Eliashberg function and superconducting properties

Tutorial Fri.4

Hands-on session

Exercise 1

In this example we are going to calculate the superconducting properties of fcc Pb by solving the isotropic Migdal-Eliashberg equations.
First copy the tutorial files and go in the first exercise:

```
$ wget http://epw.org.uk/uploads/School2018/Fri.4.Margine.tar
$ tar -xvf Fri.4.Margine.tar; cd tuto_Fri4/exercise1
```

▶ Make a self-consistent calculation for Pb.

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

Note: The `ecutwfc` need to be much larger for real calculations.

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

▶ Compute the vibrational properties of Pb on a coarse 3x3x3 $\mathbf{q}$-point grid.

```
--                                                                    ph.in
&inputph
  prefix   = 'lead',
  fildyn   = 'lead',
  amass(1) = 207.2,
  outdir   = './'
  ldisp    = .true.,
  trans    = .true.,
```

```
 fildvscf = 'dvscf',
 nq1=3,
 nq2=3,
 nq3=3,
 tr2_ph   =  1.0d-12
```

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

The calculation should take about 2 min on 4 cores. During the run, notice the 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
```

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

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

▶ Do a non self-consistent calculation on a homogeneous 6x6x6 **uniform and Γ-centered grid between [0,1] in crystal coordinates**.

```
 &control
    calculation  ='nscf',
    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
    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
216
  0.00000000  0.00000000  0.00000000  4.629630e-03
  0.00000000  0.00000000  0.16666667  4.629630e-03
...
```

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

Since EPW does not yet support G-vector parallelization, we use **k**-point parallelization only, which means that `np` needs to be always equal to `npool`.

▶ Perform an EPW calculation.

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

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

  nbndsub     =  4
  nbndskip    =  5
  wannierize  = .true.
  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 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'

  parallel_k  = .true.
  parallel_q  = .false.

  fsthick     = 0.5 ! eV
  eptemp      = 0.075 ! K
  degaussw    = 0.05 ! eV

  ephwrite  = .true.

  eliashberg  = .true.

  liso  = .true.
  limag = .true.
  lpade = .true.
  lacon = .true.

  nsiter   = 500
  conv_thr_iaxis = 1.0d-3
  conv_thr_racon = 1.0d-3

  wscut = 0.1 ! eV

  temps(1) = 0.3
  temps(2) = 0.9
  temps(3) = 1.5
  temps(4) = 2.1
  temps(5) = 2.7
  temps(6) = 3.3
  temps(7) = 3.9
  temps(8) = 4.1
  temps(9) = 4.3
  temps(10) = 4.4
  temps(11) = 4.5

  muc     = 0.09
```

```
 mp_mesh_k = .true.
 nkf1          = 12
 nkf2          = 12
 nkf3          = 12
 nqf1          = 12
 nqf2          = 12
 nqf3          = 12

 nk1          = 6
 nk2          = 6
 nk3          = 6
 nq1          = 3
 nq2          = 3
 nq3          = 3
/
4 cartesian
  0.000000000   0.000000000   0.000000000
 -0.333333333   0.333333333  -0.333333333
  0.000000000   0.666666667   0.000000000
  0.666666667  -0.000000000   0.666666667
```

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.

```
$ mpirun -np 4 /home/nfs3/smr3191/q-e/bin/epw.x -npool 4 < epw1.in > epw1.out &
```

With the above input, we are instructing EPW to:

- Fourier-transform the electron-phonon matrix elements from a coarse 6x6x6 to a dense 12x12x12 $\mathbf{k}$-point grid and from a coarse 3x3x3 to a dense 12x12x12 $\mathbf{q}$-point grid.
- Write on disk: (1) the `lead.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) the `lead.freq` file containing the phonon frequencies on the dense $\mathbf{q}$ grid, (3) the `lead.egnv` file containing the eigenvalues within the Fermi window on the dense $\mathbf{k}$ grid, and (4) the `lead.ikmap` file containing the index of the k-points on the dense (irreducible) grid within the Fermi window. All these files were produced by setting `ephwrite=.true.`. The files are formatted and required for solving the Migdal-Eliashberg equations.
- 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 `EPW` input file.

  The isotropic Migdal-Eliashberg equations take the following form:

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

  The equations are solved self-consistently for each temperature value specified in the input file `temps(1)`, `temps(2)`, `...`.

  The isotropic electron-phonon coupling strength $\lambda(\omega_j)$ entering in Eqs. (1) is defined as:

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

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 the `.ephmatXX` files were saved.

The semiempirical Coulomb parameter μ_c^* is provided as an input varible `muc` in the EPW calculation.

- Perform the analytic continuation of the solutions along the imaginary frequency axis to the real frequency axis by using Padé approximants (`lpade = .true.`) and the iterative procedure (`lacon = .true.`).

Note:

- `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 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 crash, 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 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(1)`, `temps(2)`, ... define the temperatures at which the Migdal-Eliashberg equations are evaluated. Note that the temperatures can also be defined using `nstemp,tempsmin,tempsmax` input variables.
- If temperatures larger than the critical temperature T_c are specified in the input file, the code will stop when a first such a temperature is reached since the Migdal-Eliashberg equations have no solution at that point.

The calculation should take about 4 min to be completed. While the calculation is running, notice in the `epw.out` the different steps a full `EPW` run goes into. Once the interpolation into 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.

At the end of the calculation, you should get the following output at every given temperature (only 2 decimals are shown to fit between the page margins). Note that the number of 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`.

```
temp(  1) =      0.30000 K

Solve isotropic Eliashberg equations on imaginary-axis

Total number of frequency points nsiw (      1 ) =    616

iter =      1   error =  3.15E+00   Znormi(1) =  1.64E+00   Deltai(1) =  6.53E-04
iter =      2   error =  8.75E-02   Znormi(1) =  1.64E+00   Deltai(1) =  6.70E-04
iter =      3   error =  4.83E-02   Znormi(1) =  1.64E+00   Deltai(1) =  6.86E-04
iter =      4   error =  1.15E-02   Znormi(1) =  1.64E+00   Deltai(1) =  6.95E-04
iter =      5   error =  8.01E-03   Znormi(1) =  1.64E+00   Deltai(1) =  7.01E-04
iter =      6   error =  1.75E-02   Znormi(1) =  1.64E+00   Deltai(1) =  7.13E-04
iter =      7   error =  1.42E-03   Znormi(1) =  1.64E+00   Deltai(1) =  7.13E-04
iter =      8   error =  1.26E-04   Znormi(1) =  1.64E+00   Deltai(1) =  7.13E-04
```

```
Convergence was reached in nsiter =      8

iaxis_imag   :      0.01s CPU      0.04s WALL (       1 calls)

Pade approximant of isotropic Eliashberg equations from imaginary-axis to real-axis

pade =    264   error =  1.65E+00   Re[Znorm(1)] =  1.64E+00   Re[Delta(1)] =  7.13E-04

raxis_pade   :      1.79s CPU      1.99s WALL (       1 calls)


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

Total number of frequency points nsw =   3000

iter =      1   error =  1.10E-01   Re[Znorm(1)] =  1.64E+00   Re[Delta(1)] =  7.13E-04
iter =      2   error =  1.57E-02   Re[Znorm(1)] =  1.64E+00   Re[Delta(1)] =  7.13E-04
iter =      3   error =  7.38E-03   Re[Znorm(1)] =  1.64E+00   Re[Delta(1)] =  7.13E-04
iter =      4   error =  1.96E-03   Re[Znorm(1)] =  1.64E+00   Re[Delta(1)] =  7.13E-04
iter =      5   error =  5.25E-04   Re[Znorm(1)] =  1.64E+00   Re[Delta(1)] =  7.13E-04
Convergence was reached in nsiter =      5

raxis_acon   :     15.65s CPU     15.76s WALL (       1 calls)

itemp =   1   total cpu time :    17.8 secs
```

The calculation of superconducting properties will be accompanied by significant I/O. In the following we will describe the 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.

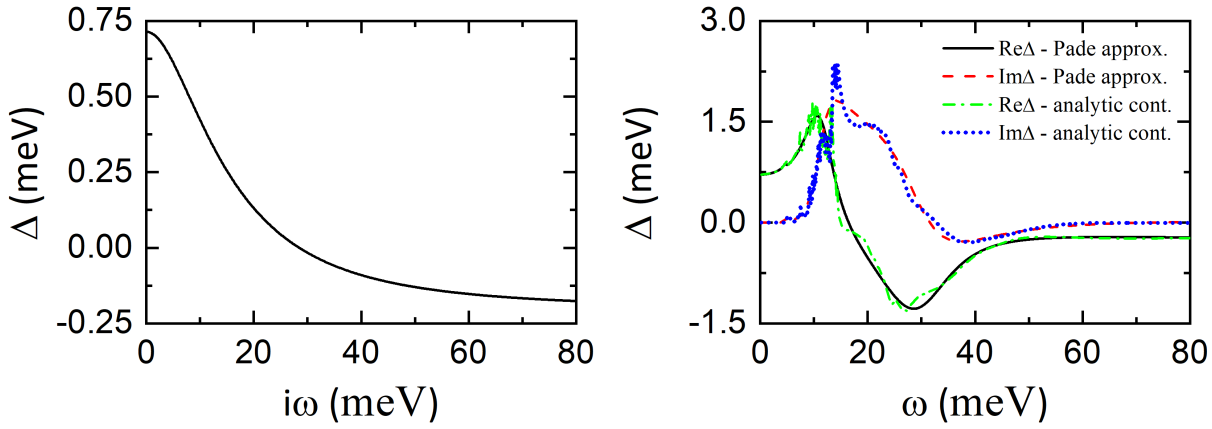
▶ Plot the superconducting gap along the imaginary frequency axis and the real frequency axis.

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

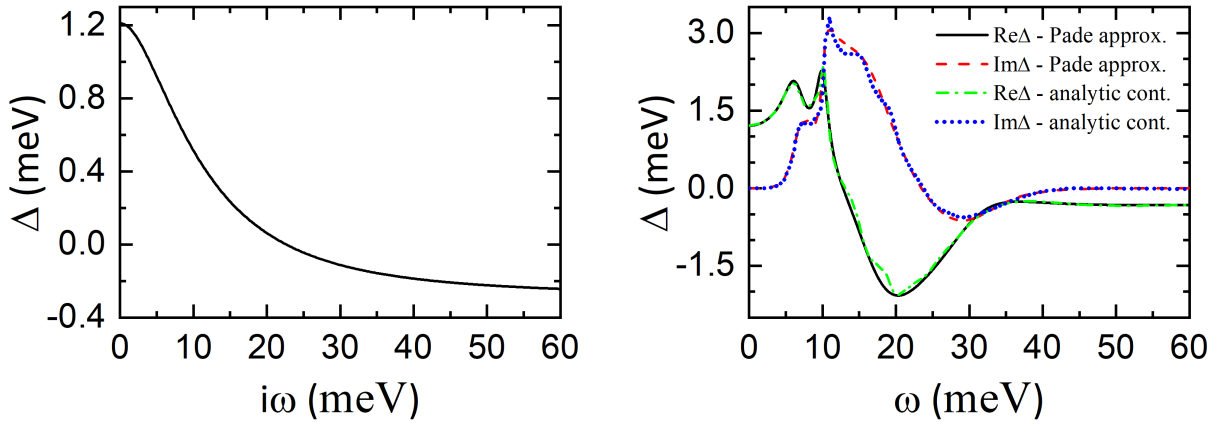
`lead.pade_iso_XX` files were generated by setting `lpade = .true.`. Each file contains 5 columns: the frequency ω (eV) along the real axis, the real part of the quasiparticle renormalization $\mathrm{Re}Z(\omega)$, the imaginary part of the quasiparticle renormalization $\mathrm{Im}Z(\omega)$, the real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

`lead.acon_iso_XX` files were generated by setting `lacon = .true.` and contain similar information as `lead.pade_iso_XX`.

You should get the following graphs at 0.3 K. The plot on the left is the superconducting gap along the imaginary axis (columns 1:3 from `lead.imag_iso_000.30`). The plot on the right is the superconducting gap on the real axis (columns 1:4 and 1:5 from `lead.pade_iso_000.30` and `lead.acon_iso_000.30`).



At convergence you should get [1]:



▶ Plot the leading edge of the superconducting gap as a function of temperature.

To obtain the `lead.imag_iso_gap0` file use the `script_gap0_imag` shell script:

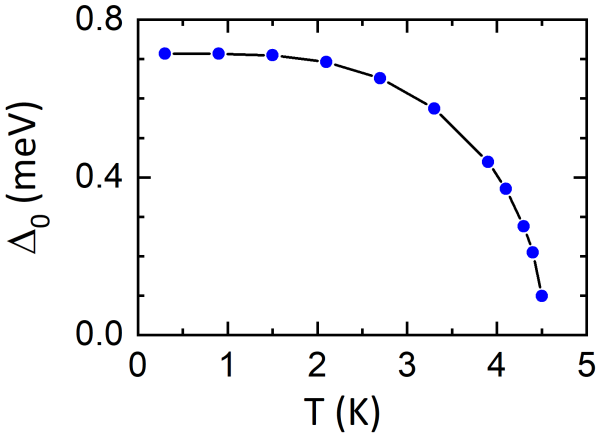
```
#!/bin/tcsh

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

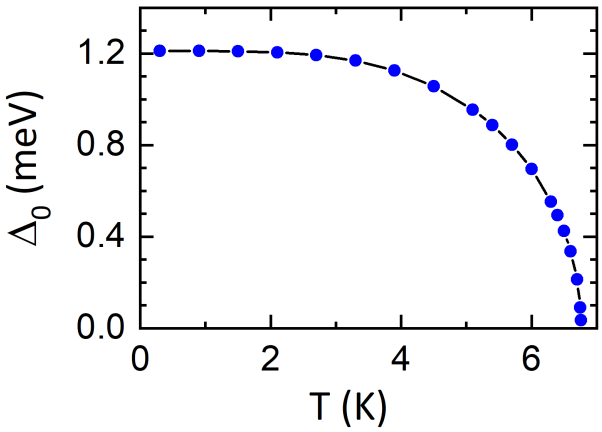
```
$ script_gap0_imag
```

You should get the following graph:

[1] Figure adapted from Margine and Giustino, Phys. Rev. B 87, 024505 (2013).



At convergence you should get a value of Δ_0 around 1.2 meV and a critical temperature T_c of about 6.8 K for $\mu_c^* = 0.1$ [2]:



You can further extract the leading edge of the superconducting gap as a function of temperature from the calculations on the real axis to compare with the results obtained on the imaginary axis. To get the `lead.pade_iso_gap0` and `lead.acon_iso_gap0` files use the `script_gap0_pade` and `script_gap0_acon` shell scripts:

```
#!/bin/tcsh

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

```
$ script_gap0_pade
```

and

```
#!/bin/tcsh

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

```
$ script_gap0_acon
```

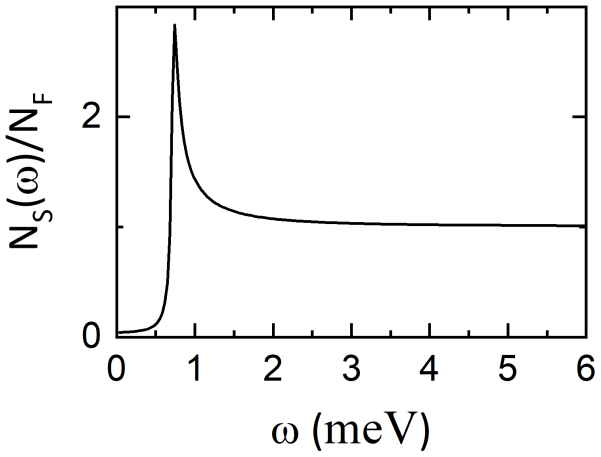
[2] Figure adapted from Margine and Giustino, Phys. Rev. B 87, 024505 (2013).

▶ Plot the superconducting quasiparticle density of states.

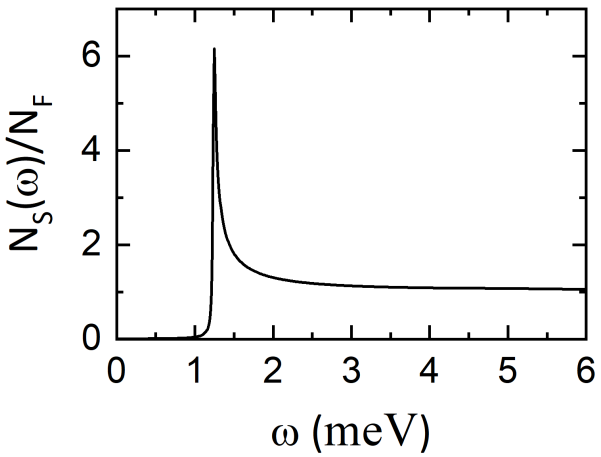
`lead.qdos_XX` files contain the quasiparticle density of states in the superconducting state relative to the density of states in the normal state $N_S(\omega)/N_{\rm F}$ as a function of frequency (eV) at various `XX` temperatures

$$\frac{N_S(\omega)}{N_{\rm F}} = {\rm Re}\left[\frac{\omega}{\sqrt{\omega^2-\Delta^2(\omega)}}\right] \tag{3}$$

You should get the following graph at 0.3 K:



At convergence you should get [3]:



▶ Solve the isotropic Migdal-Eliashberg equations starting from a file containing the Eliashberg spectral function. For this you need to have the input variables `fila2f = 'lead.a2f_iso'`, `ep_coupling = .false.`, `elph = .false.`, `wannierize = .false.`, and `ephwrite = .false.`.

Note: This procedure can only be followed when solving the isotropic Migdal-Eliashberg equations. In this case `.ephmatXX`, `.freq`, `.egnv`, and `.ikmap` files are not used.

[3] Figure adapted from Margine and Giustino, Phys. Rev. B 87, 024505 (2013).

At the end of the run, the code should have produced the `lead.a2f` and `lead.a2f_iso` files. `lead.a2f` file contains the Eliashberg spectral function as a function of frequency ω (meV) for different phonon smearing values (see the end of the file for information about the smearing). `lead.a2f_iso` file contains the Eliashberg spectral function as a function of frequency ω (meV). The second column in `lead.a2f_iso` is the Eliashberg spectral function corresponding to the first smearing in `.a2f`. The remaining (3 $\times$ number of atoms) columns in `lead.a2f_iso` contain the mode-resolved Eliashberg spectral functions (there is no specific information on which modes correspond to which atomic species).

The input file is as follow (only the differences w.r.t. `epw1.in` are shown):

```
--                                                                                    epw2.in
  ep_coupling = .false.
  elph        = .false.

  wannierize  = .false.

  ephwrite    = .false.

  fila2f      = 'lead.a2f_iso'
```

```
$ mpirun -np 1 /home/nfs3/smr3191/q-e/bin/epw.x < epw2.in > epw2.out &
```

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

Exercise 2

In this example we are going to calculate the superconducting properties of MgB_2 by solving the anisotropic Migdal-Eliashberg equations.

▶ First go to the second exercise:

```
$ cd tuto_Fri4/exercise2
```

▶ Make a self-consistent calculation for MgB_2.

```
&control                                                                              scf.in
   calculation='scf',
   restart_mode='from_scratch',
   prefix='mgb2',
   pseudo_dir = './',
   outdir='./',
   wf_collect = .true.
/
&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
8 8 8 0 0 0
```

Note: The smearing is quite large in order to get reasonable phonons in the subsequent phonon calculation.

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

▶ Compute the vibrational properties of MgB_2 on a coarse 3x3x3 $\mathbf{q}$-point grid.

ph.in
```
--
&inputph
  prefix   = 'mgb2',
  fildyn   = 'mgb2.dyn',
  amass(1) = 24.305,
  amass(2) = 10.811,
  outdir   = './'
  ldisp    = .true.,
  trans    = .true.,
  fildvscf = 'dvscf',
  nq1=3,
  nq2=3,
  nq3=3,
  tr2_ph   =  1.0d-12
```

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

The calculation should take about 7 min on 4 cores. During the run, notice the IBZ q-point grid:

```
     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
```

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

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

▶ Do a non self-consistent calculation on a homogeneous 6x6x6 **uniform and Γ-centered grid between [0,1] in crystal coordinates**.

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
...
```

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

- Perform an EPW calculation:

```
--                                                                                epw1.in
&inputepw
  prefix      = 'mgb2',
  amass(1)    = 24.305,
  amass(2)    = 10.811
  outdir      = './'

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

  etf_mem     =  1

  nbndsub     =  5
  nbndskip    =  0

  wannierize  = .true.
  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

  parallel_k  = .true.
  parallel_q  = .false.

  fsthick     = 0.4  ! eV
  eptemp      = 300  ! K
  degaussw    = 0.1 ! eV
```

```
 ephwrite    = .true.

 eliashberg  = .true.

 laniso = .true.
 limag = .true.
 lpade = .true.

 nsiter   = 500
 conv_thr_iaxis = 1.0d-3

 wscut = 0.5  ! eV

 nstemp   = 1
 tempsmin = 10.00
 tempsmax = 15.00

 muc     = 0.1

 dvscf_dir   = './save'

 nk1         = 6
 nk2         = 6
 nk3         = 6

 nq1         = 3
 nq2         = 3
 nq3         = 3

 mp_mesh_k = .true.
 nkf1 = 24
 nkf2 = 24
 nkf3 = 24

 nqf1 = 12
 nqf2 = 12
 nqf3 = 12
/
 6 cartesian
 0.0000000   0.0000000   0.0000000
 0.0000000   0.0000000   0.2918678
 0.0000000   0.3849002   0.0000000
 0.0000000   0.3849002   0.2918678
 0.3333333   0.5773503   0.0000000
 0.3333333   0.5773503   0.2918678
```

```
$ mpirun -np 4 /home/nfs3/smr3191/q-e/bin/epw.x -npool 4 < epw1.in > epw1.out &
```

The calculation should take about 10 min. In the output, notice the same steps as for exercise1:

- Fourier-transform the electron-phonon matrix elements from a coarse 6x6x6 to a dense 12x12x12 $\mathbf{k}$-point grid and from a coarse 3x3x3 to a dense 12x12x12 $\mathbf{q}$-point grid.
- Write on disk: (1) the `mgb2.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) the `mgb2.freq` file containing the phonon frequencies on the dense $\mathbf{q}$ grid, (3) the `lead.egnv` file containing the eigenvalues within the Fermi window on the dense $\mathbf{k}$ grid, and (4) the `mgb2.ikmap` file containing the index of the k-points on the dense (irreducible) grid within the Fermi window. All these files were produced by setting `ephwrite=.true.`. The files are formatted and required for solving the Migdal-Eliashberg equations. Because the electron-phonon matrix elements do not depend on the temperature at which the Migdal-Eliashberg equations are solved, the files can be reused in subsequent EPW calculations at different temperatures.

- Solve the anisotropic Migdal-Eliashberg equations on the imaginary frequency axis by setting the keywords `eliashberg = .true.`, `liso = .true.`, and `limag = .true.` in the `EPW` input file.

  The anisotropic Migdal-Eliashberg equations take the following form:

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

$$Z_{n\mathbf{k}}(i\omega_j)\Delta_{n\mathbf{k}}(i\omega_j) = \frac{\pi T}{N_\mathrm{F}} \sum_{mj'} \int \frac{d\mathbf{q}}{\Omega_\mathrm{BZ}} \frac{\Delta_{m\mathbf{k}+\mathbf{q}}(i\omega_{j'})}{\sqrt{\omega_{j'}^2 + \Delta_{m\mathbf{k}+\mathbf{q}}^2(i\omega_{j'})}} \times \left[\lambda_{n\mathbf{k},m\mathbf{k}+\mathbf{q}}(\omega_j - \omega_{j'}) - \mu_\mathrm{c}^*\right] \delta(\epsilon_{m\mathbf{k}+\mathbf{q}} - \epsilon_\mathrm{F}) \quad (4)$$

  The anisotropic electron-phonon coupling strength entering in Eqs. (4) is defined as:

$$\lambda_{n\mathbf{k},m\mathbf{k}+\mathbf{q}}(\omega_j) = N_\mathrm{F} \sum_\nu \frac{2\omega_{\mathbf{q}\nu}}{\omega_j^2 + \omega_{\mathbf{q}\nu}^2} |g_{mn\nu}(\mathbf{k},\mathbf{q})|^2 \quad (5)$$

  The semiempirical Coulomb parameter μ_c^* is provided as an input varible `muc` in the EPW calculation.

- 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 (hours to days).

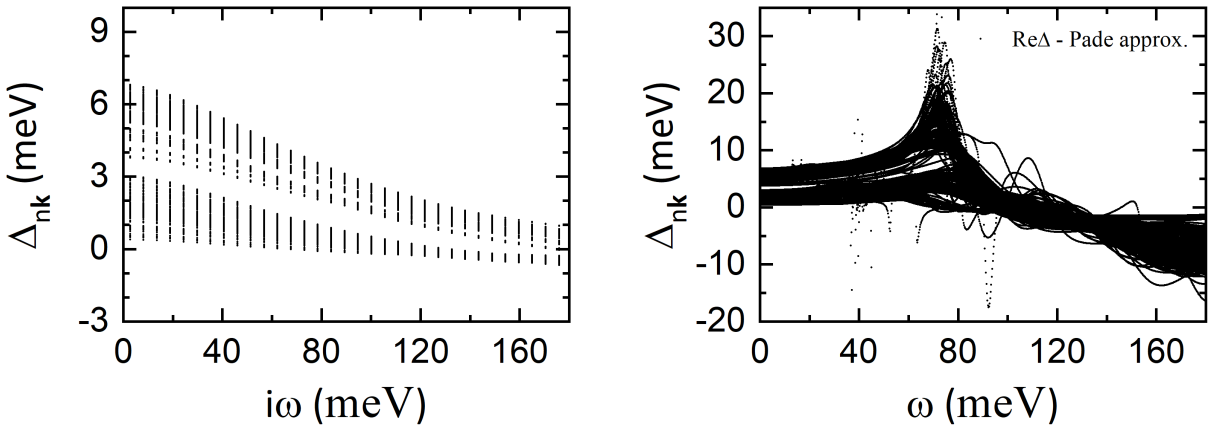
▶ Plot the superconducting gap along the imaginary frequency axis and the real frequency axis.

`mgb2.imag_aniso_XX` files were 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 $Z_{n\mathbf{k}}^N(i\omega_j)$ in the normal state.

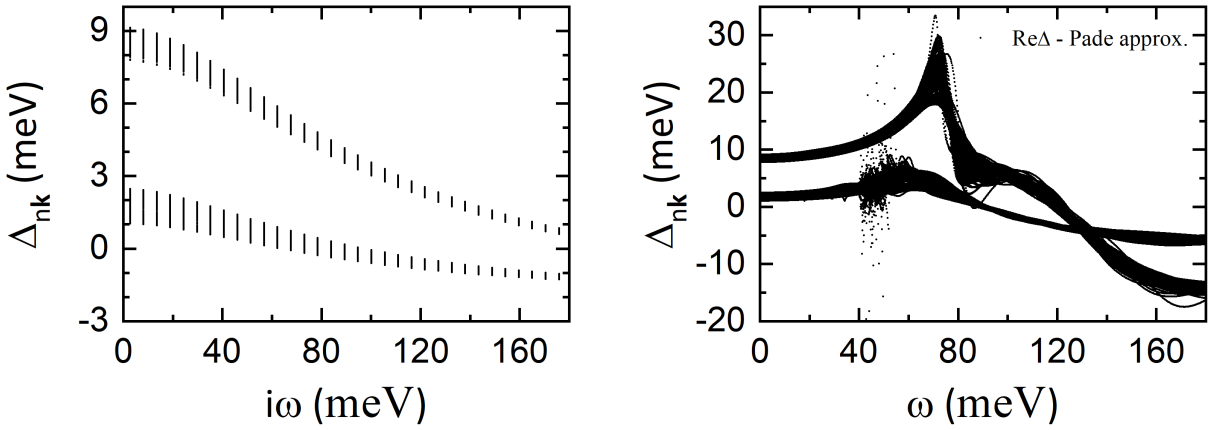
`mgb2.pade_iso_XX` files were generated by setting `lpade = .true.`. Each file contains 6 columns: the energy ω (eV) along the real axis, the Kohn-Sham eigenvalue $\epsilon_{n\mathbf{k}}$ (eV) relative to the Fermi level, the real part of the quasiparticle renormalization $\mathrm{Re}Z_{n\mathbf{k}}(\omega)$, 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 have also been generated by setting `lacon = .true.`. These files will contain similar information as `mgb2.pade_aniso_XX`.

You should get the following graphs at 10 K. The plot on the left is the superconducting gap along the imaginary axis (columns 1:4 from `mgb2.imag_aniso_010.00`). The plot on the right is the superconducting gap along the real axis (columns 1:5 and 1:6 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. However, we can already get relatively decent results. At convergence you should get [4]:



▶ Do a restart calculation (from `mgb2.imag_aniso_010.00`) to compute the superconducting gap function on the imaginary axis at other temperatures.

The input file is as follow (only the difference w.r.t. `epw1.in` are shown):

```
--                                                                    epw2.in
  ep_coupling = .false.
  elph        = .false.

  wannierize  = .false.

  iverbosity  = 1

  ephwrite    = .false.

  imag_read   = .true.

  nstemp   = 5
  tempsmin = 10.00
  tempsmax = 30.00
```

[4] Figure adapted from Margine and Giustino, Phys. Rev. B 87, 024505 (2013).

Notes:

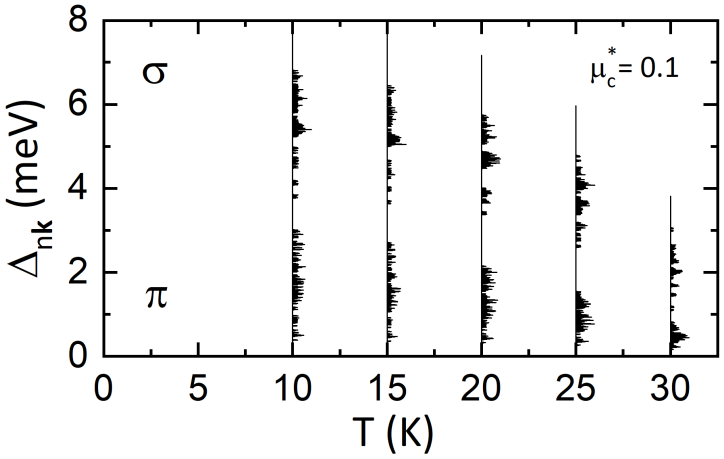
- `imag_read` works if `limag = .true.` and `laniso = .true.`
- `imag_read` allows the code to read from file the superconducting gap and renormalization function on the imaginary axis at specific temperature `XX` from file `mgb2.imag_aniso_XX`. The temperature is specified as `tempsmin = XX` or `temps(1) = XX` 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 `mgb2.imag_aniso_gap_XX_YY.cube`, where `YY` is the band number within the chosen energy window during the EPW calculation.

```
$ mpirun -np 4 /home/nfs3/smr3191/q-e/bin/epw.x -npool 4 < epw2.in > epw2.out &
```

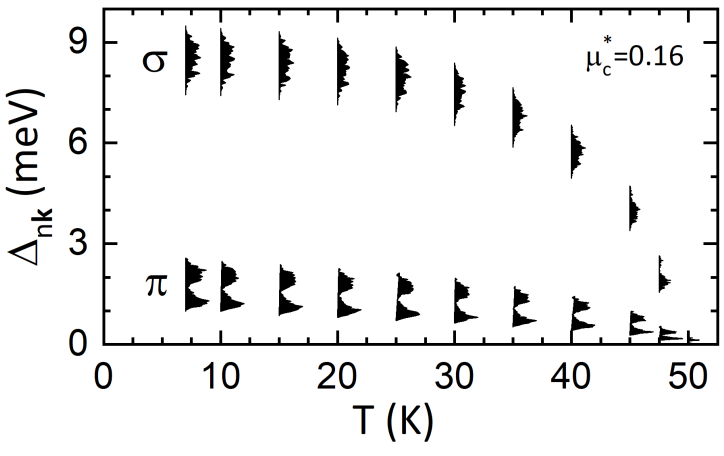
The run should take about 10 min.

▶ 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.



At convergence you should get [5]:



[5] Figure adapted from Margine and Giustino, Phys. Rev. B 87, 024505 (2013).

▶ 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.

▶ Check the effect of the Coulomb pseudopotential μ_c^* on the superconducting gap and the critical temperature by varying the input variable `muc`.