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

Calculations of superconducting properties with EPW

Hands-on session (Thu.6)

Hands-on based on QE-v.6.7 and EPW-v.5.4Beta

Exercise 1

In this example we are going to calculate the superconducting properties of fcc Pb by solving the isotropic Migdal-Eliashberg equations. The theory related to this tutorial can be found in Phys. Rev. B **87**, 024505 (2013).

```
$ cd $SCRATCH
$ mkdir EPW-SCHOOL ; cd EPW-SCHOOL
$ cp /work2/06868/giustino/EPW-SCHOOL/Thu.6.Margine.tar .
$ tar -xvf Thu.6.Margine.tar; cd Thu.6.Margine/exercise1
```

▶ 1st step: Run a self-consistent calculation on a homogeneous 12x12x12 **k**-point grid and a phonon calculation on a homogeneous 3x3x3 **q**-point grid using the following inputs and jobscript:
Note: The `ecutwfc` needs to be much larger for real calculations.

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

job.ph

```
#!/bin/bash
#SBATCH -J job.ph             # Job 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 -p small
#SBATCH --reservation=EPW-SCHOOL-06-17-2021

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

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

scf.in

```
--
&control
  calculation  = 'scf'
  restart_mode = 'from_scratch',
  prefix       = 'pb',
  pseudo_dir   = '../',
  outdir       = './'
/
&system
  ibrav         = 2,
  celldm(1)     = 9.27,
  nat           = 1,
  ntyp          = 1,
  ecutwfc       = 30.0
  occupations   = 'smearing',
```

```
  smearing      = 'mp',
  degauss       = 0.025
/
&electrons
  diagonalization = 'david'
  mixing_beta   = 0.7
  conv_thr      = 1.0d-12
/
ATOMIC_SPECIES
  Pb 207.2 pb_s.UPF
ATOMIC_POSITIONS
  Pb 0.00 0.00 0.00
K_POINTS {automatic}
  12 12 12 0 0 0
```

```
--                                                                    ph.in
 &inputph
 prefix   = 'pb',
 fildyn   = 'pb.dyn.xml',
 fildvscf = 'dvscf',
 tr2_ph   =  1.0d-17
 ldisp    = .true.,
 nq1 = 3,
 nq2 = 3,
 nq3 = 3
```

During the run, notice the irreducible (IBZ) q-point grid:

```
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
```

▶ 2nd step: Gather the `.dyn`, `.dvscf`, and `patterns` files into a new `save` directory using the pp.py python script.

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

The script will ask you to provide the `prefix` of your calculation (here "pb").

▶ 3rd step: Do a non self-consistent calculation on a 3×3×3 **homogeneous** and Γ-**centered grid between [0,1[in crystal coordinates** and an EPW calculation for the superconducting properties using the following inputs and jobscript:

Note 1: The homogeneous **k** grid for the non self-consistent calculations can be generated using the script `kmesh.pl`

```
$ /work2/06868/giustino/EPW-SCHOOL/q-e/wannier90-3.1.0/utility/kmesh.pl 3 3 3
```

Note 2: A non self-consistent calculation requires the charge density found from a previous self-consistent run with `pw.x`. In the jobscript `job.epw1` you can see that a self-consistent calculation is run first with the same `scf.in` file used in the `phonon` directory. Alternatively, one can make the `pb.save` directory and copy there the files from `phonon/pb.save` For this in `job.epw1` you need to comment the line

```
#ibrun $PATHQE/bin/pw.x -nk 8 -in scf.in > scf.out
```

and uncomment the following three lines

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

Note 3: EPW calculations with `ephwrite = .true.` require that the fine $\mathbf{k}$ or $\mathbf{q}$ grids are commensurate, i.e., `nkf1, nkf2, nkf3` to be multiple of `nqf1, nqf2, nqf3`.

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

job.epw1

```
#!/bin/bash
#SBATCH -J job.epw1           # Job 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 -p small
#SBATCH --reservation=EPW-SCHOOL-06-17-2021

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

ibrun $PATHQE/bin/pw.x -nk 8 -in scf.in > scf.out
 #alternatively to re-run a scf calculation copy files from ../phonon/pb.save
#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 8 -in nscf.in > nscf.out
ibrun $PATHQE/bin/epw.x -nk 8 -in epw1.in > epw1.out
```

nscf.in

```
--
&control
  calculation  = 'nscf',
  restart_mode = 'from_scratch',
  prefix       = 'pb',
  pseudo_dir   = '../',
  outdir       = './',
  verbosity    = 'high'
/
&system
  ibrav        = 2,
  celldm(1)    = 9.27,
  nat          = 1,
  ntyp         = 1,
  ecutwfc      = 30.0,
  occupations  = 'smearing',
  smearing     = 'mp',
  degauss      = 0.025,
  nbnd         = 10,
/
&electrons
  diagonalization = 'david'
  mixing_beta  = 0.7
  conv_thr     = 1.0d-12
/
ATOMIC_SPECIES
  Pb 207.2   pb_s.UPF
ATOMIC_POSITIONS crystal
  Pb  0.000000000   0.000000000   0.000000000
K_POINTS crystal
  27
  0.00000000  0.00000000  0.00000000  3.703704e-02
  0.00000000  0.00000000  0.33333333  3.703704e-02
  ...
```

epw1.in

```
--
&inputepw
  prefix      = 'pb',
  amass(1)    = 207.2
```

```
  outdir      = './'
  dvscf_dir   = '../phonon/save'  ! directory where .dyn, .dvscf and prefix.phsave/patterns.xx.yy
                                  ! files obtained from phonon calculation are stored
  ep_coupling = .true.          ! run e-ph coupling calculation
  elph        = .true.          ! calculate e-ph coefficients

  epwwrite    = .true.          ! write e-ph matrices in Wann representation
  epwread     = .false.         ! read e-ph matrices from 'prefix.epmatwp' file

  wannierize    = .true.               ! calculate Wannier functions using W90 library
  nbndsub       =  4                   ! number of Wannier functions to utilize
  bands_skipped = 'exclude_bands = 1-5' ! number of bands skipped during wannierization

  num_iter    = 300
  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 K 0.375 0.375 0.75'
  wdata(7)  = 'K 0.375 0.375 0.75 G 0.00 0.00 0.00'
  wdata(8)  = 'G 0.00 0.00 0.00 L 0.50 0.50 0.50'
  wdata(9)  = 'end kpoint_path'
  wdata(10) = 'bands_plot_format = gnuplot'

  fsthick     = 0.4             ! Fermi window thickness [eV]
  degaussw    = 0.1             ! smearing in energy-conserving delta functions in [eV]
  degaussq    = 0.05            ! smearing for sum over q in the e-ph coupling in [meV]

  ephwrite    = .true.          ! write ephmatXX, egnv, freq, and ikmap files in prefix.ephmat directory
  eliashberg  = .true.          ! calculate Eliashberg spectral function

  liso    = .true.              ! solve isotropic ME eqs.
  limag   = .true.              ! solve ME eqs. on imaginary axis
  lpade   = .true.              ! solve ME eqs. on real axis using Pade approximants
  lacon   = .true.              ! analytic continuation of ME eqs. from imaginary to real axis

  nsiter  = 500                 ! number of self-consistent iterations when solving ME eqs.
  npade   = 40                  ! percentage of Matsubara points used in Pade continuation.
  conv_thr_iaxis = 1.0d-3       ! convergence threshold for solving ME eqs. on imaginary axis
  conv_thr_racon = 1.0d-3       ! convergence threshold for solving ME eqs. on real axis

  wscut   = 0.1                 ! upper limit over Matsubara freq. summation in ME eqs on imag.axis in [eV]
  muc     = 0.1                 ! effective Coulomb potential used in the ME eqs.
  temps   = 0.3 0.9 1.5 2.1 2.7 3.3 3.9 4.2 4.4 4.5 4.6 4.65   ! list of temperatures at which ME eqs.
            ! are solved [equally spaced temperature points can also be used: see epw2.in]

  nk1  = 3
  nk2  = 3                      ! dimensions of coarse electronic grid
  nk3  = 3

  nq1  = 3
  nq2  = 3                      ! dimensions of coarse phonon grid
  nq3  = 3

  mp_mesh_k = .true.            ! use irreduciable electronic fine mesh
  nkf1 = 18
  nkf2 = 18                     ! dimensions of fine electronic grid
  nkf3 = 18

  nqf1 = 18
  nqf2 = 18                     ! dimensions of the phonon grid
  nqf3 = 18
/
```

With the above input, we are instructing EPW to:

- Fourier-transform the electron-phonon matrix elements from a coarse 3×3 ×3 to a dense 18×18 ×18 $\mathbf{k}/\mathbf{q}$-point grids.

```
Using uniform q-mesh:    18   18   18
Size of q point mesh for interpolation:       5832
Using uniform MP k-mesh:    18   18   18
Size of k point mesh for interpolation:        390
Max number of k points per pool:                50
```

- Pre-compute the $\mathbf{q}$-points that fall within the `fsthick`. If at a specific $\mathbf{q}$-point at least one $\mathbf{k}+\mathbf{q}$ eigenvalue falls within the user-defined `fsthick`, then the $\mathbf{q}$-point is selected.

```
Number selected, total            100            107
Number selected, total            200            215
......
Number selected, total           5500           5736
We only need to compute     5579 q-points
```

- Write on disk in the `pb.ephmat` directory the: (1) `ephmatXX` files (one per CPU) containing the electron-phonon matrix elements within the Fermi window (`fsthick`) on the dense $\mathbf{k}$ and $\mathbf{q}$ grids, (2) `freq` file containing the phonon frequencies on the dense $\mathbf{q}$ grid, (3) `egnv` file containing the eigenvalues within the Fermi window on the dense $\mathbf{k}$ grid, and (4) `ikmap` file containing the index of the $\mathbf{k}$-points on the dense (irreducible) grid within the Fermi window. All these files are produced by setting `ephwrite = .true.`. The files are unformatted and required for solving the Migdal-Eliashberg equations.

```
Nr. of irreducible k-points on the uniform grid:       195

Finish mapping k+sign*q onto the fine irreducibe k-mesh and writing .ikmap file

Nr irreducible k-points within the Fermi shell =        30 out of       195

Progression iq (fine) =        100/      5579
Progression iq (fine) =        200/      5579
......
Progression iq (fine) =       5500/      5579
             Fermi level (eV) =    0.117577170436144D+02
DOS(states/spin/eV/Unit Cell) =    0.296296405366909D+00
       Electron smearing (eV) =    0.100000000000000D+00
            Fermi window (eV) =    0.400000000000000D+00

Finish writing .ephmat files
```

- Solve the isotropic Migdal-Eliashberg equations on the imaginary frequency axis. This is achieved by setting the keywords `eliashberg = .true.`, `liso = .true.`, and `limag = .true.` in the input file. The equations are solved self-consistently for each temperature value specified in the input file. The calculation at each temperature ends when either the converge threshold (`conv_thr_iaxis`) or the maximum number of iterations (`nsiter`) is reached.

  Note 1: If at a specific temperature the maximum number of iterations is reached without achieving convergence, the code will stop and not move to the next temperature in the list.

  Note 2: Because the electron-phonon matrix elements do not depend on the temperature at which the Migdal-Eliashberg equations are solved, they can be reused in subsequent EPW calculations at different temperatures. This is the reason why `ephmatXX` files are saved in the `pb.ephmat` directory.

The isotropic Migdal-Eliashberg equations take the following form:

$$\begin{aligned} Z(i\omega_j) &= 1+\frac{\pi T}{\omega_j}\sum_{j'}\frac{\omega_{j'}}{\sqrt{\omega_{j'}^2+\Delta(i\omega_j)}}\lambda(\omega_j-\omega_{j'}) \\ Z(i\omega_j)\Delta(i\omega_j) &= \pi T\sum_{j'}\frac{\Delta(i\omega_{j'})}{\sqrt{\omega_{j'}^2+\Delta^2(i\omega_{j'})}}\left[\lambda(\omega_j-\omega_{j'})-\mu_{\rm c}^*\right] \end{aligned} \quad (1)$$

The semiempirical Coulomb parameter $\mu_{\rm c}^*$ is provided as an input varible `muc` in the EPW calculation. The isotropic electron-phonon coupling strength $\lambda(\omega_j)$ entering in Eqs. (1) is defined as:

$$\lambda(\omega_j)=\frac{1}{N_{\rm F}}\sum_{nm\nu}\int\frac{d\mathbf{k}}{\Omega_{\rm BZ}}\int\frac{d\mathbf{q}}{\Omega_{\rm BZ}}|g_{mn\nu}(\mathbf{k},\mathbf{q})|^2\frac{2\omega_{\mathbf{q}\nu}}{\omega_j^2+\omega_{\mathbf{q}\nu}^2}\delta(\epsilon_{n\mathbf{k}}-\epsilon_{\rm F})\delta(\epsilon_{m\mathbf{k}+\mathbf{q}}-\epsilon_{\rm F}) \quad (2)$$

While the calculation is running, notice in the `epw1.out` file the different steps a full `EPW` run goes through. Once the interpolation on the fine mesh is finished, the code writes and reads the files required for solving the Migdal-Eliashberg equations and then proceeds with solving the equations at the specified temperatures.

```
================================================================
Solve isotropic Eliashberg equations
================================================================
Finish reading freq file
......

1 bands within the Fermi window

Finish reading egnv file

Max nr of q-points =       956

Finish reading ikmap files

Start reading .ephmat files

Finish reading .ephmat files

Finish reading a2f file

Electron-phonon coupling strength =    1.4873625

Estimated Allen-Dynes Tc =     3.589804 K for muc =    0.10000

Estimated w_log in Allen-Dynes Tc =     2.735659 meV

Estimated BCS superconducting gap =     0.544448 meV

WARNING WARNING WARNING

The code may crash since tempsmax =    4.650 K is larger than Allen-Dynes Tc =     3.590 K

temp(  1) =      0.30000 K
```

```
Solve isotropic Eliashberg equations on imaginary-axis

Total number of frequency points nsiw(     1) =    616
Cutoff frequency wscut =     0.1001

iter      ethr          znormi      deltai [meV]
1   3.197750E+00   2.354594E+00   6.794532E-01
2   1.552187E-01   2.336376E+00   7.345737E-01
    ......
8   1.751527E-04   2.285990E+00   8.740018E-01
Convergence was reached in nsiter =      8
```

- Perform an analytic continuation of the solutions from the imaginary frequency axis to the real frequency axis. The analytic continuation can be done using Padé approximants (`lpade = .true.`) or an iterative procedure (`lacon = .true.`). The iterative procedure is performed self-consistently until either the converge threshold (`conv_thr_racon`) or the maximum number of iterations (`nsiter`) is reached.

  Note: If at a specific temperature the maximum number of iterations is reached without achieving convergence, the code will stop and not move to the next temperature in the list.

```
Pade approximant of isotropic Eliashberg equations from imaginary-axis to real-axis
Cutoff frequency wscut =     0.1000

pade    Re[znorm]   Re[delta] [meV]
246   2.286412E+00   8.742195E-01
Convergence was reached for N =    246 Pade approximants

raxis_pade   :      0.02s CPU      0.87s WALL (       1 calls)

Analytic continuation of isotropic Eliashberg equations from imaginary-axis to real-axis

Total number of frequency points nsw =   5000
Cutoff frequency wscut =     0.1000

 iter      ethr      Re[znorm]      Re[delta] [meV]
 1   1.288399E-01   2.286411E+00   8.742231E-01
 2   1.874291E-01   2.286411E+00   8.742231E-01
 ......
22   2.643651E-04   2.286411E+00   8.742231E-01
Convergence was reached in nsiter =     22
```

- At the end of the calculation, you should get a few output files at every given temperature. Note that the number of Matsubara frequency points decreases as the temperature increases because fewer frequencies $i\omega_j = i(2n+1)\pi T$ (n integer) are smaller than the cutoff frequency `wscut`.

The calculation of superconducting properties will be accompanied by significant I/O. In the following we will describe various physical quantities saved in the output files and how to process them. We will use `XX` in the name of the output files to indicate the temperature at which the equations are solved.

- Eliashberg spectral function and integrated electron-phonon coupling strength (λ).

pb.a2f, pb.a2f_iso, and pb.a2f_proj files are generated by setting eliashberg = .true.

pb.a2f file contains the isotropic Eliashberg spectral function $\alpha^2F(\omega)$ and cumulative electron-phonon coupling strength λ as a function of frequency ω (meV) for different phonon smearing values (see the end of the file for information about the smearing).

pb.a2f_iso and pb.a2f_proj files contain the Eliashberg spectral function as a function of frequency ω (meV), where the 2nd column in both files is the Eliashberg spectral function corresponding to the first smearing in pb.a2f. The remaining (3×number of atoms) columns in pb.a2f_proj contain the mode-resolved Eliashberg spectral functions corresponding to the first smearing in pb.a2f (there is no specific information on which modes correspond to which atomic species).

- Superconducting gap along the imaginary frequency axis and the real frequency axis.

  pb.imag_iso_XX files are generated by setting eliashberg = .true., liso = .true., and limag = .true.. Each file contains 4 columns: the Matsubara frequency $i\omega_j$ (eV) along the imaginary axis, the quasiparticle renormalization function $Z(i\omega_j)$, the superconducting gap $\Delta(i\omega_j)$ (eV), and the quasiparticle renormalization function $Z^N(i\omega_j)$ in the normal state.

  pb.pade_iso_XX files are generated by setting lpade = .true.. Each file contains 5 columns: the frequency ω (eV) along the real axis, the real part of the quasiparticle renormalization function $\mathrm{Re}Z(\omega)$, the imaginary part of the quasiparticle renormalization function $\mathrm{Im}Z(\omega)$, the real part of the superconducting gap $\mathrm{Re}\Delta(\omega)$ (eV), and the imaginary part of the superconducting gap $\mathrm{Im}\Delta(\omega)$ (eV).

  pb.acon_iso_XX files are generated by setting lacon = .true. and contain similar information as pb.pade_iso_XX.

▶ 4th step: Plot the superconducting gap along the imaginary and real frequency axis.

You can use the following gnuplot scripts to plot pb.imag_iso_000.30, pb.pade_iso_000.30, and pb.acon_iso_000.30. You should get something similar to Fig. 1 at 0.3 K.

```
$ gnuplot
gnuplot> set xlabel "iw (meV)"
gnuplot> set ylabel "Delta (meV)"
gnuplot> plot "pb.imag_iso_000.30" u ($1*1000):($3*1000) w l lw 2 lt rgb "black" \
> title "Delta-Imag"

gnuplot> set xlabel "w (meV)"
gnuplot> set ylabel "Delta (meV)"
gnuplot> plot "pb.pade_iso_000.30" u ($1*1000):($4*1000) w l lw 1 lt rgb "black" \
> title "Re(Delta)-Pade", \
> "" u ($1*1000):($5*1000) w l lw 1 lt rgb "red" title "Im(Delta)-Pade", \
> "pb.acon_iso_000.30" u ($1*1000):($4*1000) w l lw 1 lt rgb "blue" \
> title "Re(Delta)-analytic cont.", \
> "" u ($1*1000):($5*1000) w l lw 1 lt rgb "green" \
> title "Im(Delta)-analytic cont."
```

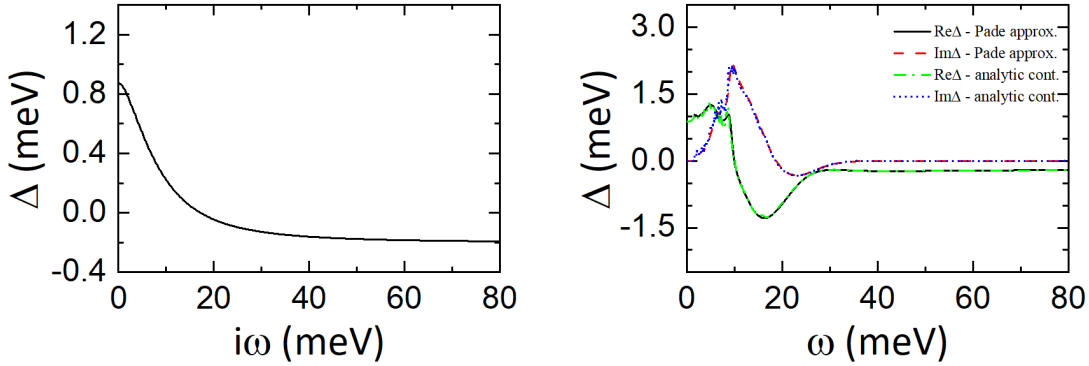

Fig. 1 Left: Superconducting gap along the imaginary axis (columns 1:3 from `pb.imag_iso_000.30`). Right: Superconducting gap on the real axis (columns 1:4 and 1:5 from `pb.pade_iso_000.30` and `pb.acon_iso_000.30`)

At convergence you should get something closer to:

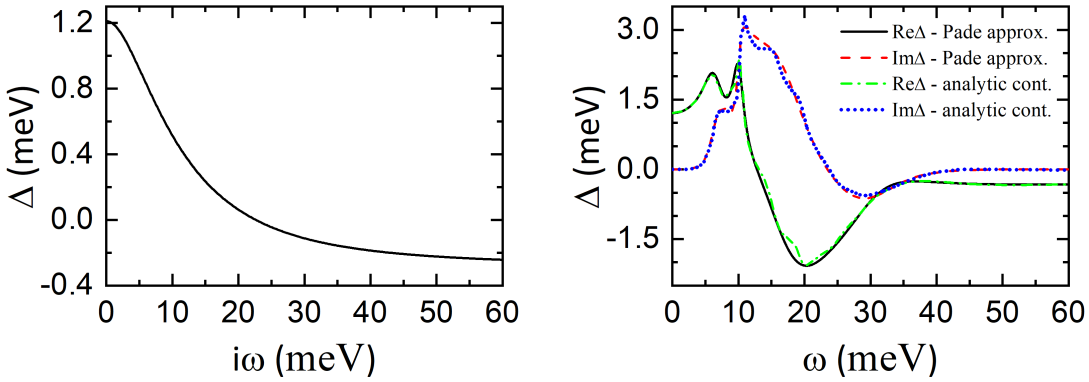

Fig. 2 Figure adapted from Phys. Rev. B **87**, 024505 (2013).

▶ 5th step: Plot the leading edge of the superconducting gap as a function of temperature.

Use the shell script `script_gap0_imag`(also shown below) to extract the leading edge of the superconducting gap as a function of temperature and save the information in a new file `pb.imag_iso_gap0`.

```
                                                                    script_gap0_imag
#!/bin/tcsh

awk 'FNR==2 {print FILENAME,$0}' pb.imag_iso_* | awk '{print $1 "   " $4*1000}' > pb.imag_iso_gap0
sed -i 's/pb.imag_iso_//' pb.imag_iso_gap0
```

```
$ ./script_gap0_imag
```

You can use the following gnuplot script to plot `pb.imag_iso_gap0`. You should get something similar to Fig. 3

```
$ gnuplot
gnuplot> set xlabel "T (K)"
gnuplot> set ylabel "Delta_0 (meV)"
gnuplot> set xrange [0:6]
gnuplot> set yrange [0:1.2]
gnuplot> plot "pb.imag_iso_gap0" with lp ls 3 notitle
```

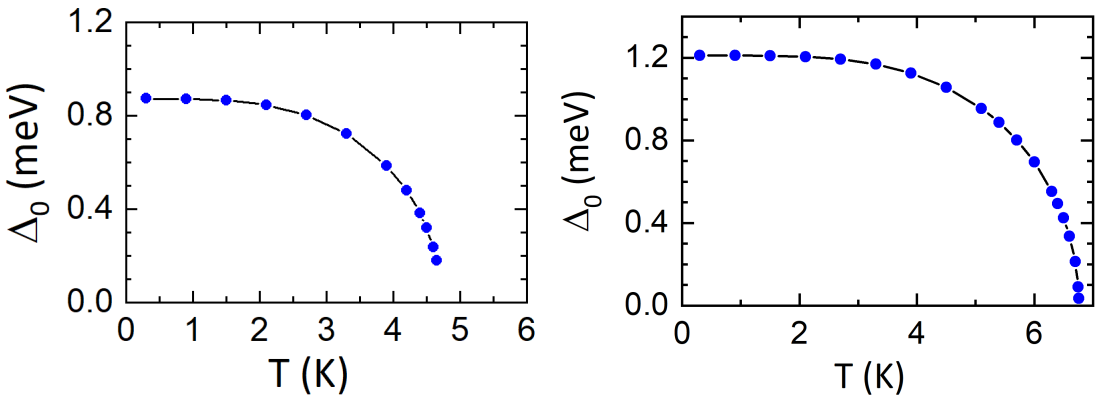

Fig. 3 Calculated isotropic gap of Pb as a function of temperature. At convergence you should get something closer to the right hand-side figure.

You can further extract the leading edge of the superconducting gap as a function of temperature from the calculations on the real axis and compare it with the one obtained on the imaginary axis shown in Fig. 3. You can use the shell scripts `script_gap0_pade` and `script_gap0_acon` to get the `pb.pade_iso_gap0` and `pb.acon_iso_gap0` files. Next plot these files using gnuplot as was done above for the `pb.imag_iso_gap0`.

```
$ ./script_gap0_pade
```

script_gap0_pade

```
#!/bin/tcsh

awk 'FNR==2 {print FILENAME,$0}' pb.pade_iso_* | awk '{print $1 "   " $5*1000}' > pb.pade_iso_gap0
sed -i 's/pb.pade_iso_//' pb.pade_iso_gap0
```

and

```
$ ./script_gap0_acon
```

script_gap0_acon

```
#!/bin/tcsh

awk 'FNR==2 {print FILENAME,$0}' pb.acon_iso_* | awk '{print $1 "   " $5*1000}' > pb.acon_iso_gap0
sed -i 's/pb.acon_iso_//' pb.acon_iso_gap0
```

▶ 6th step: Solve the linearized isotropic Eliashberg equations for the critical temperature

Near T_c, $\Delta(i\omega_j) \to 0$ and the system of Eqs. (1) reduces to a linear matrix equation for $\Delta(i\omega_j)$:

$$\Delta(i\omega_j) = \sum_{j'} \frac{1}{|2j'+1|}[\lambda(\omega_j - \omega_{j'}) - \mu_c^* - \delta_{jj'} \sum_{j''} \lambda(\omega_j - \omega_{j''}) s_j s_{j''}]\Delta(i\omega_{j'}) \quad (3)$$

where $s_j = sign(\omega_j)$. The critical temperature T_c is defined as the value at which the maximum eigenvalue is close to 1.

This step can be done by starting from a file containing the Eliashberg spectral function (`pb.a2f_iso`) using the following input (only the differences with respect to the `epw1.in` file are shown) and job-script:

Note 1: In this case `ephmatXX`, `freq`, `egnv`, and ikmap files (saved in the `prefix.ephmat` directory) are not used. You can also solve the isotropic Migdal-Eliashberg equations at other temperatures starting from a file containing the Eliashberg spectral function (`pb.a2f_iso`). This procedure does not work for solving the anisotropic Migdal-Eliashberg equations.

Note 2: You only need to use one CPU if the isotropic Migdal-Eliashberg or linearized Eliashberg equations are solved starting from the Eliashberg spectral function.

```
$ sbatch job.epw2
```

job.epw2

```
#!/bin/bash
#SBATCH -J job.epw2           # Job name
#SBATCH -N 1                  # Total # of nodes
#SBATCH --ntasks-per-node 1
#SBATCH -t 01:00:00           # Run time (hh:mm:ss)
#SBATCH -A EPW-SCHOOL
#SBATCH -p small
#SBATCH --reservation=EPW-SCHOOL-06-17-2021

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

ibrun $PATHQE/bin/epw.x -nk 1 -in epw2.in > epw2.out
```

epw2.in

```
--
  ep_coupling = .false.
  elph        = .false.
  epwwrite    = .false.
  epwread     = .true.
  wannierize  = .false.
  ephwrite    = .false.

  fila2f      = 'pb.a2f_iso'
  lpade       = .false.
  lacon       = .false.
  tc_linear   = .true.          ! solve linearized ME eqn. for Tc
  tc_linear_solver  = 'power'   ! algorithm to solve Tc eigenvalue problem: 'power' OR 'lapack'

  nstemp   = 21                 ! number of temperature points
  temps    = 0.25 5.25          ! evenly spaced nstemp temperature points according to
                                ! (temps(2)-temps(1))/(nstemp-1).
```

```
Start: Solving (isotropic) linearized Eliashberg equation with solver = power

For the first Temp.    0.25  K
  Total number of frequency points nsiw(     1) =    739
  Cutoff frequency wscut =     0.1001

 Superconducting transition temp. Tc is the one which has Max. eigenvalue close to 1
 ---------------------------------------------------------------
         Temp.       Max.         nsiw     wscut    Nr. of iters
         (K)       eigenvalue    (itemp)   (eV)      to Converge
 ---------------------------------------------------------------
         0.25      4.5835979      739      0.1001        6
         0.50      3.6708097      369      0.1000        7
         ....      ........       ...      ......       ..
         5.25      0.8996288       35      0.1009       27
---------------------------------------------------------------
Finish: Solving (isotropic) linearized Eliashberg equation
```

You can extract the maximum eigenvalue as a function of temperature from the `epw2.out` using `script_max_eigenvalue` and save the date in `data_max_eigenvalue.dat` file.

```
$ ./script_max_eigenvalue
```

script_max_eigenvalue

```
#!/bin/tcsh

grep -A 25 "Max. eigenvalue" epw2.out | tail -21 | awk '{print $1 "     " $2}' > data_max_eigenvalue.dat
```

Plot `data_max_eigenvalue.dat` to obtain the T_c. The critical temperature is defined as the value for which the maximum eigenvalue is close to 1. You can use the gnuplot script below to get the graph shown in Fig. 4

```
$ gnuplot
gnuplot> set xlabel "T (K)"
gnuplot> set ylabel "Max. eigenvalue"
gnuplot> set xrange [0:6]
gnuplot> set arrow from 0,1 to  6,1 nohead lt 2 lw 1
gnuplot> plot "data_max_eigenvalue.dat" with lp ls 3 notitle
```

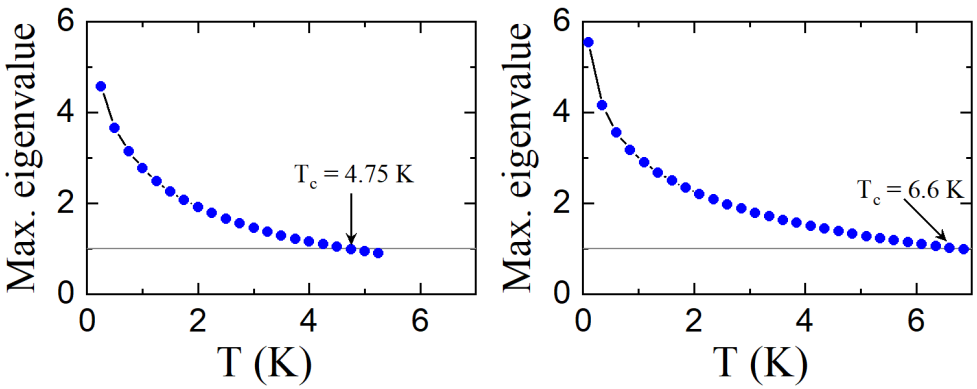

Fig. 4 Calculated maximum eigenvalue as a function of temperature. At convergence you should get something closer to the right hand-side figure.

Exercise 2

In this tutorial we are going to calculate the superconducting properties of MgB_2 by solving the anisotropic Migdal-Eliashberg equations. The theory related to this tutorial can be found in the Phys. Rev. B **87**, 024505 (2013).

Go to `exercise2`:

```
$ cd ../../exercise2
```

▶ 1st step: Run a self-consistent calculation on a homogeneous 12x12x12 **k**-point grid and a phonon calculation on a homogeneous 3x3x3 **q**-point grid using the following inputs and jobscript for MgB_2:
Note: The smearing is quite large in order to get reasonable values in subsequent calculations.

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

job.ph

```
#!/bin/bash
#SBATCH -J job.ph             # Job name
#SBATCH -N 1                  # Total # of nodes
#SBATCH --ntasks-per-node 56
#SBATCH -t 01:00:00           # Run time (hh:mm:ss)
#SBATCH -A EPW-SCHOOL
#SBATCH -p small
#SBATCH --reservation=EPW-SCHOOL-06-17-2021

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

ibrun $PATHQE/bin/pw.x -nk 56 -in scf.in > scf.out
ibrun $PATHQE/bin/ph.x -nk 56 -in ph.in > ph.out
```

scf.in

```
--
&control
 calculation  = 'scf'
 restart_mode = 'from_scratch',
 prefix       = 'mgb2',
 pseudo_dir   = '../',
 outdir       = './',
/
&system
  ibrav        = 4,
  celldm(1)    = 5.8260252227888,
  celldm(3)    = 1.1420694129095,
  nat          = 3,
  ntyp         = 2,
  ecutwfc      = 40
  smearing     = 'mp'
  occupations  = 'smearing'
  degauss      = 0.05
/
&electrons
  diagonalization = 'david'
  mixing_mode     = 'plain'
  mixing_beta     = 0.7
  conv_thr        = 1.0d-9
/
ATOMIC_SPECIES
  Mg  24.305  Mg.pz-n-vbc.UPF
  B   10.811  B.pz-vbc.UPF
ATOMIC_POSITIONS crystal
  Mg       0.000000000   0.000000000   0.000000000
  B        0.333333333   0.666666667   0.500000000
  B        0.666666667   0.333333333   0.500000000
K_POINTS AUTOMATIC
  12 12 12 0 0 0
```

ph.in

```
--
&inputph
  prefix   = 'mgb2',
  fildyn   = 'mgb2.dyn.xml',
  tr2_ph   =  1.0d-16
  fildvscf = 'dvscf',
  ldisp    = .true.,
  nq1 = 3,
  nq2 = 3,
  nq3 = 3
```

```
Dynamical matrices for ( 3, 3, 3)  uniform grid of q-points
(   6 q-points):
  N         xq(1)         xq(2)         xq(3)
```

```
1   0.000000000   0.000000000   0.000000000
2   0.000000000   0.000000000   0.291867841
3   0.000000000   0.384900179   0.000000000
4   0.000000000   0.384900179   0.291867841
5   0.333333333   0.577350269   0.000000000
6   0.333333333   0.577350269   0.291867841
```

▶ 2nd step: Gather the `.dyn`, `.dvscf`, and `patterns` files into a new `save` directory using the pp.py python script.

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

The script will ask you to provide the `prefix` of your calculation (here "mgb2").

▶ 3rd step: Do a non self-consistent calculation on a 6×6×6 **uniform** and Γ-**centered grid between [0,1[in crystal coordinates** and an EPW calculation for the anisotropic superconducting properties using the following inputs and jobscript:

Note 1: The homogeneous $\mathbf{k}$ grid for the non self-consistent calculations can be generated using the script `kmesh.pl`

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

Note 2: A non self-consistent calculation requires the charge density found from a previous self-consistent run with `pw.x`. In the jobscript `job.epw1` you can see that a self-consistent calculation is run first with the same `scf.in` file used in the `phonon` directory. Alternatively, one can make the `mgb2.save` directory and copy there the files from `phonon/mgb2.save` For this in `job.epw1` you need to comment the line

```
#ibrun $PATHQE/bin/pw.x -nk 8 -in scf.in > scf.out
```

and uncomment the following three lines

```
mkdir mgb2.save
cp ../phonon/mgb2.save/charge-density.dat mgb2.save/
cp ../phonon/mgb2.save/data-file-schema.xml mgb2.save/
```

Note 3: EPW calculations with `ephwrite = .true.` require that the fine $\mathbf{k}$ or $\mathbf{q}$ grids are commensurate, i.e., `nkf1, nkf2, nkf3` to be multiple of `nqf1, nqf2, nqf3`.

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

job.epw1

```
#!/bin/bash
#SBATCH -J job.epw1           # Job name
#SBATCH -N 1                  # Total # of nodes
#SBATCH --ntasks-per-node 56
#SBATCH -t 01:00:00           # Run time (hh:mm:ss)
#SBATCH -A EPW-SCHOOL
#SBATCH -p small
#SBATCH --reservation=EPW-SCHOOL-06-17-2021

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

ibrun $PATHQE/bin/pw.x -nk 56 -in scf.in > scf.out
#alternatively to re-run a scf calculation copy files from ../phonon/mgb2.save
#mkdir mgb2.save
#cp ../phonon/mgb2.save/charge-density.dat mgb2.save/
#cp ../phonon/mgb2.save/data-file-schema.xml mgb2.save/

ibrun $PATHQE/bin/pw.x -nk 56 -in nscf.in > nscf.out
ibrun $PATHQE/bin/epw.x -nk 56 -in epw1.in > epw1.out
```

nscf.in

```
&control
  calculation  = 'nscf'
  prefix       = 'mgb2',
  pseudo_dir   = '../',
  outdir       = './',
/
&system
  ibrav        = 4,
  celldm(1)    = 5.8260252227888,
  celldm(3)    = 1.1420694129095,
  nat          = 3,
  ntyp         = 2,
  ecutwfc      = 40
  smearing     = 'mp'
  occupations  = 'smearing'
  degauss      = 0.05
/
&electrons
  diagonalization = 'david'
  mixing_mode     = 'plain'
  mixing_beta     = 0.7
  conv_thr        = 1.0d-9
/
ATOMIC_SPECIES
  Mg  24.305  Mg.pz-n-vbc.UPF
  B   10.811  B.pz-vbc.UPF
ATOMIC_POSITIONS crystal
  Mg       0.000000000   0.000000000   0.000000000
  B        0.333333333   0.666666667   0.500000000
  B        0.666666667   0.333333333   0.500000000
K_POINTS crystal
  216
  0.00000000  0.00000000  0.00000000  4.629630e-03
  0.00000000  0.00000000  0.16666667  4.629630e-03
 ...
```

epw1.in

```
--
&inputepw
  prefix      = 'mgb2',
  amass(1)    = 24.305,
  amass(2)    = 10.811
  outdir      = './'
  dvscf_dir   = '../phonon/save'  ! directory where .dyn, .dvscf and prefix.phsave/patterns.xx.yy
                                  ! files obtained from phonon calculation are stored

  ep_coupling = .true.          ! run e-ph coupling calculation
  elph        = .true.          ! calculate e-ph coefficients
  epwwrite    = .true.          ! write e-ph matrices in the Wann representation
  epwread     = .false.         ! read e-ph matrices from the 'prefix.epmatwp' file

  etf_mem     =  1              ! more IO (slower) but less memory is required

  wannierize  = .true.                  ! calculate Wannier functions using W90 library
  nbndsub     = 5                       ! number of Wannier functions to utilize

  num_iter      = 500
  dis_froz_max  = 8.8
  proj(1)       = 'B:pz'
  proj(2)       = 'f=0.5,1.0,0.5:s'
  proj(3)       = 'f=0.0,0.5,0.5:s'
  proj(4)       = 'f=0.5,0.5,0.5:s'

  iverbosity    = 2             ! 2 = verbose output for the SC part

  fsthick       = 0.2           ! Fermi window thickness [eV]
  degaussw      = 0.05          ! smearing in the energy-conserving delta functions in [eV]

  fermi_plot    = .true.        ! write files to plot Fermi surface
  ephwrite      = .true.        ! write ephmatXX, egnv, freq, and ikmap files in 'prefix.ephmat' directory
```

```
  eliashberg    = .true.        ! calculate Eliashberg spectral function

  laniso    = .true.            ! solve anisotropic ME eqs.
  limag     = .true.            ! solve ME eqs. imaginary axis
  lpade     = .true.            ! solve ME eqs. on real axis using Pade approximants

  nsiter    = 500               ! number of self-consistent iterations when solving ME eqs.
  conv_thr_iaxis = 1.0d-3       ! convergence threshold for solving ME eqs. on imaginary axis
  wscut     = 0.5               ! upper limit over Matsubara freq. summation in ME eqs on imag. axis [eV]
  muc       = 0.05              ! effective Coulomb potential used in ME eqs.

  nstemp    = 3                 ! number of temperature points at which the ME eqs. are solved
  temps     = 10  20            ! even space mode: step between points is (temps(2)-temps(1))/(nstemp-1)

  nk1       = 6                 ! dimensions of the coarse electronic grid
  nk2       = 6
  nk3       = 6

  nq1       = 3                 ! dimensions of the coarse phonon grid
  nq2       = 3
  nq3       = 3

  mp_mesh_k = .true.            ! use irreduciable electronic fine mesh
  nkf1      = 40
  nkf2      = 40                ! dimensions of the fine electronic grid
  nkf3      = 40

  nqf1      = 20
  nqf2      = 20                ! dimensions of the fine phonon grid
  nqf3      = 20
/
```

With the above input, we are instructing EPW to:

- Fourier-transform the electron-phonon matrix elements from a coarse 6×6×6 to a dense 40×40×40 $\mathbf{k}$-point grid and from a coarse 3×3×3 to a dense 20×20×20 $\mathbf{q}$-point grid.

```
Using uniform q-mesh:    20   20   20
Size of q point mesh for interpolation:       8000
Using uniform MP k-mesh:    40   40   40
Size of k point mesh for interpolation:       6468
Max number of k points per pool:               116
```

- Pre-compute the $\mathbf{q}$-points that fall within the `fsthick`. If at a specific $\mathbf{q}$-point at least one $\mathbf{k}+\mathbf{q}$ eigenvalue falls within the user-defined `fsthick`, then the $\mathbf{q}$-point is selected.

```
Number selected, total            100            105
Number selected, total            200            205
......
Number selected, total           7800           7956
We only need to compute     7844 q-points
```

- Write on disk in the `mgb2.ephmat` directory the: (1) `ephmatXX` files (one per CPU) containing the electron-phonon matrix elements within the Fermi window (`fsthick`) on the dense $\mathbf{k}$ and $\mathbf{q}$ grids, (2) `freq` file containing the phonon frequencies on the dense $\mathbf{q}$ grid, (3) `egnv` file containing the eigenvalues within the Fermi window on the dense $\mathbf{k}$ grid, and (4) `ikmap` file containing the index of the k-points on the dense (irreducible) grid within the Fermi window. All these files are produced by setting `ephwrite = .true.`. These files are unformatted and required for solving the anisotropic Migdal-Eliashberg equations.

```
Nr. of irreducible k-points on the uniform grid:       3234

Finish mapping k+sign*q onto the fine irreducibe k-mesh and writing .ikmap file

Nr irreducible k-points within the Fermi shell =       449 out of      3234

Progression iq (fine) =        100/      7844
Progression iq (fine) =        200/      7844
....
....
Progression iq (fine) =       7800/      7844
             Fermi level (eV) =    0.747114140623985D+01
DOS(states/spin/eV/Unit Cell) =    0.333022707682310D+00
       Electron smearing (eV) =    0.500000000000000D-01
            Fermi window (eV) =    0.200000000000000D+00

Finish writing .ephmat files
```

- Write the Fermi surfaces files `mgb2.fs_YY.cube` (YY = band index within the `fsthick`) by setting `fermi_plot = .true.`. These files can be visualized with VESTA.

```
Fermi surface calculation on fine mesh
             Fermi level (eV) =   7.471141
      3   bands within the Fermi window
```

- Calculate the isotropic and anisotropic electron-phonon coupling strength by setting the keywords `eliashberg = .true.` in the EPW input file.

  The anisotropic electron-phonon coupling strength takes the following form:

$$\lambda_{n\mathbf{k},m\mathbf{k+q}}(\omega_j) = N_\mathrm{F} \sum_\nu \frac{2\omega_{\mathbf{q}\nu}}{\omega_j^2 + \omega_{\mathbf{q}\nu}^2} |g_{mn\nu}(\mathbf{k},\mathbf{q})|^2 \quad (4)$$

  The interaction function defined in Eq. (4) for $\omega_j = 0$ represents the strength for a pair of electrons at states $\mathbf{k}$ and $-\mathbf{k}$ scattering to $\mathbf{k}'$ and $-\mathbf{k}'$ by phonon exchange.

  The band- and wavevectron-dependent electron-phonon coupling strength $\lambda_{n\mathbf{k}}(\omega_j)$ is defined as:

$$\lambda_{n\mathbf{k}}(\omega_j) = \sum_m \int \frac{d\mathbf{q}}{\Omega_\mathrm{BZ}} \frac{\delta(\epsilon_{m\mathbf{k+q}} - \epsilon_\mathrm{F})}{N_\mathrm{F}} \lambda_{n\mathbf{k},m\mathbf{k+q}}(\omega_j) \quad (5)$$

  One can use the function defined in Eq. (5) for $\omega_j = 0$ to calculate the variation of the electronphonon interaction on the Fermi surface.

  The isotropic electron-phonon coupling strength takes the form:

$$\lambda(\omega_j) = \sum_n \int \frac{d\mathbf{k}}{\Omega_\mathrm{BZ}} \frac{\delta(\epsilon_{n\mathbf{k}} - \epsilon_\mathrm{F})}{N_\mathrm{F}} \lambda_{n\mathbf{k}}(\omega_j) \quad (6)$$

  The standard electronphonon coupling strength λ in the literature corresponds to $\omega_j = 0$ in the above notation.

  The isotropic Eliashberg spectral function takes the following form:

$$\alpha^2F(\omega) = \frac{1}{N_\mathrm{F}} \sum_{mn\nu} \int \frac{d\mathbf{k}}{\Omega_\mathrm{BZ}} \int \frac{d\mathbf{q}}{\Omega_\mathrm{BZ}} |g_{mn\nu}(\mathbf{k},\mathbf{q})|^2 \, \delta(\omega - \omega_{\mathbf{q}\nu}) \delta(\epsilon_{n\mathbf{k}} - \epsilon_\mathrm{F}) \delta(\epsilon_{m\mathbf{k+q}} - \epsilon_\mathrm{F}) \quad (7)$$

- Solve the anisotropic Migdal-Eliashberg equations on the imaginary frequency axis by setting the keywords `eliashberg = .true.`, `laniso = .true.`, and `limag = .true.` in the EPW input file. The equations are solved self-consistently for each temperature value specified in the input file. The calculation at each temperature ends when either the converge threshold (`conv_thr_iaxis`) or the maximum number of iterations (`nsiter`) is reached.

  Note 1: If at a specific temperature the maximum number of iterations is reached without achieving convergence, the code will stop and not move to the next temperature in the list.

  Note 2: Because the electron-phonon matrix elements do not depend on the temperature at which the Migdal-Eliashberg equations are solved, they can be reused in subsequent EPW calculations at different temperatures. This is the reason why `ephmatXX` files are saved in the `mgb2.ephmat` directory.

  The anisotropic Migdal-Eliashberg equations take the following form:

$$Z_{n\mathbf{k}}(i\omega_j) = 1 + \frac{\pi T}{\omega_j N_\mathrm{F}} \sum_{mj'} \int \frac{d\mathbf{q}}{\Omega_\mathrm{BZ}} \frac{\omega_{j'}}{\sqrt{\omega_{j'}^2 + \Delta_{m\mathbf{k}+\mathbf{q}}^2(i\omega_{j'})}} \times \lambda_{n\mathbf{k},m\mathbf{k}+\mathbf{q}}(\omega_j - \omega_{j'}) \delta(\epsilon_{m\mathbf{k}+\mathbf{q}} - \epsilon_\mathrm{F})$$

$$Z_{n\mathbf{k}}(i\omega_j)\Delta_{n\mathbf{k}}(i\omega_j) = \frac{\pi T}{N_\mathrm{F}} \sum_{mj'} \int \frac{d\mathbf{q}}{\Omega_\mathrm{BZ}} \frac{\Delta_{m\mathbf{k}+\mathbf{q}}(i\omega_{j'})}{\sqrt{\omega_{j'}^2 + \Delta_{m\mathbf{k}+\mathbf{q}}^2(i\omega_{j'})}} \times \left[\lambda_{n\mathbf{k},m\mathbf{k}+\mathbf{q}}(\omega_j - \omega_{j'}) - \mu_\mathrm{c}^*\right] \delta(\epsilon_{m\mathbf{k}+\mathbf{q}} - \epsilon_\mathrm{F}), \quad (8)$$

  where $\lambda_{n\mathbf{k},m\mathbf{k}+\mathbf{q}}(\omega_j)$ is the anisotropic electron-phonon coupling strength. The semiempirical Coulomb parameter μ_c^* is provided as an input varible `muc` in the EPW calculation.

```
================================================================
Solve anisotropic Eliashberg equations
================================================================
......
Electron-phonon coupling strength =    0.7125181

Estimated Allen-Dynes Tc =    34.592806 K for muc =    0.05000

Estimated w_log in Allen-Dynes Tc =    57.716675 meV

Estimated BCS superconducting gap =     5.246520 meV

temp(  1) =     10.00000 K

Solve anisotropic Eliashberg equations on imaginary-axis

Total number of frequency points nsiw(     1) =     92
Cutoff frequency wscut =     0.5008


Size of allocated memory per pool: ~=    0.0756 Gb
iter      ethr          znormi      deltai [meV]
 1   2.614393E+00   1.698390E+00   6.136877E+00
 2   7.335688E-02   1.695460E+00   6.435702E+00
 ......
14   7.888682E-04   1.688745E+00   7.144835E+00
Min. value of superconducting gap =     0.000000 meV
Convergence was reached in nsiter =     14
```

- Perform the analytic continuation of the solutions along the imaginary frequency axis to the real frequency axis by using Padé approximants (`lpade = .true.`). Note the analytic continuation with the iterative procedure (`lacon = .true.`) is not performed since this is very expensive computationally in the anisotropic case (hours to days).

```
Pade approximant of anisotropic Eliashberg equations from imaginary-axis to real-axis
Cutoff frequency wscut =     0.5000

pade Re[znorm] [eV] Re[delta] [eV]
82   1.725924E+00   6.598036E+00

Convergence was reached for N =     82 Pade approximants
```

The calculation of superconducting properties will be accompanied by significant I/O. In the following we will describe various physical quantities saved in the output files and how to process them. We will use `XX` in the name of the output files to indicate the temperature at which the equations are solved.

▶ 4th step: Plot the isotropic and anisotropic electron-phonon coupling strength

`mgb2.lambda_pairs`, `mgb2.lambda_k_pairs`, and `mgb2.a2f` files are generated by setting `eliashberg = .true.`

`mgb2.lambda_pairs` file contains the anisotropic electron-phonon coupling strength $\lambda_{n\mathbf{q},m\mathbf{k+q}}(0)$ on the Fermi surface.

`mgb2.lambda_k_pairs` file contains the band- and wavevector-dependent anisotropic electron-phonon coupling strength $\lambda_{n\mathbf{k}}(0)$ on the Fermi surface.

`mgb2.a2f` file contains the isotropic Eliashberg spectral function $\alpha^2F(\omega)$ and cumulative electron-phonon coupling strength as a function of frequency ω (meV) for different phonon smearing values (see the end of the file for information about the smearing).

You can use the following gnuplot scripts to plot. You should get something similar to Fig. 5.

```
$ gnuplot
gnuplot> set xlabel "lambda_(nk,mk+q)(0)"
gnuplot> set xrange [0:4]
gnuplot> set yrange [0:1.1]
gnuplot> plot "mgb2.lambda_pairs" w l lw 2 lt rgb "black" notitle

gnuplot> set xlabel "lambda_(nk)(0)"
gnuplot> set xrange [0:1.5]
gnuplot> set yrange [0:1.1]
gnuplot> plot "mgb2.lambda_k_pairs" w l lw 2 lt rgb "black" notitle

# First, put # infront of the 1st line and last 7 lines of mgb2.a2f,
# otherwise gnuplot does not work
gnuplot> set xlabel "w (meV)"
gnuplot> set xrange [0:110]
gnuplot> set yrange [0:1.5]
gnuplot> plot "mgb2.a2f" w l lw 2 lt rgb "black" title "a2f", \
> "" u ($1):($12) w l lw 2 lt rgb "red" title "lambda"
```

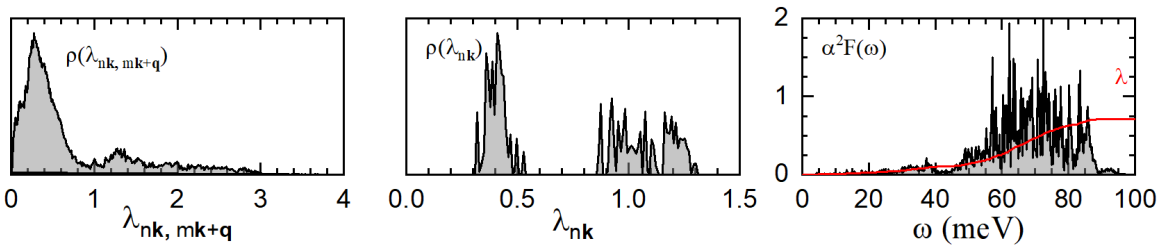

Fig. 5 Left: The anisotropic electron-phonon coupling strength $\lambda_{n\mathbf{q},m\mathbf{k}+\mathbf{q}}(0)$ (from `mgb2.lambda_pairs`). Middle: The anisotropic electron-phonon coupling strength $\lambda_{n\mathbf{k}}(0)$ on the Fermi surface (from `mgb2.lambda_k_pairs`). Right: The isotropic Eliashberg spectral function $\alpha^2F(\omega)$ (columns 1:2 from `mgb2.a2f`) and integrated electron-phonon coupling strength λ (columns 1:12 from `mgb2.a2f`).

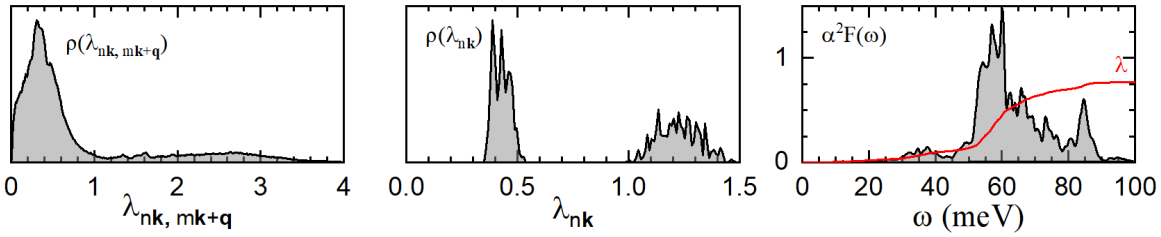

Fig. 6 At convergence you should get something close to this figure.

▶ 5th step: Plot the superconducting gap along the imaginary frequency axis and the real frequency axis.

> `mgb2.imag_aniso_XX` files are generated by setting `eliashberg = .true.`, `laniso = .true.`, and `limag = .true.`. Each file contains 5 columns: the frequency $i\omega_j$ (eV) along the imaginary axis, the Kohn-Sham eigenvalue $\epsilon_{n\mathbf{k}}$ (eV) relative to the Fermi level, the quasiparticle renormalization $Z_{n\mathbf{k}}(i\omega_j)$, the superconducting gap $\Delta_{n\mathbf{k}}(i\omega_j)$ (eV), and the quasiparticle renormalization $Z^N_{n\mathbf{k}}(i\omega_j)$ in the normal state.
>
> `mgb2.pade_aniso_XX` files are generated by setting `lpade = .true.`. Each file contains 6 columns: the energy ω (eV) along the real axis, the Kohn-Sham eigenvalue $\epsilon_{n\mathbf{k}}$ (eV) relative to the Fermi level, the real part of the quasiparticle renormalization $\mathrm{Re}Z_{n\mathbf{k}}(\omega)$, the imaginary part of the quasiparticle renormalization $\mathrm{Im}Z_{n\mathbf{k}}(\omega)$, the real part of the superconducting gap $\mathrm{Re}\Delta_{n\mathbf{k}}(\omega)$ (eV), and the imaginary part of the superconducting gap $\mathrm{Im}\Delta_{n\mathbf{k}}(\omega)$ (eV).
>
> `mgb2.acon_aniso_XX` files could also be generated by setting `lacon = .true.`. These files will contain similar information as `mgb2.pade_aniso_XX`.

You can use the following gnuplot scripts to plot. You should get something similar to Fig. 7 at 10 K.

```
$ gnuplot
gnuplot> set xlabel "iw (meV)"
gnuplot> set ylabel "Delta_nk (meV)"
gnuplot> set xrange [0:180]
gnuplot> set yrange [0:15]
gnuplot> plot "mgb2.imag_aniso_010.00" u ($1*1000):($4*1000) with points pt 7 \
> notitle
```

```
gnuplot> set xlabel "w (meV)"
gnuplot> set ylabel "Delta_nk (meV)"
gnuplot> set xrange [0:180]
gnuplot> set yrange [-20:45]
gnuplot> plot "mgb2.pade_aniso_010.00" u ($1*1000):($5*1000) with points pt 7 \
> notitle
```

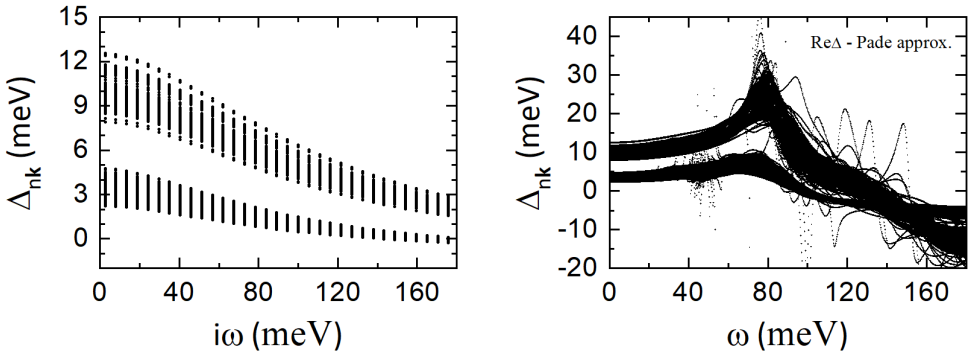

Fig. 7 Left: Superconducting gap along the imaginary axis (columns 1:4 from `mgb2.imag_aniso_010.00`). Right: Superconducting gap along the real axis (columns 1:5 from `mgb2.pade_aniso_010.00` - this file is about 70MB).

The fine $\mathbf{k}$ and $\mathbf{q}$ point grids need to be much denser for real calculations. At convergence you should get:

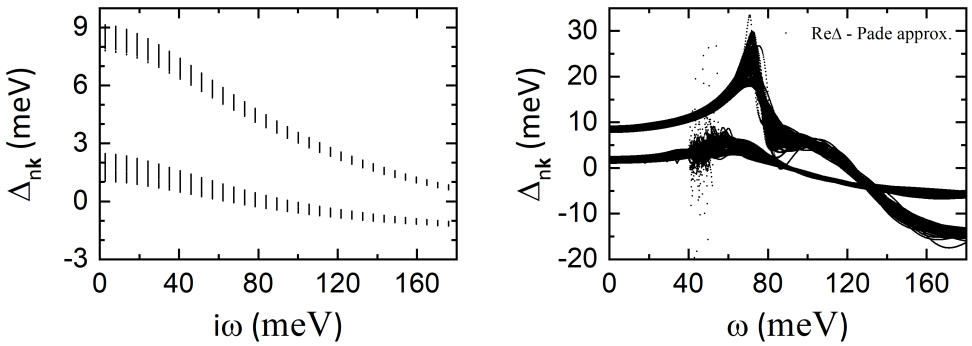

Fig. 8 Figure adapted from Phys. Rev. B **87**, 024505 (2013). (Note: Only about half of the points are shown.)

▶ 6th step: Do a restart calculation to compute the superconducting gap function on the imaginary axis at other temperatures.

The restart is done by reading the superconducting gap and renormalization function on the imaginary axis at 10 K from file `mgb2.imag_aniso_010.00` using the following input (only the differences with respect to the `epw1.in` file are shown) and jobscript:

```
$ sbatch job.epw2
```

```
                                                                    job.epw2
#!/bin/bash
#SBATCH -J job.epw2           # Job name
```

```
#SBATCH -N 1                  # Total # of nodes
#SBATCH --ntasks-per-node 56
#SBATCH -t 01:00:00           # Run time (hh:mm:ss)
#SBATCH -A EPW-SCHOOL
#SBATCH -p small
#SBATCH --reservation=EPW-SCHOOL-06-17-2021

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

ibrun $PATHQE/bin/epw.x -nk 56 -in epw2.in > epw2.out
```

```
 --                                                                                                    epw2.in
 ep_coupling = .false.
 elph        = .false.
 epwwrite    = .false.
 epwread     = .true.
 wannierize  = .false.
 fermi_plot  = .false.
 ephwrite    = .false.
 imag_read   = .true.                  ! superdconducting gap at temps(1) is read from file

 temps       = 10 25 30 35 40 45 50 55  ! list of temeratures in [K]
```

▶ 7th step: Plot the leading edge of the superconducting gap as a function of temperature.

You should get the following graph by plotting the data from all `mgb2.imag_aniso_gap0_XX` files. Use the following `gnuplot` script.

```
$ gnuplot
gnuplot> set xlabel "T (K)"
gnuplot> set ylabel "Delta_nk (meV)"
gnuplot> set xrange [0:60]
gnuplot> set yrange [0:15]
gnuplot> plot "mgb2.imag_aniso_gap0_010.00" w l lw 1 lt rgb "black" notitle, \
> "mgb2.imag_aniso_gap0_015.00" w l lw 1 lt rgb "black" notitle, \
> "mgb2.imag_aniso_gap0_020.00" w l lw 1 lt rgb "black" notitle, \
> "mgb2.imag_aniso_gap0_025.00" w l lw 1 lt rgb "black" notitle, \
> "mgb2.imag_aniso_gap0_030.00" w l lw 1 lt rgb "black" notitle, \
> "mgb2.imag_aniso_gap0_035.00" w l lw 1 lt rgb "black" notitle, \
> "mgb2.imag_aniso_gap0_040.00" w l lw 1 lt rgb "black" notitle, \
> "mgb2.imag_aniso_gap0_045.00" w l lw 1 lt rgb "black" notitle, \
> "mgb2.imag_aniso_gap0_050.00" w l lw 1 lt rgb "black" notitle, \
> "mgb2.imag_aniso_gap0_055.00" w l lw 1 lt rgb "black" notitle
```

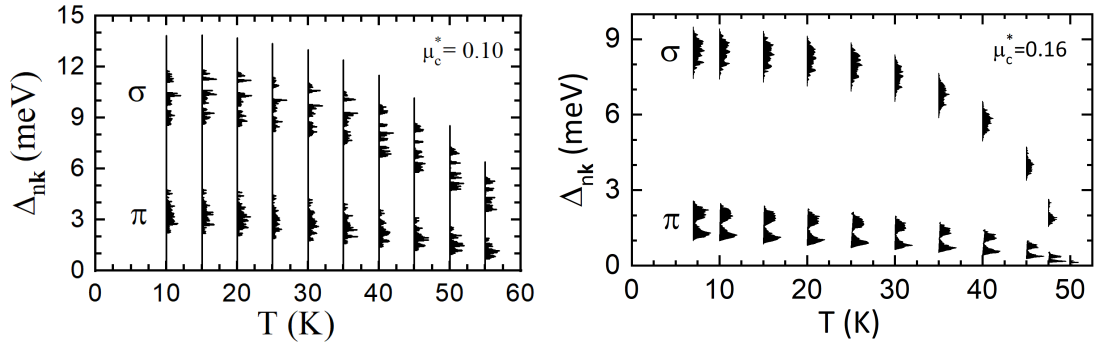

Fig. 9 Calculated anisotropic superconducting gap of MgB_2 on the Fermi surface as a function of temperature. At convergence you should get the right hand-side figure adapted from Phys. Rev. B **87**, 024505 (2013). (Note: the heights of the histograms are multiplied by a factor of 2 while plotting for visibility.)

▶ 8th step: Plot the superconducting quasiparticle density of states.

The quasiparticle density of states (DOS) in the superconducting state relative to the DOS in the normal state is given by:

$$\frac{N_S(\omega)}{N_{\rm F}} = \sum_n \int \frac{d\mathbf{k}}{\Omega_{\rm BZ}} \frac{\delta(\epsilon_{n\mathbf{k}} - \epsilon_{\rm F})}{N_{\rm F}} {\rm Re}\left[\omega / \sqrt{\omega^2 - \Delta^2_{n\mathbf{k}}(\omega)}\right] \tag{9}$$

`mgb2.qdos_XX` files contain the quasiparticle density of states in the superconducting state relative to the density of states in the normal state $N_S(\omega)/N_{\rm F}$ as a function of frequency (eV) at various `XX` temperatures.

Use the following gnuplot script to plot `mgb2.qdos_010.00`. You should get something similar to Fig. 10 (left) at 10 K:

```
$ gnuplot
gnuplot> set xlabel "w (meV)"
gnuplot> set ylabel "N_s(w)/N_F"
gnuplot> set xrange [0:15]
gnuplot> set yrange [0:2.25]
gnuplot> plot "mgb2.qdos_010.00" u ($1*1000):($2/0.3330227) w l lw 2 lt \
> rgb "black" notitle
```

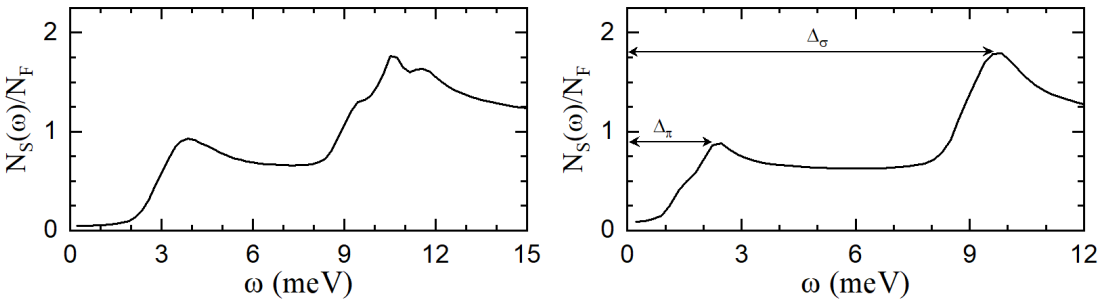

Fig. 10 Calculated $N_S(\omega)/N_{\rm F}$ as a function of frequency at 10 K. At convergence you should get something closer to the right hand-side figure. (Note: the second column of `mgb2.qdos_XX` should be divided by the value of DOS from the `epw1.out`).

▶ 9th step: Try to increase the fine grids and see if you can get a result closer to convergence. Note that if either $\mathbf{k}$ or $\mathbf{q}$ is changed you need to obtain new `ephmatXX`, `egnv`, `freq`, and `ikmap` files (saved in the `mgb2.ephmat` directory).

▶ 10th step: Check the effect of the Coulomb pseudopotential μ_c^* on the superconducting gap and the critical temperature by varying the input variable `muc`. For this step you can re-use the files saved in the `mgb2.ephmat` directory.

How to plot the superconducting gap on the Fermi surface with VESTA:

1. Plot Fermi surface (FS)

   `mgb2.fs_YY.cube` (YY = band index within the `fsthick`) files were generated by setting `fermi_plot = .true.` in `epw1.in`. Each file contains the energy eigenvalues relative to the Fermi level, and can be visualized with VESTA.

   To visualize, open `mgb2.fs_1.cube` with VESTA and then import `mgb2.fs_2.cube`, ... files one-by-one as follows:

   Edit → Edit Data → Volumetric Data → Import (under Isosurface) → Choose: Multiply to current data

   Uncheck: Style → Structural Models → Show models

   Set: Properties → Isosurfaces → Isosurface level: 0

   Set: Properties → Sections → Opacity of drawn sections(%): 0

2. Color the FS based on the superconducting gap values at a specific temperature (e.g., 10.0 K)

   Import: mgb2.imag_aniso_gap0_010.00_1.cube, mgb2.imag_aniso_gap0_010.00_2.cube, ... files one-by-one on FS as the following;

   Edit → Edit Data → Volumetric Data → Import (under Surface coloring) → Choose: Add to current data

You should get the following plot at 10 K

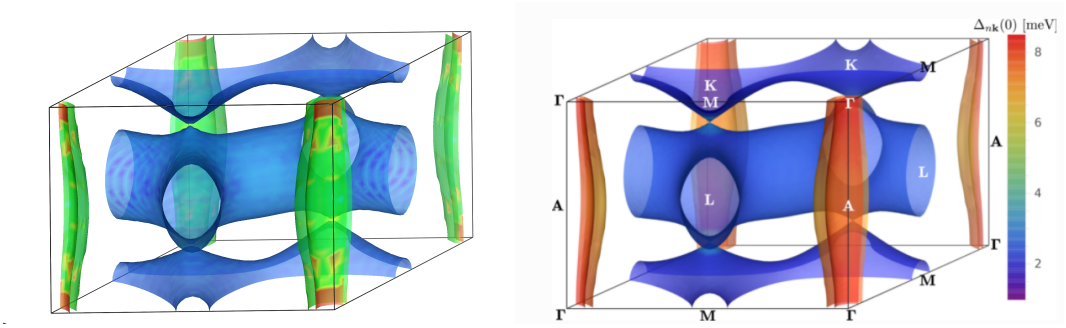

Fig. 11 Calculated superconducting gap of MgB_2 on the Fermi surface at 10 K. At convergence you should get the right hand-side figure adapted from Comp. Phys. Comm. **209**, 116 (2016).

Notes on input variables:

- `ephwrite = .true.` does not work with random $\mathbf{k}$ or $\mathbf{q}$ grids and requires `nkf1, nkf2, nkf3` to be multiple of `nqf1, nqf2, nqf3`.
- `mp_mesh_k = .true.` specifies that only the irreducible points for the dense $\mathbf{k}$ grid are used. This significantly decreases the computational cost when solving the Migdal-Eliashberg equations.
- If the Migdal-Eliashberg equations are solved in a separate run from the one in which the `ephmatXX`, `freq`, `egnv`, and `ikmap` files saved in `prefix.ephmat` directory were generated, the code requires to use the same number of CPUs as the number of `ephmatXX` files. If you forget this the code will stop with a message asking to use `npool` equal to the number of `ephmatXX` files.
- `lpade = .true.` requires `limag = .true.`
- `lacon = .true.` requires both `limag = .true.` and `lpade = .true.`.
- `wscut` gives the upper limit (in eV) of the summation over the Matsubara frequencies on the imaginary axis in the Migdal-Eliashberg equations (`limag = .true.`). Note that the input variable `wscut` is ignored if the number of frequency points is given using the input variable `nswi`. In this case, the number of frequency points in the summation is the same irrespective of the temperature.
- `temps = t1 t2 t3 ...` define the list of temperatures at which the Migdal-Eliashberg equations are evaluated. Note that an evenly spaced temperature grids can also be defined using `nstemp, temps = min.temp max.temp` input variables.
- If temperatures larger than the critical temperature T_c estimated using the Allen-Dynes formula are specified in the input file a warning message is written in the output file. The code may stop when such a temperature is reached if the Migdal-Eliashberg equations do not have a solution at that point.
- `imag_read` works if `limag = .true.` and `laniso = .true.` and it allows the code to read from file the superconducting gap and renormalization function on the imaginary axis at specific temperature `XX` from file `prefix.imag_aniso_XX`. The temperature is specified as `temps = XX` (first temperature) in the EPW input file.
- `imag_read` can be used to: (1) solve the anisotropic Migdal-Eliashberg equations on the imaginary axis at temperatures greater than `XX` using as a starting point the superconducting gap estimated at temperature `XX`. (2) obtain the solutions of the Migdal-Eliashberg equations on the real axis with `lpade = .true.` or `lacon = .true.` starting from the imaginary axis solutions at temperature `XX`; (3) write to file the superconducting gap on the Fermi surface in cube format at temperature `XX` for `iverbosity = 2`. The generated output files are `prefix.imag_aniso_gap_XX_YY.cube`, where `YY` is the band number within the chosen energy window during the EPW calculation.

Restart options (this requires to use the same number of cores as in the original run):

1. Restart from an interrupted $\mathbf{q}$-point while writing `ephmatXX` files.

   Required files: `prefix.epmatwp`, `prefix.ukk`, `crystal.fmt`, `epwdata.fmt`, `vmedata.fmt` (or `dmedata.fmt`), `restart.fmt`, and `selecq.fmt` (`selecq.fmt` only needed if `selecqread = .true.` otherwise it will be re-created).

   Input setup:

```
ep_coupling = .true.
elph        = .true.

epwwrite    = .false.
epwread     = .true.          ! read *.epmatwp and *.fmt files

wannierize  = .false.         ! read *.ukk file
ephwrite    = .true.
```

2. Restart by reading `ephmatXX` files.

   Required files: `prefix.ephmat` directory (which contains `egnv`, `freq`, `ikmap`, `ephmatXX` files), `selecq.fmt`, and `crystal.fmt`

   Input setup:

```
ep_coupling = .false.
elph        = .false.

epwwrite    = .false.
epwread     = .true.

wannierize  = .false.
ephwrite    = .false.
```