmt` files):

```
     Writing Hamiltonian, Dynamical matrix and EP vertex in Wann rep to file
```

We should always check whether the Wannier-interpolated electron and phonon band structures match well to those calculated using `pw.x` and `matdyn.x`, respectively. With the input flag of `band_plot = .true.` in `pb.epw1.in` we instructed EPW to save on files the interpolated electronic bands (`band.eig`) and phonon frequencies (`phband.freq`) and we chose to interpolate them onto the same Brillouin-zone path as previous `pw.x` and matdyn.x runs; this path is read from the file of `path2.dat` which is specified by the keywords of filkf and filqf.

To extract easily the data to plot, you can simply run the program of `plotband.x` in the `Quantum ESPRESSO` package and enter the input file (`band.eig` or `phband.freq`), the energy range (for example, -1,20), the output file with the data to plot (`band.dat` or `freq.dat`); other inputs are not relevant and simply push the ENTER key when asked:

```
$ /work2/06868/giustino/EPW-SCHOOL/q-e/bin/plotband.x
     Input file > band.eig
Reading    4 bands at    181 k-points
Range:   -0.4085   19.0142eV  Emin, Emax, [firstk, lastk] > -1,20
high-symmetry point:  0.0000 0.0000 0.0000   x coordinate   0.0000
high-symmetry point: -1.0000 0.0000 0.0000   x coordinate   1.0000
high-symmetry point: -1.0000 0.5000 0.0000   x coordinate   1.5000
high-symmetry point: -0.5000 0.5000 0.5000   x coordinate   2.2071
```

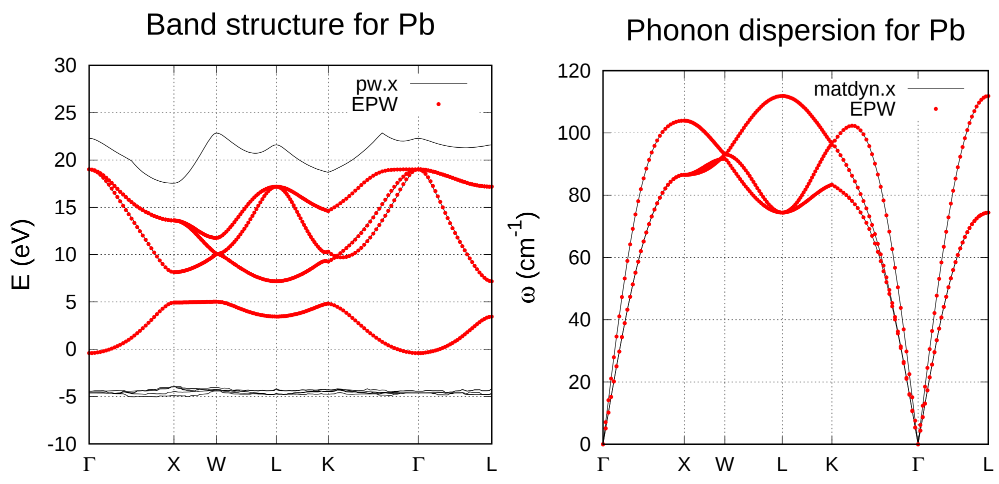


Figure 1

```
high-symmetry point: -0.7500 0.7500 0.0000   x coordinate   2.8195
high-symmetry point:  0.0000 0.0000 0.0000   x coordinate   3.8801
high-symmetry point: -0.5000 0.5000 0.5000   x coordinate   4.7462
output file (gnuplot/xmgr) > band.dat
bands in gnuplot/xmgr format written to file band.dat

output file (ps) >
stopping ...

$ /work2/06868/giustino/EPW-SCHOOL/q-e/bin/plotband.x
     Input file > phband.freq
Reading    3 bands at     181 k-points
Range:    0.0000   13.8675eV  Emin, Emax, [firstk, lastk] > 0,14
high-symmetry point:  0.0000 0.0000 0.0000   x coordinate   0.0000
high-symmetry point: -1.0000 0.0000 0.0000   x coordinate   1.0000
high-symmetry point: -1.0000 0.5000 0.0000   x coordinate   1.5000
high-symmetry point: -0.5000 0.5000 0.5000   x coordinate   2.2071
high-symmetry point: -0.7500 0.7500 0.0000   x coordinate   2.8195
high-symmetry point:  0.0000 0.0000 0.0000   x coordinate   3.8801
high-symmetry point: -0.5000 0.5000 0.5000   x coordinate   4.7462
output file (gnuplot/xmgr) > freq.dat
bands in gnuplot/xmgr format written to file freq.dat

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

You can now compare the interpolated electronic and phonon band structures with those from the previous runs with `pw.x` (4th step) and `matdyn.x` (2nd step). If you plot these results together, you should obtain the plot in Figure 1:
(assume you connected with X11 support on Frontera and you are in the epw folder)

```
$ cp ../band/bands.out.gnu .
$ cp ../phonon/pb.freq.gp .
$ gnuplot
gnuplot> plot "bands.out.gnu" u 1:2 w l title "pw.x", "band.dat" u 1:2 pt 7 title "EPW"
gnuplot> plot "pb.freq.gp" u 1:2 w l title "matdyn.x", \
              "" u 1:3 w l notitle, \
              "" u 1:4 w l notitle, \
              "freq.dat" u 1:($2*8.06554) pt 7 title "EPW"
```

Note: The multiplication by `8.06554` is due to the different units between `matdyn.x` and `EPW`.

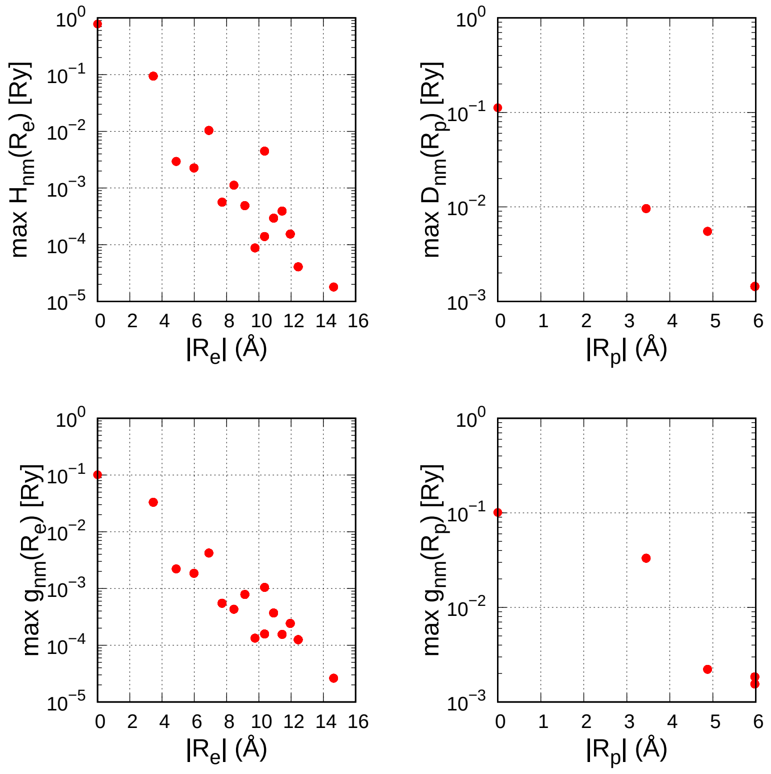


Figure 2

The Wannier-Fourier interpolation technique is based on the decay properties of the Wannier functions and of the phonon perturbation in real space. To check how each quantity is decaying within the supercell corresponding to our initial $\mathbf{k}$ and $\mathbf{q}$ grids, we can plot the files of `decay.H` (Hamiltonian), `decay.dynmat` (dynamical matrix) and `decay.epmate (decay.epmatp)` (electron-phonon matrix elements). You should obtain the plots in Figure 2 (note the log scale on the y axis):
(assume you connected with X11 support on Frontera and you are in the epw folder)

```
$ gnuplot
gnuplot> set logscale y
gnuplot> set pointsize 2
gnuplot> plot "decay.H" u 1:2 w p pt 7
gnuplot> plot "decay.dynmat" u 1:2 w p pt 7
gnuplot> plot "decay.epmate" u 1:2 w p pt 7
gnuplot> plot "decay.epmatp" u 1:2 w p pt 7
```

From these plots we see clearly that we need a larger supercell (denser $\mathbf{q}$-point grid) in order to interpolate more accurately the phonon properties, whereas the electronic parts decay well (remember that we are using a $6 \times 6 \times 6$ $\mathbf{k}$-point grid and a $3 \times 3 \times 3$ $\mathbf{q}$-point grid).

▶ 6th step: We now calculate the phonon linewidth $\gamma_{\mathbf{q}\nu}$, the imaginary part of the phonon self-energy $\Pi''_{\mathbf{q}\nu}$, and the electron-phonon coupling strength $\lambda_{\mathbf{q}\nu}$ associated with a specific phonon mode ν and wavevector $\mathbf{q}$ within the commonly used double-delta approximation along the high-symmetry lines by setting phonselfen=.true. and delta_approx=.true.:

$$\gamma_{\mathbf{q}\nu} = \Pi''_{\mathbf{q}\nu} = 2\pi\omega_{\mathbf{q}\nu} \sum_{nm} \int_{\mathrm{BZ}} \frac{d\mathbf{k}}{\Omega_{\mathrm{BZ}}} |g_{mn,\nu}(\mathbf{k},\mathbf{q})|^2 \delta(\varepsilon_{n\mathbf{k}} - \varepsilon_{\mathrm{F}})\delta(\varepsilon_{m\mathbf{k}+\mathbf{q}} - \varepsilon_{\mathrm{F}}), \tag{1}$$

where ε_{F} is the Fermi level.

$$\lambda_{\mathbf{q}\nu} = \frac{1}{N(\varepsilon_{\mathrm{F}})\omega_{\mathbf{q}\nu}} \sum_{nm} \int_{\mathrm{BZ}} \frac{d\mathbf{k}}{\Omega_{\mathrm{BZ}}} |g_{mn,\nu}(\mathbf{k},\mathbf{q})|^2 \delta(\varepsilon_{n\mathbf{k}} - \varepsilon_{\mathrm{F}})\delta(\varepsilon_{m\mathbf{k}+\mathbf{q}} - \varepsilon_{\mathrm{F}}) = \frac{\gamma_{\mathbf{q}\nu}}{\pi N(\varepsilon_{\mathrm{F}})\omega^2_{\mathbf{q}\nu}}, \tag{2}$$

where $N(\varepsilon_{\mathrm{F}})$ is the density of states per spin at the Fermi level.
In this run, we use the restart feature by reading all quantities written during the previous EPW run (epwread = .true.):

```
$ sbatch job.epw2
```

```
                                                                    job.epw2
#!/bin/bash
#SBATCH -J myjob           # Job name
#SBATCH -p small     # Queue (partition) name
#SBATCH -N 1               # Total # of nodes
#SBATCH --ntasks-per-node 8
#SBATCH -t 01:00:00        # Run time (hh:mm:ss)
#SBATCH -A EPW-SCHOOL
#SBATCH --reservation=EPW-SCHOOL-06-15-2021

module list
pwd
date

# Launch MPI code...
export PATHQE=/work2/06868/giustino/EPW-SCHOOL/q-e

ibrun $PATHQE/bin/epw.x -nk 8 -in pb.epw2.in > pb.epw2.out

date
```

```
--                                                                pb.epw2.in
&inputepw
  prefix      = 'pb',
  amass(1)    = 207.2
  outdir      = './'
  dvscf_dir   = '../phonon/save'

  elph        = .true.
  epwwrite    = .false.
  epwread     = .true.

  wannierize  = .false.
  nbndsub     =  4
  bands_skipped = 'exclude_bands = 1-5'
  num_iter    = 300
  dis_win_max = 21
  dis_froz_max= 13.5
  proj(1)     = 'Pb:sp3'
  phonselfen  = .true.
  delta_approx= .true.

  fsthick     = 1 ! eV
  temps       = 0.075 ! K
  degaussw    = 0.2 ! eV

  filqf       = 'path2.dat'
  nkf1         = 20
  nkf2         = 20
  nkf3         = 20

  nk1         = 6
  nk2         = 6
  nk3         = 6
  nq1         = 3
  nq2         = 3
  nq3         = 3
 /
```

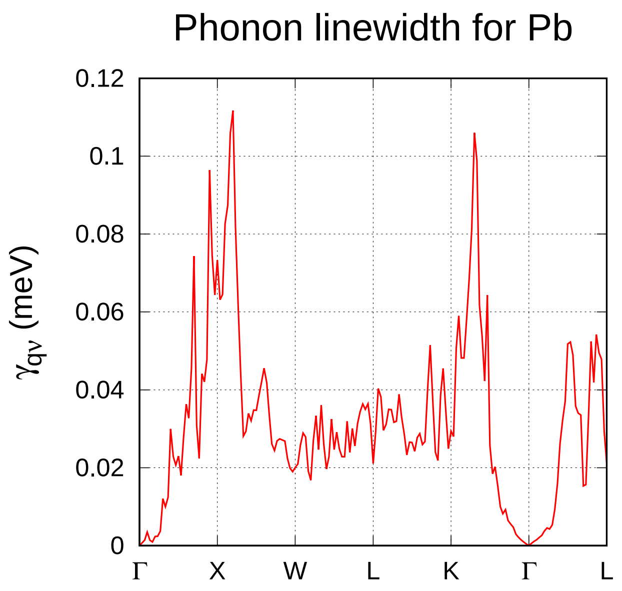
Note 1: The parameter fsthick determines the energy window around the Fermi level for which the electron-phonon matrix elements are interpolated. This can reduce significantly the cost of calculations: for example, only electronic states within small phonon energy from the Fermi level will contribute to the phonon linewidth, therefore we can use fsthick $= 1$ (eV).

Note 2: Since calculating $\gamma_{\mathbf{q}\nu}$ and $\lambda_{\mathbf{q}\nu}$ requires an integration over $\mathbf{k}$, we interpolate the electron-phonon matrix elements onto a denser $20 \times 20 \times 20$ $\mathbf{k}$-point grid.

You can see that now the phonon linewidths and electron-phonon coupling strengths are printed in the output file of `pb.epw2.out` for each phonon wavevector $\mathbf{q}$ and mode ν (`lambda___` and `gamma___`). The sum of $\lambda_{\mathbf{q}\nu}$ over all phonon modes, $\lambda_{\mathbf{q}}$, is also written (`lambda___( tot )`). $\gamma_{\mathbf{q}\nu}$ and $\lambda_{\mathbf{q}\nu}$ are also stored in the files of `linewidth.phself.XXXK` and `lambda.phself.XXXK`, respectively. Inspect those files to familiarize yourself with the format and learn how to plot these quantities. For example, to plot the $\mathbf{q}$-dependent linewidth of the third phonon, you can type in `gnuplot`:

```
$ gnuplot
gnuplot> plot "linewidth.phself.0.075K" u 1:4 every 3::2 w l lw 2
```

and you should be able to produce a plot like this:



Exercise 2

In this exercise we will examine the electron-phonon interaction in semiconducting 3C-SiC. 3C-SiC is a polar material in which electrons can interact strongly with longitudinal optical modes, resulting in $1/|\mathbf{q}|$ divergence of the electron-phonon matrix elements for $|\mathbf{q}| \to 0$. Here we start by checking the quality of Wannier interpolation as we did in Exercise 1; then we will see how to correctly interpolate the electron-phonon matrix elements in this case and calculate the electron linewidth.

► 1st step: Run a self-consistent calculation on a homogeneous $8 \times 8 \times 8$ $\mathbf{k}$-point grid and a phonon calculation on a homogeneous $3 \times 3 \times 3$ $\mathbf{q}$-point grid using the following inputs and jobscript:

```
$ cd ../../exercise2
$ cd phonon
$ sbatch job.ph
```

job.ph

```
#!/bin/bash
#SBATCH -J myjob           # Job name
#SBATCH -p small    # Queue (partition) name
#SBATCH -N 1               # Total # of nodes
#SBATCH --ntasks-per-node 8
#SBATCH -t 01:00:00        # Run time (hh:mm:ss)
#SBATCH -A EPW-SCHOOL
#SBATCH --reservation=EPW-SCHOOL-06-15-2021

module list
pwd
date

# Launch MPI code...
export PATHQE=/work2/06868/giustino/EPW-SCHOOL/q-e

ibrun $PATHQE/bin/pw.x -nk 4 -in sic.scf.in > sic.scf.out
ibrun $PATHQE/bin/ph.x -nk 4 -in sic.ph.in > sic.ph.out

date
```

sic.scf.in

```
--
 &control
    calculation     = 'scf'
    prefix          = 'sic'
    restart_mode    = 'from_scratch'
    pseudo_dir      = '../'
    outdir          = './'
 /
 &system
    ibrav           = 2
    celldm(1)       = 8.237
    nat             = 2
    ntyp            = 2
    ecutwfc         = 30.0
 /
 &electrons
    diagonalization = 'david'
    mixing_beta     = 0.7
    conv_thr        = 1.0d-10
 /
ATOMIC_SPECIES
  Si   28.0855     Si.pz-vbc.UPF
  C   12.01078    C.UPF
ATOMIC_POSITIONS alat
  Si  0.00  0.00  0.00
  C   0.25  0.25  0.25
K_POINTS automatic
8 8 8 0 0 0
```

```
--                                                                    sic.ph.in
&inputph
  prefix   = 'sic'
  fildvscf = 'dvscf'
  ldisp    = .true
  fildyn   = 'sic.dyn'
  nq1=3,
  nq2=3,
  nq3=3,
  tr2_ph   =  1.0d-12
 /
```

Note that the output of `sic.ph.out` now contains the Born effective charges Z^* and the electronic dielectric constant ϵ^{∞}:

```
     Electric Fields Calculation
     ...
     End of electric fields calculation

           Dielectric constant in cartesian axis

           (       7.214254706      -0.000000000       0.000000000 )
           (      -0.000000000       7.214254706       0.000000000 )
           (       0.000000000       0.000000000       7.214254706 )

           Effective charges (d Force / dE) in cartesian axis

            atom      1   Si
      Ex  (        2.67034       -0.00000       -0.00000 )
      Ey  (       -0.00000        2.67034       -0.00000 )
      Ez  (       -0.00000        0.00000        2.67034 )
            atom      2   C
      Ex  (       -2.69642       -0.00000       -0.00000 )
      Ey  (       -0.00000       -2.69642       -0.00000 )
      Ez  (        0.00000       -0.00000       -2.69642 )
```

These quantities are automatically calculated for an insulating system, using the response to a finite electric field, and determine the frequency splitting between LO and TO phonon modes.

▶ 2nd step: Calculate the phonon dispersion by using Fourier interpolation. This can be done in the following two steps:
(1) Bring the interatomic force constant to real space using a Fourier transformation with the program of `q2r.x` program.

```
$ /work2/06868/giustino/EPW-SCHOOL/q-e/bin/q2r.x -in q2r.in > q2r.out
```

```
                                                              q2r.in
&input
  zasr='simple', fildyn='sic.dyn', flfrc='sic333.fc'
/
```

Note in the output if the Fast Fourier transform (FFT) was successful:

```
      fft-check success (sum of imaginary terms < 10^-12)
```

The matrix of interatomic force constants in real space should have been created (file `sic333.fc`).

(2) Calculate the phonon dispersion along the high-symmetry lines using the program of `matdyn.x`.

```
$ /work2/06868/giustino/EPW-SCHOOL/q-e/bin/matdyn.x -in matdyn.in > matdyn.out
```

```
                                                           matdyn.in
&input
  asr              = 'simple',
  flfrc            = 'sic333.fc'
  flfrq            = 'sic.freq'
  q_in_band_form   = .true.
  q_in_cryst_coord = .true.
/
  7
  0.000 0.000 0.000  30
  0.500 0.000 0.500  30
  0.500 0.250 0.750  30
  0.500 0.500 0.500  30
  0.375 0.375 0.750  30
  0.000 0.000 0.000  30
  0.500 0.500 0.500  30
```

This produces the file named `sic.freq.gp` with the phonon frequencies along the path, expressed in cm^{-1} unit. This will be checked against the phonon frequencies along the same path from `EPW`.

Note that now there are 3 acoustic and 3 optical branches, separated by a gap, and that there is a LO-TO splitting near Γ.

► 3rd step: Gather the `.dyn`, `.dvscf` and `patterns` files into a `save/` directory using the `pp.py` script:

```
$ python3 /work2/06868/giustino/EPW-SCHOOL/q-e/EPW/bin/pp.py
```

The script will ask you to enter the `prefix` of your calculation (sic):

```
Enter the prefix used for PH calculations (e.g. diam)
sic
```

► 4th step: Run a calculation for the direct evaluation of electron-phonon matrix elements along the selected high-symmetry lines using density-functional perturbation theory. For this purpose, we are going to use the modified version of `ph.x` located in the directory of `$PATHQE/test-suite/not_epw_comp/`:

```
$ cd ../ephline
$ sbatch job.ph
```

```
                                                                                          job.ph
#!/bin/bash
#SBATCH -J myjob           # Job name
#SBATCH -p small    # Queue (partition) name
#SBATCH -N 1               # Total # of nodes
#SBATCH --ntasks-per-node 36
#SBATCH -t 01:00:00        # Run time (hh:mm:ss)
#SBATCH -A EPW-SCHOOL
#SBATCH --reservation=EPW-SCHOOL-06-15-2021

module list
pwd
date

# Launch MPI code...
export PATHQE=/work2/06868/giustino/EPW-SCHOOL/q-e

ibrun $PATHQE/bin/pw.x -nk 18 -in sic.scf.in > sic.scf.out
ibrun $PATHQE/test-suite/not_epw_comp/ph.x -nk 18 -in sic.ephline.in > sic.ephline.out

date
```

```
--                                                                                 sic.ephline.in
&inputph
  prefix   = 'sic'
  fildvscf = 'dvscf'
  ldisp    = .true.
  fildyn   = 'sic.dyn'
  tr2_ph   =  1.0d-12
  qplot    = .true.
  q_in_band_form = .true.
  electron_phonon = 'epw'
!  kx = 1.2500000
!  ky = 0.8750000
!  kz = 1.2500000
/
3
  1.0000 0.0000 0.0000 20 # X
  0.0000 0.0000 0.0000 20 # Gamma
 -0.5000 0.5000 0.5000 1  # L
```

Note: if you use your local clusters, compile this modified version of `ph.x` after compiling `Quantum ESPRESSO`:

```
$ cd YOUR-QE-PATH/test-suite/not_epw_comp ; make
```

The electron-phonon matrix elements for $\mathbf{k} = \Gamma$ (in default) and each $\mathbf{q}$ along the BZ path are written in the output of `sic.ephline.out`, with columns corresponding to each band and phonon mode. Since electron-phonon matrix elements are gauge-dependent, an (gauge-invariant) average of their norm over degenerate subspaces of bands and modes has been performed. If you want to change the wavevector $\mathbf{k}$ of the initial state, you can do it by specifying the input flags of `kx, ky, kz` (in Cartesian $2\pi/a$ units).

To extract and plot the electron-phonon matrix elements for the valence-band top $(n = m = 4)$ and the highest-frequency phonon branch $(\nu = 6)$, for example, you can type with the eight spaces between digits:

```
$ grep "4        4        6" sic.ephline.out > data
$
```

This data will be processed together with the corresponding data from EPW for comparison.

▶ 5th step: Run a non self-consistent calculation for electronic band structures using the charge density and other outputs from the previous self-consistent run:

```
$ cd ../band
$ sbatch job.band
```

job.band

```
#!/bin/bash
#SBATCH -J myjob           # Job name
#SBATCH -p small    # Queue (partition) name
#SBATCH -N 1               # Total # of nodes
#SBATCH --ntasks-per-node 8
#SBATCH -t 01:00:00        # Run time (hh:mm:ss)
#SBATCH -A EPW-SCHOOL
#SBATCH --reservation=EPW-SCHOOL-06-15-2021

module list
pwd
date

# Launch MPI code...
export PATHQE=/work2/06868/giustino/EPW-SCHOOL/q-e

mkdir sic.save
cp ../phonon/sic.save/charge-density.dat sic.save/
cp ../phonon/sic.save/data-file-schema.xml sic.save/

ibrun $PATHQE/bin/pw.x -nk 4 -in sic.band.in > sic.band.out

date
```

sic.band.in

```
--
&control
    calculation     = 'bands'
    prefix          = 'sic'
    pseudo_dir      = '../'
    outdir          = './'
 /
 &system
    ibrav           = 2
    celldm(1)       = 8.237
    nat             = 2
    ntyp            = 2
    ecutwfc         = 30.0
    nbnd            = 4
 /
 &electrons
    diagonalization = 'david'
    mixing_beta     = 0.7
    conv_thr        = 1.0d-10
 /
ATOMIC_SPECIES
  Si   28.0855     Si.pz-vbc.UPF
  C    12.01078    C.UPF
ATOMIC_POSITIONS alat
  Si   0.00   0.00   0.00
  C    0.25   0.25   0.25
K_POINTS crystal_b
```

```
 7
 0.000 0.000 0.000  30
 0.500 0.000 0.500  30
 0.500 0.250 0.750  30
 0.500 0.500 0.500  30
 0.375 0.375 0.750  30
 0.000 0.000 0.000  30
 0.500 0.500 0.500  30
```

Then, run the program of `bands.x` to obtain the band structure data:

```
$ /work2/06868/giustino/EPW-SCHOOL/q-e/bin/bands.x -in sic.bands.in > sic.bands.out
```

This will produce the file of bands.out.gnu which will be compared with the interpolated electronic band structure from `EPW`.

▶ 6th step: Run a non self-consistent calculation on a homogeneous $6 \times 6 \times 6$ $\mathbf{k}$-point grid for generating the input of wave functions for `EPW` and an `EPW` calculation for the Wannier interpolation of electron-phonon matrix elements along the high-symmetry lines:

Note: The positive-definite homogeneous $6 \times 6 \times 6$ $\mathbf{k}$-point grid between 0 and 1 in `sic.nscf.in` can be generated by using the script of `kmesh.pl` included in the Wannier90 package:

```
$ /work2/06868/giustino/EPW-SCHOOL/q-e/wannier90-3.1.0/utility/kmesh.pl 6 6 6
```

```
$ cd ../epw
$ sbatch job.epw1
```

job.epw1

```
#!/bin/bash
#SBATCH -J myjob           # Job name
#SBATCH -p small    # Queue (partition) name
#SBATCH -N 1               # Total # of nodes
#SBATCH --ntasks-per-node 8
#SBATCH -t 01:00:00        # Run time (hh:mm:ss)
#SBATCH -A EPW-SCHOOL
#SBATCH --reservation=EPW-SCHOOL-06-15-2021

module list
pwd
date

# Launch MPI code...
export PATHQE=/work2/06868/giustino/EPW-SCHOOL/q-e

mkdir sic.save
cp ../phonon/sic.save/charge-density.dat sic.save/
cp ../phonon/sic.save/data-file-schema.xml sic.save/

ibrun $PATHQE/bin/pw.x -nk 4 -in sic.nscf.in > sic.nscf.out
ibrun $PATHQE/bin/epw.x -nk 8 -in sic.epw1.in > sic.epw1.out

date
```

sic.nscf.in

```
--
 &control
    calculation     = 'nscf'
    prefix          = 'sic'
    pseudo_dir      = '../'
    outdir          = './'
 /
 &system
    ibrav           = 2
    celldm(1)       = 8.237
    nat             = 2
    ntyp            = 2
    ecutwfc         = 30.0
    nbnd            = 4
 /
 &electrons
    diagonalization = 'david'
    mixing_beta     = 0.7
    conv_thr        = 1.0d-10
 /
ATOMIC_SPECIES
  Si  28.0855     Si.pz-vbc.UPF
  C   12.01078    C.UPF
ATOMIC_POSITIONS alat
  Si  0.00  0.00  0.00
  C   0.25  0.25  0.25
K_POINTS crystal
216
  0.00000000  0.00000000  0.00000000  4.629630e-03
  0.00000000  0.00000000  0.16666667  4.629630e-03
  0.00000000  0.00000000  0.33333333  4.629630e-03
...
```

```
--                                                                        sic.epw1.in
&inputepw
  prefix      = 'sic'
  amass(1)    = 28.0855
  amass(2)    = 12.0107
  outdir      = './'
  dvscf_dir   = '../phonon/save'

  elph        = .true.
  epwwrite    = .true.
  epwread     = .false.
  lpolar      = .true.

  wannierize  = .true.
  nbndsub     =  4
  num_iter    = 300
  proj(1)     = 'Si:sp3'

  prtgkk      = .true.

  filqf       = 'path1.dat'
  nkf1        = 1
  nkf2        = 1
  nkf3        = 1

  nk1         = 6
  nk2         = 6
  nk3         = 6
  nq1         = 3
  nq2         = 3
  nq3         = 3
 /
```

In this calculation, we consider the BZ path of X-Γ-L. Note also that in order to speed up the calculation, we chose to print the electron-phonon matrix elements for the initial electronic states only at $\mathbf{k} = \Gamma$ (`nkf1=nkf2=nkf3=1`).

Since $3C$-SiC is a polar material (non-zero Born effective charges), we set `lpolar = .true.` in order to correctly treat the long-range interaction in bulk crystals. The strategy consists in subtracting the long-range component $g^{\mathcal{L}}$ from the full electron-phonon matrix element g before interpolation and adding it back after interpolation. An analogous strategy is implemented to correctly interpolate the dynamical matrix including the long-range dipole-dipole interaction which results in the LO-TO splitting.

The electron-phonon matrix elements for $\mathbf{k} = \Gamma$ and each $\mathbf{q}$ along the BZ path are written in the output of `sic.epw1.out`, with columns corresponding to each band and phonon mode. Since electron-phonon matrix elements are gauge-dependent, an (gauge-invariant) average of their norm over degenerate subspaces of bands and modes has been performed.

To extract and plot the electron-phonon matrix elements for the valence-band top $(n = m = 4)$ and the highest-frequency phonon branch $(\nu = 6)$ same as in the 4th step, you can type with the eight spaces between digits:

```
$ grep "4        4        6" sic.epw1.out > data.epw
```

The plot should look like the one in Figure 3:
(assume you connected with X11 support on Frontera and you are in the epw folder)

```
$ cp ../ephline/data .
$ gnuplot
gnuplot> set pointsize 2
gnuplot> plot "data" u 7 pt 6 lc rgb "black" title "ph.x", \
              "data.epw" u 7 pt 7 lc rgb "red" title "EPW"
```

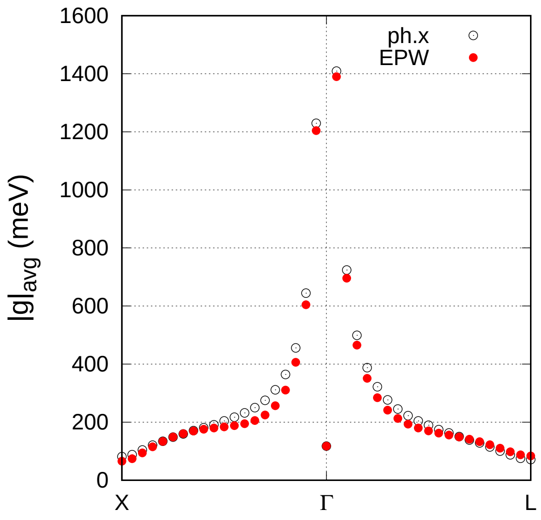


Figure 3

Note that the interpolated ones are not fully accurate, since a larger coarse $\mathbf{q}$-grid is needed in order to accurately interpolate the electron-phonon matrix elements.

Note: if you want to inspect the results of the interpolation when the long-range contribution to the electron-phonon matrix elements is not taken into account, you can run a new calculation using `lpolar=.false.`. Remember to run this test in a new directory, or to run again an EPW calculation from scratch using `lpolar=.false.`.

▶ 7th step: Run an EPW calculation for the Wannier interpolation of electronic band structure and phonon dispersion along the high-symmetry lines. In this run, we use the restart feature by reading all quantities written during the previous EPW run (`epwread = .true.`):

```
$ sbatch job.epw2
```

job.epw2

```
#!/bin/bash
#SBATCH -J myjob           # Job name
#SBATCH -p small     # Queue (partition) name
#SBATCH -N 1               # Total # of nodes
#SBATCH --ntasks-per-node 8
#SBATCH -t 01:00:00        # Run time (hh:mm:ss)
#SBATCH -A EPW-SCHOOL
#SBATCH --reservation=EPW-SCHOOL-06-15-2021

module list
pwd
date

# Launch MPI code...
export PATHQE=/work2/06868/giustino/EPW-SCHOOL/q-e

ibrun $PATHQE/bin/epw.x -nk 8 -in sic.epw2.in > sic.epw2.out

date
```

sic.epw2.in

```
--
&inputepw
  prefix      = 'sic'
  amass(1)    = 28.0855
  amass(2)    = 12.0107
  outdir      = './'
  dvscf_dir   = '../phonon/save'

  elph        = .true.
  epwwrite    = .false.
  epwread     = .true.
  lpolar      = .true.

  wannierize  = .false.
  nbndsub     =  4
  num_iter    = 300
  proj(1)     = 'Si:sp3'

  band_plot   = .true.

  filqf       = 'path2.dat'
  filkf       = 'path2.dat'

  nk1         = 6
  nk2         = 6
  nk3         = 6
  nq1         = 3
  nq2         = 3
```

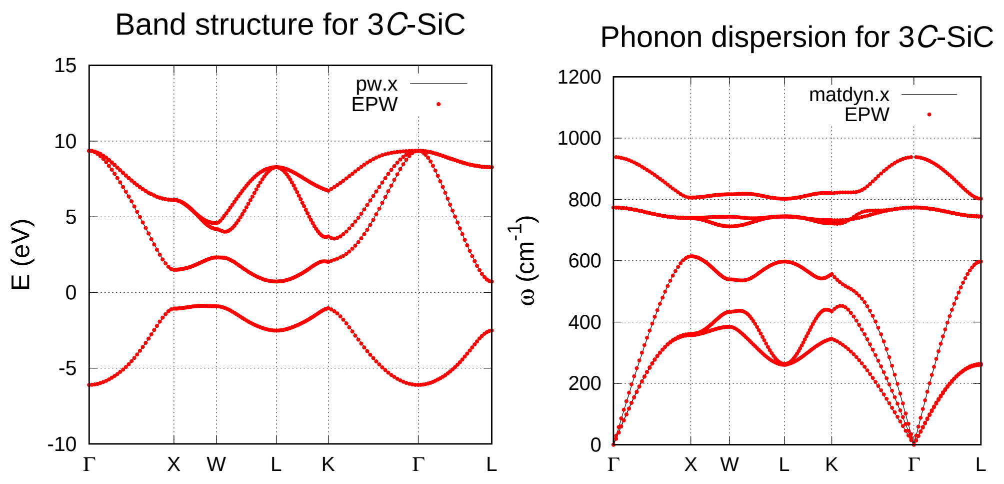


Figure 4

```
 nq3         = 3
/
```

To extract easily the data to plot, you can simply run the program of `plotband.x` in the `Quantum ESPRESSO` package and enter the input file (`band.eig` or `phband.freq`), the energy range (for example, -7,10), the output file with the data to plot (`band.dat` or `freq.dat`); other inputs are not relevant and simply push the ENTER key when asked:

```
$ /work2/06868/giustino/EPW-SCHOOL/q-e/bin/plotband.x
     Input file > band.eig
Reading    4 bands at     181 k-points
Range:   -6.1066    9.3595eV  Emin, Emax, [firstk, lastk] > -7,10
high-symmetry point:  0.0000 0.0000 0.0000   x coordinate   0.0000
high-symmetry point: -1.0000 0.0000 0.0000   x coordinate   1.0000
high-symmetry point: -1.0000 0.5000 0.0000   x coordinate   1.5000
high-symmetry point: -0.5000 0.5000 0.5000   x coordinate   2.2071
high-symmetry point: -0.7500 0.7500 0.0000   x coordinate   2.8195
high-symmetry point:  0.0000 0.0000 0.0000   x coordinate   3.8801
high-symmetry point: -0.5000 0.5000 0.5000   x coordinate   4.7462
output file (gnuplot/xmgr) > band.dat
bands in gnuplot/xmgr format written to file band.dat
output file (ps) >
stopping ...

$ /work2/06868/giustino/EPW-SCHOOL/q-e/bin/plotband.x
     Input file > phband.freq
Reading    6 bands at     181 k-points
Range:    0.0000  116.3078eV  Emin, Emax, [firstk, lastk] > 0,117
high-symmetry point:  0.0000 0.0000 0.0000   x coordinate   0.0000
high-symmetry point: -1.0000 0.0000 0.0000   x coordinate   1.0000
high-symmetry point: -1.0000 0.5000 0.0000   x coordinate   1.5000
high-symmetry point: -0.5000 0.5000 0.5000   x coordinate   2.2071
high-symmetry point: -0.7500 0.7500 0.0000   x coordinate   2.8195
high-symmetry point:  0.0000 0.0000 0.0000   x coordinate   3.8801
```

```
high-symmetry point: -0.5000 0.5000 0.5000   x coordinate   4.7462
output file (gnuplot/xmgr) > freq.dat
bands in gnuplot/xmgr format written to file freq.dat
output file (ps) >
stopping ...
```

The plots should look like those in Figure 4:
(assume you connected with X11 support on Frontera and you are in the epw folder)

```
$ cp ../band/bands.out.gnu .
$ cp ../phonon/sic.freq.gp .
$ gnuplot
gnuplot> plot "bands.out.gnu" u 1:2 w l title "pw.x", "band.dat" u 1:2 pt 7 title "EPW"
gnuplot> plot "sic.freq.gp" u 1:2 w l title "matdyn.x", \
"" u 1:3 w l notitle, \
"" u 1:4 w l notitle, \
"" u 1:5 w l notitle, \
"" u 1:6 w l notitle, \
"freq.dat" u 1:($2*8.06554) pt 7 title "EPW"
```

Note: The multiplication by `8.06554` is due to the different units between `matdyn.x` and `EPW`.

► 8th step: We now calculate the linewidths of the valence states in SiC along high-symmetry lines, which correspond to twice the imaginary part of the electron self-energy $\Sigma''_{n\mathbf{k}}$:

$$\begin{aligned}\Sigma''_{n\mathbf{k}}(\omega,T) =& \pi\sum_{m\nu}\int_{\mathrm{BZ}}\frac{d\mathbf{q}}{\Omega_{\mathrm{BZ}}}|g_{mn,\nu}(\mathbf{k},\mathbf{q})|^2\{[n_{\mathbf{q}\nu}(T)+f_{m\mathbf{k}+\mathbf{q}}(T)]\delta(\omega-(\varepsilon_{m\mathbf{k}+\mathbf{q}}-\varepsilon_{\mathrm{F}})+\omega_{\mathbf{q}\nu}) \\ &+[n_{\mathbf{q}\nu}(T)+1-f_{m\mathbf{k}+\mathbf{q}}(T)]\delta(\omega-(\varepsilon_{m\mathbf{k}+\mathbf{q}}-\varepsilon_{\mathrm{F}})-\omega_{\mathbf{q}\nu})\},\end{aligned} \tag{3}$$

where $n_{\mathbf{q}\nu}(T)$ is the Bose-Einstein distribution and $f_{n\mathbf{k}}(T)$ is the electronic occupation. To calculate the electron linewidth, we need to use the input keyword of `elecselfen = .true.`, the $\mathbf{k}$-point path, and a homogeneous $\mathbf{q}$-point grid for the integration:

```
$ sbatch job.epw3
```

```
                                                                    job.epw3
#!/bin/bash
#SBATCH -J myjob           # Job name
#SBATCH -p small    # Queue (partition) name
#SBATCH -N 1               # Total # of nodes
#SBATCH --ntasks-per-node 8
#SBATCH -t 01:00:00        # Run time (hh:mm:ss)
#SBATCH -A EPW-SCHOOL
#SBATCH --reservation=EPW-SCHOOL-06-15-2021

module list
pwd
date

# Launch MPI code...
export PATHQE=/work2/06868/giustino/EPW-SCHOOL/q-e

ibrun $PATHQE/bin/epw.x -nk 8 -in sic.epw3.in > sic.epw3.out

date
```

```
--                                                                                      sic.epw3.in
&inputepw
  prefix      = 'sic'
  amass(1)    = 28.0855
  amass(2)    = 12.0107
  outdir      = './'
  dvscf_dir   = '../phonon/save'

  elph        = .true.
  epwwrite    = .false.
  epwread     = .true.
  lpolar      = .true.

  wannierize  = .false.
  nbndsub     =  4
  num_iter    = 300
  proj(1)     = 'Si:sp3'

  elecselfen  = .true.
  efermi_read = .true.
  fermi_energy= 9.6

  fsthick     = 7.0
  temps       = 20
  degaussw    = 0.05

  filkf       = 'path2.dat'
  nqf1        = 20
  nqf2        = 20
  nqf3        = 20

  nk1         = 6
  nk2         = 6
  nk3         = 6
  nq1         = 3
  nq2         = 3
  nq3         = 3
 /
```

Note that since we use a $\mathbf{k}$-point path, the Fermi energy calculated from the fine grid will not be accurate. To overcome this problem, we provide the Fermi energy in the input by using the input keywords of `efermi_read = .true.` and `fermi_energy=9.6` (just above the valence-band top).

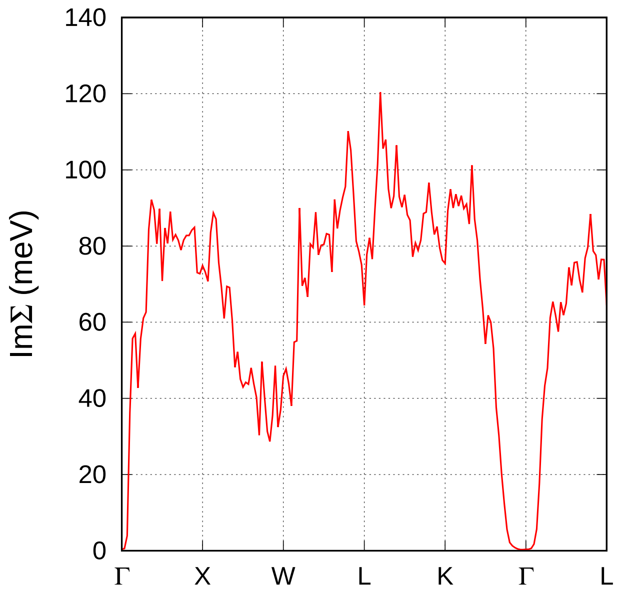
In the output you can monitor the progression of the $\mathbf{q}$ integration, before reading the electron self-energy for each $\mathbf{k}$ point:

```
Progression iq (fine) =        100/      8000
Progression iq (fine) =        200/      8000
Progression iq (fine) =        300/      8000
...
...

ik =       1 coord.:    0.0000000    0.0000000    0.0000000
-------------------------------------------------------------------
E(   2 )=  -0.2405 eV   Re[Sigma]=      95.950728 meV  Im[Sigma]=       0.392034 meV
E(   3 )=  -0.2405 eV   Re[Sigma]=      95.950728 meV  Im[Sigma]=       0.392034 meV
E(   4 )=  -0.2405 eV   Re[Sigma]=      95.950728 meV  Im[Sigma]=       0.392034 meV
```

```
    -----------------------------------------------------------------------
    ...
```

Note also that the electron energies are now reported with respect to E_F. Moreover, the self-energy for the first valence band is not computed since `fsthick` is 7 eV.
To plot the electron linewidths you can use the file of `linewidth.elself.20.000K` that has been created. For example, ImΣ for the highest valence band should look like:

```
$ gnuplot
gnuplot> plot "linewidth.elself.20.000K" u 1:4 every 3::2 w lp
```



If you want to plot the electron lifetimes, these are given by $\tau_{n\mathbf{k}} = \hbar/(2|\mathrm{Im}\Sigma_{n\mathbf{k}}|)$.

Problem

Repeat the 4th and 6th steps of Exercise 2 with the wavevector $\mathbf{k}$ of the initial state set to `(-0.5,-1.0,0.0)` (in Cartesian $2\pi/a$ units).
Hint: You can proceed in the following way:

```
* Go to the problem folder under the Tue.5.Lee folder.
$ cd ephline
* Modify the following lines in sic.ephline.in:
!  kx = 1.2500000
!  ky = 0.8750000
!  kz = 1.2500000
$ sbatch job.ph
$ cd ../epw/
* Modify the following lines in sic.epw1.in:
  nkf1        = 1
  nkf2        = 1
  nkf3        = 1
* And create a necessary file.
$ sbatch job.epw1
```

Note: The run in the epw folder uses the save directory generated in the 3rd step of Exercise 2 and answers to this problem are in ephline/answer and epw/answer directories.