ICTP/Psi-k/CECAM School on Electron-Phonon Physics from First Principles

Electron-phonon coupling with EPW

Hands-on Session (Wed.4)

In this session we will learn to use the core capabilities of `EPW`.
First copy the tutorial input files and go in the first exercise:

```
$ wget http://epw.org.uk/uploads/School2018/Wed.4.Verdi.tar
$ tar -xvf Wed.4.Verdi.tar ; cd tuto_Wed4/exercise1
```

Exercise 1

In this exercise we will repeat the calculation of the electron-phonon coupling strength and of the Eliashberg spectral function of **lead** that you performed yesterday using `Quantum Espresso` and the `PHonon` code. This time we will use `EPW`, which will allow us to calculate these quantities using much denser grids at small additional cost.

▶ Run a self-consistent calculation for lead as in Exercise 1 of tutorial Tue.5.
Note: The energy cutoff ecutwfc needed for convergence should be 90 Ry.

```
$ mkdir exercise1; cd exercise1; cp ../../tuto_Tues5/exercise1/pb.scf.in .
$ cp ../../tuto_Tues5/exercise1/pb_s.UPF .
$ mpirun -np 4 /home/nfs3/smr3191/q-e/bin/pw.x -npool 4 < pb.scf.in > pb.scf.out
```

▶ Run a phonon calculation on a homogeneous $3 \times 3 \times 3$ $\mathbf{q}$-point grid using the following input:

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

```
$ mpirun -np 4 /home/nfs3/smr3191/q-e/bin/ph.x -npool 4 < pb.ph.in > pb.ph.out
```

The keyword fildvscf 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, locate the list of 4 irreducible $\mathbf{q}$ points in the Brillouin Zone (IBZ):

```
 Dynamical matrices for ( 3, 3, 3)  uniform grid of q-points
(   4q-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 lead.dyn0 file. If you type `ls`, you can see a lead.dynX file containing the dynamical matrix has been produced for each irreducible $\mathbf{q}$ point. The dvscf files are all named `lead.dvscf1` and are located inside the `_ph0/lead.q_X/` folders, except for the one corresponding to the first $\mathbf{q}$ point (Γ) that is located in `_ph0/`.

▶ Gather the `.dyn` and `.dvscf` files into a new `save/` directory which `EPW` will read. The files in `_ph0/lead.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:

```
$ python /home/nfs3/smr3191/q-e/EPW/bin/pp.py
```

The script will ask you to prompt the `prefix` of your calculation (lead).

Note: if this doesn't work, you need to install `numpy` on your desktop. Just type "`pip install numpy`" and run the script again.

▶ Run a non self-consistent calculation on a homogeneous $6 \times 6 \times 6$ $\mathbf{k}$ grid. You cannot use the `K_POINTS automatic` feature of `pw.x`, but you need to generate a positive definite grid between 0 and 1 and specify `K_POINTS crystal`. To do so, first copy the input file for the self-consistent calculation into a new one:

```
$ cp pb.scf.in pb.nscf.in
```

then using `vi` or another text editor modify the input variable `calculation` to `nscf`, remove `wf_collect=.true.`, set the number of bands to `nbnd=10`, and delete the last two lines specifying the $\mathbf{k}$-point grid. Then you can use, for example, a simple script provided in the `wannier90` distribution to generate the homogeneous $\mathbf{k}$ grid, `kmesh.pl`:

```
$ /home/nfs3/smr3191/q-e/wannier90-2.1.0/utility/kmesh.pl 6 6 6 >> pb.nscf.in
```

Now your input file should look like this:

```
&control                                                        pb.nscf.in
    calculation='nscf'
    restart_mode='from_scratch',
    prefix='lead',
    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
...
```

Note: in general you should add `nosym=.true.` in the `&system` namelist to make sure `pw.x` calculates each $\mathbf{k}$ point provided in the list, without using any symmetry operations. 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 $6 \times 6 \times 6$ $\mathbf{k}$ grid, the $3 \times 3 \times 3$ grid used in the phonon calculation is appropriate, however a $6 \times 6 \times 6$ $\mathbf{q}$ grid would be needed in order to interpolate more accurately the dynamical matrix and the electron-phonon matrix elements.

Now you can run:

```
$ mpirun -np 4 /home/nfs3/smr3191/q-e/bin/pw.x -npool 4 < pb.nscf.in > pb.nscf.out
```

Since `EPW` does not yet support G-vector parallelization, we use $\mathbf{k}$-point parallelization, which means that `np` needs to be always equal to `npool`. If you forget this the code will crash, asking to use only one processor per pool.

▶ We are now ready to perform a calculation using `EPW`. Prepare the following input file and run `EPW`:

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

  elph        = .true.
  kmaps       = .false.
  epbwrite    = .true.
  epbread     = .false.
  epwwrite    = .true.
  epwread     = .false.

  wannierize  = .true.
  nbndsub     =  4
  nbndskip    =  5
  num_iter    = 300
  dis_win_max = 21
  dis_win_min = -3
  dis_froz_min= -3
  dis_froz_max= 13.5
  proj(1)     = 'Pb:sp3'
  wdata(1) = 'bands_plot = .true.'
  wdata(2) = 'begin kpoint_path'
  wdata(3) = 'G 0.00 0.00 0.00 X 0.00 0.50 0.50'
  wdata(4) = 'X 0.00 0.50 0.50 W 0.25 0.50 0.75'
  wdata(5) = 'W 0.25 0.50 0.75 L 0.50 0.50 0.50'
  wdata(6) = 'L 0.50 0.50 0.50 G 0.00 0.00 0.00'
  wdata(7) = 'G 0.00 0.00 0.00 K 0.375 0.375 0.75'
  wdata(8) = 'end kpoint_path'
  wdata(9) = 'bands_num_points = 10'

  elecselfen  = .false.
  phonselfen  = .false.
  a2f          = .false.

  parallel_k   = .true.
  parallel_q   = .false.

  fsthick      = 1 ! eV
  eptemp       = 0.075 ! K
  degaussw     = 0.1 ! eV

  nkf1          = 6
  nkf2          = 6
  nkf3          = 6
  nqf1          = 3
  nqf2          = 3
  nqf3          = 3

  nk1          = 6
  nk2          = 6
  nk3          = 6
  nq1          = 3
  nq2          = 3
  nq3          = 3
 /
  4 cartesian
  0.000000000000000E+00   0.000000000000000E+00   0.000000000000000E+00
 -0.333333333333333E+00   0.333333333333333E+00  -0.333333333333333E+00
  0.000000000000000E+00   0.666666666666667E+00   0.000000000000000E+00
  0.666666666666667E+00  -0.555111512312578E-16   0.666666666666667E+00
```

```
$ mpirun -np 4 /home/nfs3/smr3191/q-e/bin/epw.x -npool 4 < pb.epw.in > pb.epw.out
```

Note: Note: The list of $\mathbf{q}$ points given at the end of the input file should be exactly the same as the list contained in the file `prefix.dyn0`. In `dvscf_dir = './save'` we specify the directory where the `.dyn, .dvscf` and patterns files are stored.

With this input, we are not instructing `EPW` to calculate any physical quantities; instead, `EPW` will perform the following main steps (see the technical lecture Wed.3 summarizing each step):

- Wannierizing the band structure using `wannier90` as an internal library. In the output you can look at the Wannier function centers and spreads obtained:

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

(    0.07779    0.07779    0.07779) :    2.22268
(    0.07779   -0.07779   -0.07779) :    2.22268
(   -0.07779    0.07779   -0.07779) :    2.22268
(   -0.07779   -0.07779    0.07779) :    2.22268
```

  while the full output from the `wannier90` run is in the file `lead.wout`. The input variables 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 nbndskip is the number of valence bands not wannierized (generally a set of bands lying at lower energies, such as semicore states in this example). For the other input variables you can refer to the technical lecture Tue.3, and to the EPW website http://epw.org.uk/Documentation/Inputs.

It is always possible to add extra variables that are read by `wannier90` by using the input wdata(index), with increasing `index` number. Here we use these extra variables in order to plot the interpolated band structure.

- Calculating the electron-phonon matrix elements on the initial $(\mathbf{k}, \mathbf{q})$ grid 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
...
```

- Writing on disk the files `lead.epbX` (one per CPU) containing the Hamiltonian, the dynamical matrices and the electron-phonon matrix elements on the initial (coarse) $\mathbf{k}$ and $\mathbf{q}$ grids in the full Brillouin zone:

```
     Writing epmatq on .epb files
     The .epb files have been correctly written
```

- 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 (`lead.epmatwp1, crystal.fmt, dmedata.fmt, and epwdata.fmt` files):

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

▶ We should always check that the Wannier-interpolated electron and phonon band structure correspond to the ones calculated using `pw.x` and `ph.x`, respectively. To do so, we use the restart feature by reading all quantities written at the end of the previous run (epwread = .true.). Copy the input file into a new one:

```
$ cp pb.epw.in pb.epw.in2
```

and make the following modifications:

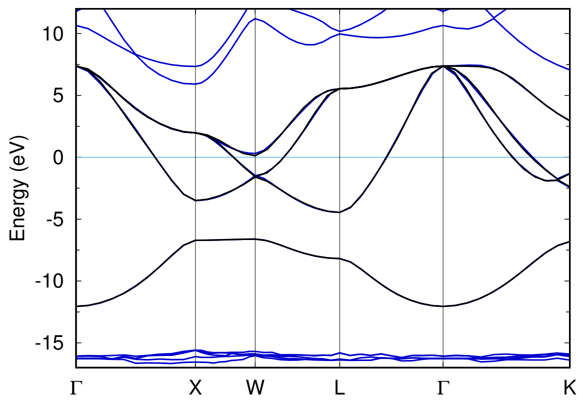
```
 ...                                                            pb.epw.in2
 kmaps       = .true.
 epbwrite    = .false.
 epbread     = .false.
 epwwrite    = .false.
```

```
epwread      = .true.

wannierize   = .false.
...
band_plot    = .true.
...

filqf        = 'lead_band.kpt'
filkf        = 'lead_band.kpt'
!nkf1           = 6
...
!nqf3           = 3
...
```

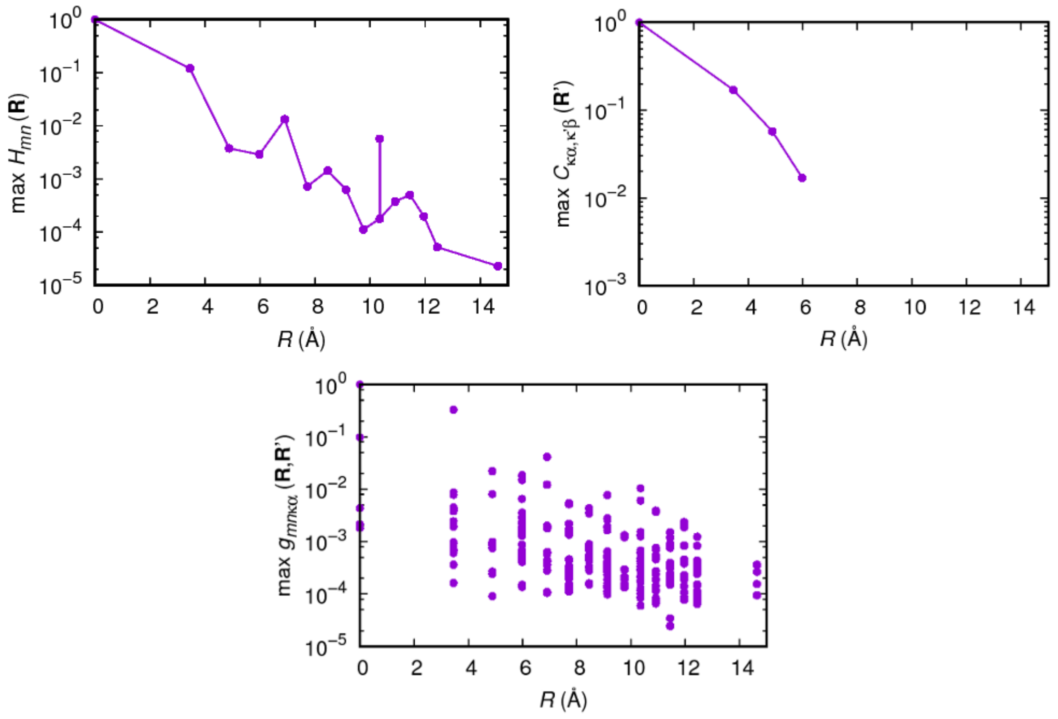
Here we instructed EPW to save on file the interpolated bands (band_plot = .true.), and we commented the input variables nkf1 ... nqf3 defining homogeneous $\mathbf{k}$ and $\mathbf{q}$ grids for the interpolated quantities; instead, we choose to interpolate them onto a Brillouin-zone path defined by filqf, filkf and read from the file lead_band.kpt. This file has been conveniently generated for us by wannier90, and the coordinates are in units of the reciprocal lattice vectors (crystal). EPW always reads coordinates from file in crystal units. If you now run EPW again:

```
$ mpirun -np 4 /home/nfs3/smr3191/q-e/bin/epw.x -npool 4 < pb.epw.in2 > pb.epw.out2
```

the files band.eig and phband.freq are produced. To extract easily the data to plot, you can simply run /home/nfs3/smr3191/q-e/bin/plotband.x and prompt the input file (band.eig or phband.freq), the energy range (-5,20 for example), the output file with the data to plot (band.dat or phband.dat); the other inputs are not important, just put 1 when asked.
You can now compare the phonon dispersions with the results from the tutorial Tue.5 (note: we have a denser $\mathbf{q}$ path here). You can also create a new folder and run a non self-consistent calculation to obtain the electronic band structure with pw.x as you learned in the tutorial Mon.4. If you compare it with the wannierized band structure you should obtain this plot (with the wannierized bands superimposed in black):



▶ 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 decay.H (Hamiltonian), decay.D (dynamical matrix) and decay.epmat_wanep (matrix elements; you can plot the data using the first and last column). You should obtain the plots reported here (note the log scale on the y axis).



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

▶ We now calculate the phonon linewidths $\gamma_{\mathbf{q}\nu}$ in the commonly used double-delta approximation (Eq (1), tutorial Tue.5) along the Brillouin zone, as well as the electron-phonon coupling strength $\lambda_{\mathbf{q}\nu}$ (Eq. (2), tutorial Tue.5) by setting `selfen_phon=.true.` and `delta_approx=.true.`. Since calculating $\gamma_{\mathbf{q}\nu}$ and $\lambda_{\mathbf{q}\nu}$ requires an integration over $\mathbf{k}$, we interpolate the matrix elements onto a denser $20 \times 20 \times 20$ $\mathbf{k}$ grid.

Copy the file `pb.epw.in2`, which we already used to restart an `EPW` calculation, to `pb.epw.in3`, modify it as follows:

```
                                                                    pb.epw.in3
 ...
 phonselfen  = .true.
 a2f         = .false.
 band_plot   = .false.
 delta_approx= .true.

 ...
 filqf       = 'lead_band.kpt'
 !filkf       = 'lead_band.kpt'
 nkf1         = 20
 nkf2         = 20
 nkf3         = 20
 !nqf1         = 3
 !nqf2         = 3
 !nqf3         = 3
 ...
```

and run `EPW`:

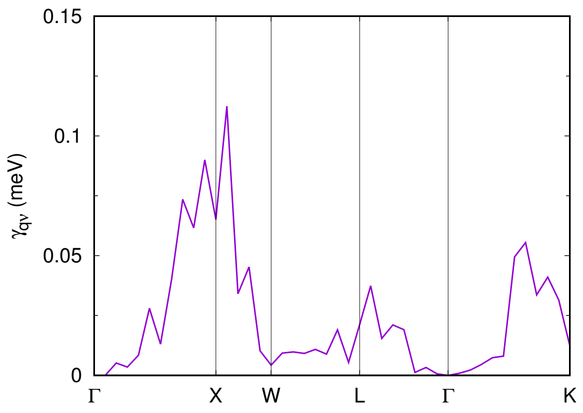
```
$ mpirun -np 4 /home/nfs3/smr3191/q-e/bin/epw.x -npool 4 < pb.epw.in3 > pb.epw.out3
```

Note: 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 a few phonon energies from the Fermi level will contribute to the phonon linewidth, therefore we can use fsthick $= 1$ (eV).

You can see that now the phonon linewidths and coupling strengths are printed in output 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 stored in the files `linewidth.phself` and `lambda.phself`, 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> plot "linewidth.phself" u 1:4 every 3::2 w l lw 2
```

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



Although these linewidths are still unconverged, are you able to explain the main features?

▶ Let's calculate the total (integrated) electron-phonon coupling strength λ and the isotropic Eliashberg spectral function $\alpha^2F(\omega)$ (see Eq. (3) of tutorial Tue.3) by setting a2f $=$ .true. and modifying the input as follows:

```
$ cp pb.epw.in3 pb.epw.in4
```

```
 ...                                                                    pb.epw.in4
 a2f         = .true.

 ...
 !filqf       = 'lead_band.kpt'
 !filkf       = 'lead_band.kpt'
 nkf1         = 20
 nkf2         = 20
 nkf3         = 20
 nqf1         = 12
 nqf2         = 12
 nqf3         = 12
 ...
```

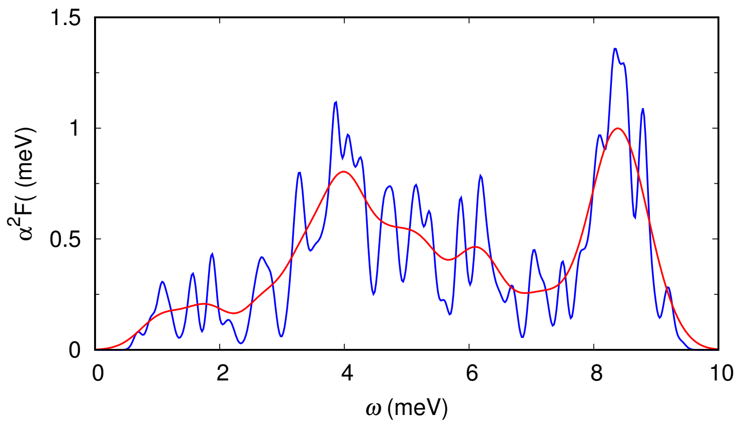
```
$ mpirun -np 4 /home/nfs3/smr3191/q-e/bin/epw.x -npool 4 < pb.epw.in4 > pb.epw.out4
```

Note that now we need to perform a Brillouin-zone integration also over $\mathbf{q}$, hence we interpolate the matrix elements onto a $20 \times 20 \times 20$ $\mathbf{k}$ grid and $12 \times 12 \times 12$ $\mathbf{q}$ grid. At the end of the calculation we should get λ:

```
===================================================================
Eliashberg Spectral Function in the Migdal Approximation
===================================================================

lambda :     1.7886205
```

Note that the converged value for λ should be around 1.1.

You can plot $\alpha^2F(\omega)$ from the file `lead.a2f.01`. This file contains 11 columns: the first one is the frequency grid, and the remainder 10 columns correspond to $\alpha^2F(\omega)$ calculated using different values of broadening, starting from `degaussq` = 0.05 meV, with steps of `delta_qsmear` = 0.05 meV. If you plot this function for two values of broadening (0.1 meV in blue and 0.5 meV in red, for example) you should obtain:



▶ Try increasing the coarse $\mathbf{q}$ grid to better converge the phonons. If you use a $4 \times 4 \times 4$ $\mathbf{q}$ grid, what $\mathbf{k}$ grids should you use?

▶ Try using denser $\mathbf{k}$ and $\mathbf{q}$ meshes for the interpolation (increased `nqf1,...,nkf3`), and investigate the convergence of λ and $\alpha^2F(\omega)$ with respect to the BZ sampling and to the smearing `degaussw`. You should note that the sampling over the electron wavevectors $\mathbf{k}$ is more critical than the sampling over the phonon wavevectors $\mathbf{q}$.
Is the convergence faster if you use randomly generated grids? The input parameters are `rand_q = .true.` and `rand_nq = 2000`, for example.
For the converged results and parameters, you can refer to the figure in tutorial Tue.5 and to the paper https://www.sciencedirect.com/science/article/pii/S0010465516302260

Note: during the first `EPW` run, the maximum number of CPUs you can use is limited by the total number of $\mathbf{k}$ points in the irreducible BZ (since there is no $\mathbf{G}$ parallelization). However, when restarting from `epwread = .true.` you can also use a larger number of CPUs than in the first run. Try running with 8 CPUs and observe the speedup of the calculation.

▶ Explore the effect of spin-orbit coupling in lead by re-running all calculations including SOC. You need to:

- add `noncolin = .true.` and `lspinorb = .true.` in the `&system` namelist of the `scf` and `nscf` calculations;
- change the lattice parameter to the relaxed value including SOC, `celldm(1) = 9.269771512`;

- increase the number of bands in the nscf calculation to `nbnd=26`;
- double the number of bands `nbndsub` and `nbndskip` in the `EPW` input file.

Exercise 2

In this exercise we will examine the electron-phonon interactions in **SiC**. As you learned in tutorial Tue.5, SiC is a polar material, where the long-range polar coupling results in a $1/|\mathbf{q}|$ divergence of the matrix elements near Γ. Here we will see how to correctly interpolate the matrix elements in this case, and we will calculate the electron linewidths.

▶ Run a self-consistent calculation for SiC as in Exercise 2 of tutorial Tue.5.

```
$ cd ../ ; mkdir exercise2; cd exercise2
$ cp ../../tuto_Tues5/exercise2/scf.in sic.scf.in
$ cp ../../tuto_Tues5/exercise2/*.UPF .
$ mpirun -np 4 /home/nfs3/smr3191/q-e/bin/pw.x -npool 4 < sic.scf.in > sic.scf.out
```

▶ Run a phonon calculation on a homogeneous $3 \times 3 \times 3$ $\mathbf{q}$-point grid.

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

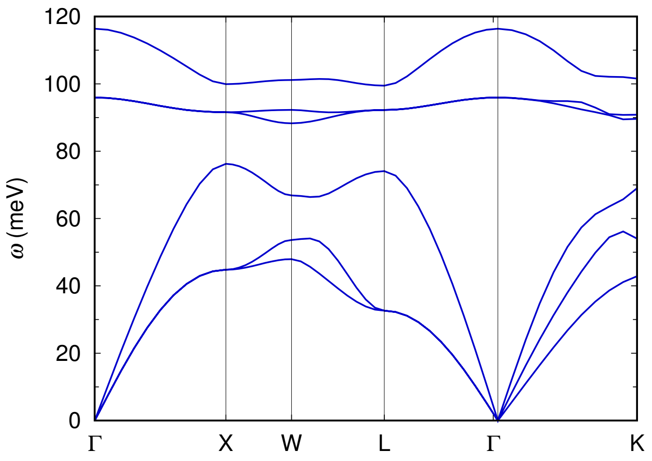
```
$ mpirun -np 4 /home/nfs3/smr3191/q-e/bin/ph.x -npool 4 < sic.ph.in > sic.ph.out
```

Note that the output now contains also the Born effective charges Z^* and the electronic dielectric constant ϵ_∞:

```
      Electric Fields Calculation

      ....

      End of electric fields calculation

            Dielectric constant in cartesian axis

            (       7.214167210       0.000000000       0.000000000 )
      ....

            Effective charges (d Force / dE) in cartesian axis

             atom       1    Si
       Ex   (        2.67035         0.00000         0.00000 )
       ....
```

These quantities are automatically calculated for an insulating system, using the finite response to an electric field, and determine the energy splitting of the TO and LO phonons.

You should now know well how to plot the phonon dispersions using `q2r.x` and `matdyn.x`. The plot should look like this:



Note that now there are 3 acoustic and 3 optical branches, separated by an energy gap, and that there is the LO-TO splitting at Γ.

▶ Gather the `.dyn`, `.dvscf` and `patterns` files into a `save/` directory as in Exercise 1 using the `pp.py` script:

```
$ python /home/nfs3/smr3191/q-e/EPW/bin/pp.py
```

▶ Run a non self-consistent calculation on a homogeneous $6 \times 6 \times 6$ $\mathbf{k}$ grid. To do so, first copy the input file for the self-consistent calculation into a new file `pb.nscf.in`; then modify the input variable `calculation` to `nscf`, set the number of bands to `nbnd=4` (we will look only at the valence bands here), and delete the last two lines specifying the $\mathbf{k}$-point grid. Then you can use again the script `kmesh.pl` to generate the homogeneous $\mathbf{k}$ grid:

```
$ /home/nfs3/smr3191/q-e/wannier90-2.1.0/utility/kmesh.pl 6 6 6 >> sic.nscf.in
```

Now your input file should look like this:

```
&control                                                               sic.nscf.in
    calculation     = 'nscf'
    prefix          = 'sic'
    wf_collect      = .false.
    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
...
```

Now you can run:

```
$ mpirun -np 4 /home/nfs3/smr3191/q-e/bin/pw.x -npool 4 < sic.nscf.in > sic.nscf.out
```

▶ Calculate the interpolated electron-phonon matrix elements along some Brillouin-zone directions (input variable `prtgkk = .true.`).

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

  elph        = .true.
  kmaps       = .false.
  epbwrite    = .true.
  epbread     = .false.
  epwwrite    = .true.
  epwread     = .false.
  lpolar      = .true.

  wannierize  = .true.
  nbndsub     =  4
  nbndskip    =  0
  num_iter    = 300
  proj(1)     = 'Si:sp3'

  elecselfen  = .false.
  phonselfen  = .false.
  a2f         = .false.
  prtgkk      = .true.

  parallel_k  = .true.
  parallel_q  = .false.

  fsthick     = 7.0
  eptemp      = 20
  degaussw    = 0.05

  filqf       = 'path.dat'
  nkf1        = 1
  nkf2        = 1
  nkf3        = 1
  nk1         = 6
  nk2         = 6
  nk3         = 6
  nq1         = 3
  nq2         = 3
  nq3         = 3
 /
  4
  0.000000000000000E+00   0.000000000000000E+00   0.000000000000000E+00
 -0.333333333333333E+00   0.333333333333333E+00  -0.333333333333333E+00
  0.000000000000000E+00   0.666666666666667E+00   0.000000000000000E+00
  0.666666666666667E+00  -0.555111512312578E-16   0.666666666666667E+00
```

Since the unit cell of SiC is the same as Pb, we can re-use the same BZ path, after copying it from Exercise 1:

```
$ cp ../exercise1/lead_band.kpt path.dat
```

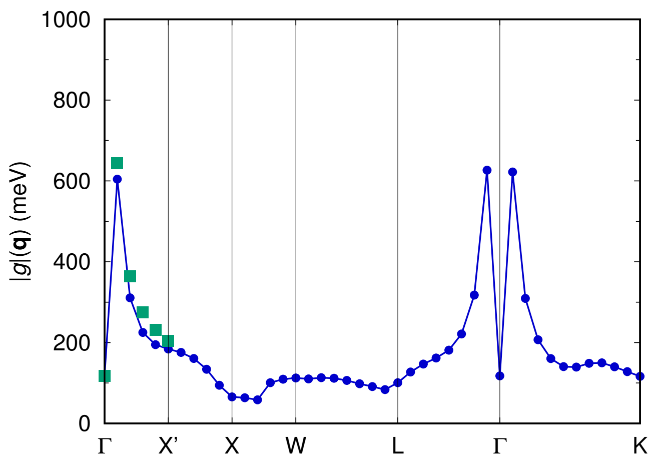
Note also that in order to speed up the calculation, we chose to only print the matrix elements for the initial electronic states at $\mathbf{k} = \Gamma$ (nkf1=nkf2=nkf3=1).

Since 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 matrix element g before interpolation and adding it back after interpolation (see lectures Wed.3 and Thu.1). An analogous strategy is implemented to correctly interpolate the dynamical matrix including the long-range dipole-dipole interactions which result in the LO-TO splitting.

The matrix elements for each $\mathbf{q}$ along the BZ path are written in the output, with columns corresponding to each band and phonon mode. An average over degenerate bands and modes has already been performed. If you want to plot the matrix elements for the valence-band top $(n = m = 4)$ and the highest-energy phonon branch $(\nu = 6)$, for example, you can type:

```
$ grep '4          4          6' sic.epw.out | awk '{print $7}' > matel.dat
```

and the plot should look like:



If you use a denser $\mathbf{q}$-point path, the matrix elements will keep diverging near Γ, in agreement with the direct calculations using ph.x.

The matrix elements calculated in tutorial Tue.5 are reproduced as green squares. 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 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=.true.`.

▶ Plot the interpolated electron and phonon band structure and compare them with the results from `pw.x` (you will need to perform a band structure calculation) and from `matdyn.x`, respectively. Note that we are only wannierizing the 4 valence bands. You can also look at the decays, as done in Exercise 1.

▶ 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. To do so, we need to use the input variable `elecselfen = .true.`, the $\mathbf{k}$-point path, and a homogeneous $\mathbf{q}$ grid for the integration. Copy the `EPW` input into a new one:

```
$ cp sic.epw.in sic.epw.in2
```

and modify it as follows:

```
 ...                                                          sic.epw.in2
 elph        = .true.
 kmaps       = .true.
 epbwrite    = .false.
 epbread     = .false.
 epwwrite    = .false.
 epwread     = .true.
 lpolar      = .true.

 wannierize  = .false.
 ...

 elecselfen  = .true.
 phonselfen  = .false.
 a2f         = .false.
 prtgkk      = .false.
 efermi_read = .true.
 fermi_energy= 9.6
 ...

 !filqf        = 'path.dat'
 filkf       = 'path.dat'
 nqf1        = 20
 nqf2        = 20
 nqf3        = 20
 ...
```

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

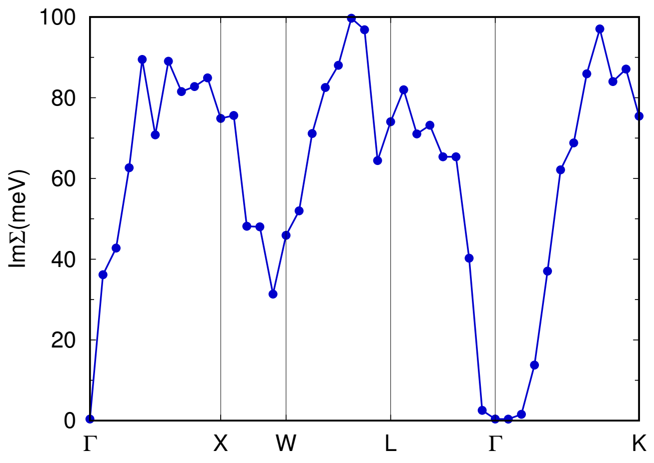
In the output you can monitor the progression of the $\mathbf{q}$ integration, before reading the electron self-energy for each $\mathbf{k}$ point:

```
     Progression iq (fine) =          50/       8000
```

```
Progression iq (fine) =        100/      8000
...
...
Average over degenerate eigenstates is performed
WARNING: only the eigenstates within the Fermi window are meaningful

ik =       1 coord.:    0.0000000   0.0000000   0.0000000
--------------------------------------------------------------------
E(   2 )=  -0.2405 eV   Re[Sigma]=      95.950765 meV Im[Sigma]=       0.392034 meV
E(   3 )=  -0.2405 eV   Re[Sigma]=      95.950765 meV Im[Sigma]=       0.392034 meV
E(   4 )=  -0.2405 eV   Re[Sigma]=      95.950765 meV Im[Sigma]=       0.392034 meV
--------------------------------------------------------------------
...
```

Note 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 linewidths you can use the file `linewidth.elself` that has been created. For example, ImΣ for the highest valence band should look like:

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



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

Can you understand the behavior of the linewidths along the Brillouin-zone path? It is useful to look also at the electronic band structure.

Note: You can also look at the contribution of each phonon mode by using `iverbosity = 3` in the input. The file `linewidth.elself` will then contain the mode-resolved linewidths. Can you tell which phonon has the largest contribution and why?

▶ Try increasing the $\mathbf{q}$ grid, and also using a random set of $\mathbf{q}$ points: you will see the linewidths are not well converged yet. For polar materials it is indeed more difficult to converge Brillouin-zone integrals, due to the $1/|\mathbf{q}|$ divergence of the Fröhlich matrix elements.

Exercise 3

In this exercise we will calculate the spectral function of n-doped **MgO**.

▶ Run a self-consistent calculation for MgO. First, create a new folder and download the pseudopotentials: here we will use pseudopotentials from the ONCV library.

```
$ cd ../ ; mkdir exercise3; cd exercise3
$ wget http://www.quantum-simulation.org/potentials/sg15_oncv/upf/O_ONCV_PBE-1.0.upf
$ wget http://www.quantum-simulation.org/potentials/sg15_oncv/upf/Mg_ONCV_PBE-1.0.upf
```

Prepare the scf input and run the calculation:

```
&control                                                                    mgo.scf.in
 calculation='scf',
 prefix='mgo',
 pseudo_dir= './',
 outdir = './',
/
&system
 ibrav=2,
 celldm(1) = 7.9595
 nat=2,
 ntyp=2,
 ecutwfc = 30,
/
&electrons
 conv_thr=1.d-10,
/
ATOMIC_SPECIES
 Mg 24.305  Mg_ONCV_PBE-1.0.upf
 O  15.999  O_ONCV_PBE-1.0.upf
ATOMIC_POSITIONS crystal
 Mg       0.0 0.0 0.0
 O        0.5 0.5 0.5
K_POINTS {automatic}
 6 6 6 0 0 0
```

```
$ mpirun -np 4 /home/nfs3/smr3191/q-e/bin/pw.x -npool 4 < mgo.scf.in > mgo.scf.out
```

▶ Run a phonon calculation on a homogeneous $4 \times 4 \times 4$ $\mathbf{q}$-point grid.

```
--                                                                           mgo.ph.in
&inputph
  prefix    = 'mgo'
  fildvscf  = 'dvscf'
  ldisp     = .true
  fildyn    = 'mgo.dyn'
  nq1=4,
  nq2=4,
  nq3=4,
  tr2_ph    =  1.0d-12
/
```

```
$ mpirun -np 4 /home/nfs3/smr3191/q-e/bin/ph.x -npool 4 < mgo.ph.in > mgo.ph.out
```

The Born effective charges Z^* and the electronic dielectric constant ϵ_∞ are also computed, since MgO is a polar insulator.

Plot the phonon dispersions using `q2r.x` and `matdyn.x` (note that for real calculations you should relax the structure using a converged `ecutwfc`, and use at least a $4 \times 4 \times 4$ $\mathbf{q}$ grid for the phonons).

▶ Gather the `.dyn`, `.dvscf` and `patterns` files into a `save/` directory as in Exercise 1 and 2, using the script `/home/nfs3/smr3191/q-e-/EPW/bin/pp.py`.

▶ Run a non self-consistent calculation on a homogeneous $4 \times 4 \times 4$ $\mathbf{k}$ grid, preparing the input as in Esercise 1 and 2, with the difference that now you will prompt:
`$ /home/nfs3/smr3191/q-e/wannier90-2.1.0/utility/kmesh.pl 4 4 4 >> mgo.nscf.in`
Use `nbnd=12`.

```
$ mpirun -np 4 /home/nfs3/smr3191/q-e/bin/pw.x -npool 4 < mgo.nscf.in > mgo.nscf.out
```

▶ Run `EPW`: wannierize the electronic band structure and check the interpolation using `band_plot=.true.`.

```
--                                                                   mgo.epw.in
&inputepw
  prefix      = 'mgo'
  amass(1)    = 24.305
  amass(2)    = 15.999
  outdir      = './'
  dvscf_dir   = './save'

  elph        = .true.
  kmaps       = .false.
  epbwrite    = .true.
  epbread     = .false.
  epwwrite    = .true.
  epwread     = .false.
  lpolar      = .true.

  wannierize  = .true.
  nbndsub     =  4
  nbndskip    =  5
  num_iter    = 300
  dis_win_min = 0
  dis_froz_max= 11.0
  dis_win_max = 21.8
  proj(1)     = 'O:p'
  proj(2)     = 'Mg:s'

  elecselfen  = .false.
  phonselfen  = .false.
  a2f         = .false.
  band_plot   = .true.

  parallel_k  = .true.
  parallel_q  = .false.

  fsthick     = 2.0
  eptemp      = 20
  degaussw    = 0.05

  filqf       = 'path.dat'
  filkf       = 'path.dat'

  nk1         = 4
  nk2         = 4
  nk3         = 4
  nq1         = 4
  nq2         = 4
  nq3         = 4
 /
  8
  0.000000000000000E+00   0.000000000000000E+00   0.000000000000000E+00
 -0.250000000000000E+00   0.250000000000000E+00  -0.250000000000000E+00
  0.500000000000000E+00  -0.500000000000000E+00   0.500000000000000E+00
  0.000000000000000E+00   0.500000000000000E+00   0.000000000000000E+00
  0.750000000000000E+00  -0.250000000000000E+00   0.750000000000000E+00
  0.500000000000000E+00   0.000000000000000E+00   0.500000000000000E+00
  0.000000000000000E+00  -0.100000000000000E+01   0.000000000000000E+00
 -0.500000000000000E+00  -0.100000000000000E+01   0.000000000000000E+00
```

You can use again the Brillouin-zone path from Exercise 1 and 2.

▶ Calculate the spectral function $A(\mathbf{k}, \omega)$ in the Fan-Migdal approximation (see lectures Thu.1 and Thu.2) for n-doped MgO. Doping will be treated in the rigid band approximation, that is by a rigid shift of the Fermi level. First, introduce a small doping charge of 0.00037 electrons/unit cell, which corresponds to a density of about 2×10^{19} cm^{-3}: open the file `crystal.fmt` and modify line 3, from 16 (number of electrons) to 16.00037. Then calculate the Fermi level using `EPW` modifying the input as follows:

```
$ cp mgo.epw.in mgo.epw.in2
```

```
  ...                                                                  mgo.epw.in2
  kmaps       = .true.
  epbwrite    = .false.
  epbread     = .false.
  epwwrite    = .false.
  epwread     = .true.
  lpolar      = .true.

  wannierize  = .false.
  ...
  band_plot   = .false.
  ...

  degaussw    = 0.002

  !filqf        = 'path.dat'
  !filkf        = 'path.dat'
  nkf1        = 100
  nkf2        = 100
  nkf3        = 100
  nqf1        = 1
  nqf2        = 1
  nqf3        = 1
  ...
```

Note the very dense $\mathbf{k}$ grid and small smearing `degaussw`: these parameters are needed in order to converge the Fermi energy at very small dopings. **Stop** the calculation just after the Fermi energy has been written in the output. You should obtain:

```
      Skipping the first     5 bands:

      The Fermi level will be determined with    6.00037 electrons
```

```
Fermi energy is calculated from the fine k-mesh: Ef =  13.381852 eV
```

hence the Fermi energy is shifted about 80 meV above the conduction band bottom.

For calculating the electron spectral function you need to specify `specfun_el = .true.` and an energy range, as referred to the Fermi level. The latter is read from the input file, and corresponds to the value that you just calculated. Modify the input as follows:

```
$ cp mgo.epw.in2 mgo.epw.in3
```

```
  ...                                                                    mgo.epw.in3
  specfun_el  = .true.
  wmax_specfun = 0.1
  wmin_specfun = -0.5
  nw_specfun   = 300
  efermi_read = .true.
  fermi_energy= 13.381852

  fsthick     = 2.0
  eptemp      = 20
  degaussw    = 0.01

  !filqf       = 'path.dat'
  filkf       = 'path2.dat'
  nqf1         = 40
  nqf2         = 40
  nqf3         = 40
  ...
```

The file path2.dat contains a dense mesh of $\mathbf{k}$ points only along the ΓX direction (to speed up the calculation, we only look at this BZ direction). This path is provided in the exercise3/ folder, and it can be generated for example using the following script:

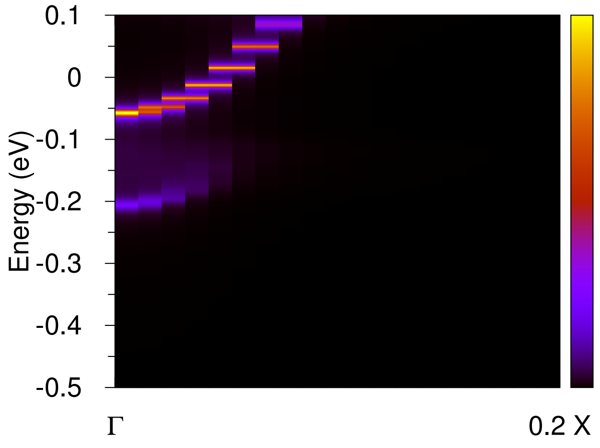
```
cat > kpath.py << EOF
import numpy as np

nk=0.2/0.005
print int(nk)

for k in np.arange(0.0,0.2,0.005):
  print ' 0.0 '+str(k)+' '+str(k)+' 1.0'
EOF
```

```
$ python kpath.py > path2.dat
```

After running EPW, the calculated electron spectral function will be saved in the file specfun.elself, whereas the file specfun_sup.elself contains the frequency-dependent self-energy $\Sigma_{n\mathbf{k}}(\omega)$. You can plot the spectral function for example by using gnuplot, with the command:

```
gnuplot> set view map ; splot "specfun.elself" w pm3d
```

You should see a satellite band below the conduction band bottom, however its position and intensity are not converged. Try using a finer $\mathbf{q}$ mesh and/or random $\mathbf{q}$ points, and observe the speed of convergence. A more converged plot is reported below.


0.1
0
-0.1
-0.2
-0.3
-0.4
-0.5
Energy (eV)
Γ
0.2 X