Superconducting properties with EPW

Tutorial Fri.2

Hands-on session

Hands-on based on Quantum Espresso 6.5

Migdal-Eliashberg equations

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

The detail study including the theory related to this tutorial can be found in the Phys. Rev. B **87**, 024505 (2013). In this example we solve the anisotropic Eqs. (21) and (22).

Preliminary calculations with Quantum Espresso

First copy the tutorial input files of the exercise:

```
$ git clone https://github.com/wannier-developers/wannier-tutorials.git
$ cd 2020_03_Oxford/3_epw/tuto_epw_super/
```

▶ Make a self-consistent calculation for MgB_2.

```
 &control                                                                                  scf.in
    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
8 8 8 0 0 0
```

Note: The smearing is quite large in order to get reasonable values in subsequent calculations.

```
$ mpirun -np 2 pw.x < scf.in > scf.out
```

▶ Compute the vibrational properties of MgB_2 on a coarse 2x2x2 $\mathbf{q}$-point grid.

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

```
$ mpirun -np 2 ph.x < ph.in > ph.out &
```

The calculation should take about 5 min on 2 cores. During the run, notice the IBZ q-point grid:

```
Dynamical matrices for ( 2, 2, 2)  uniform grid of q-points
(   4 q-points):
  N         xq(1)         xq(2)         xq(3)
  1   0.000000000   0.000000000   0.000000000
  2   0.000000000   0.000000000  -0.437801761
  3   0.000000000  -0.577350269   0.000000000
  4   0.000000000  -0.577350269  -0.437801761
```

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

```
$ python pp.py
```

▶ Clean up unnecessary files and directories.

```
$ rm -r _ph0/
$ rm -r mgb2.*
```

Solving the anisotropic Migdal-Eliashberg equations with EPW

▶ Re-run the self-consistent calculation for MgB_2. (We have removed the density files because the earlier phonon calculation might have modified them.)

```
$ mpirun -np 2 pw.x < scf.in > scf.out
```

▶ Do a non self-consistent calculation on a homogeneous 6x6x6 **uniform and Γ-centered grid between [0,1[in crystal coordinates**. You can use again the script `kmesh.pl` to generate the homogeneous k grid as; `wannier90-3.0.0/utility/kmesh.pl 6 6 6`

```
&control                                                              nscf.in
   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 2 pw.x -npool 2 < 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.          ! run e-ph coupling calculation
  elph        = .true.          ! calculate el-ph coefficients
  kmaps       = .false.         ! generate the map k+q --> k for folding the rotation matrix U(k+q)
                                ! on the coarse grid
  epbwrite    = .true.          ! write el-ph matrices in the coarse Bloch representation
  epbread     = .false.         ! read from the 'prefix.epb' pool dependent files
  epwwrite    = .true.          ! write el-ph matrices in the Wann representation
  epwread     = .false.         ! read el-ph matrices from the 'prefix.epmatwp1' file

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

  nbndsub     =  5              ! number of wannier functions to utilize
  nbndskip    =  0              ! number of bands skipped in the wannierization lying below
                                ! the disentanglement
  wannierize  = .true.          ! calculate the Wannier functions using W90 library
  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]
  eptemp      = 300             ! smearing for the Fermi occupation in [K]
  degaussw    = 0.05            ! smearing in the energy-conserving delta functions in [eV]

  ephwrite    = .true.          ! write .ephmatXX, .egnv, .freq, and .ikmap files

  eliashberg  = .true.          ! solve the ME eqs.

  laniso = .true.               ! solve the anisotropic ME eqs.
  limag = .true.                ! solve the imag-axis ME eqs.
  lpade = .true.                ! solve ME eqs. on the real axis using Pade approximants
  nsiter   = 500                ! number of self-consistent iterations when solving the ME eqs.

  conv_thr_iaxis = 1.0d-3       ! convergence threshold for solving ME eqs. on imag-axis

  wscut = 0.5                   ! upper limit over Matsubara frequency summation in ME eqs
                                !  on imag-axis in [eV]
  nstemp   = 1                  ! number of temperature points at which the ME eqs. are solved
  tempsmin = 10.00              ! step between points is (tempsmax - tempsmin ) / (nstemp-1)
  tempsmax = 15.00

  muc     = 0.05                ! effective Coulomb potential used in the ME eqs.

  dvscf_dir   = './save'

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

  nq1         = 2               ! dimensions of the coarse phonon grid
  nq2         = 2
  nq3         = 2

  mp_mesh_k = .true.            ! use irreducible electronic fine mesh
  nkf1 = 24                   ! dimensions of the fine electron grid
  nkf2 = 24
  nkf3 = 24

  nqf1 = 12                   ! dimensions of the fine phonon grid
  nqf2 = 12
  nqf3 = 12
 /
  4 cartesian                 ! list of irreducible q-points on coarse grid (copy from ph.out or .dyn0)
  0.000000000   0.000000000   0.000000000
  0.000000000   0.000000000  -0.437801761
  0.000000000  -0.577350269   0.000000000
  0.000000000  -0.577350269  -0.437801761
```

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.

Note The **k** and **q** coarse grids need to be commensurate.

```
$ mpirun -np 2 epw.x -npool 2 < epw1.in > epw1.out &
```

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

The calculation should take about 12 min on 2 cores. With the above input, we are instructing EPW to:

- Fourier-transform the electron-phonon matrix elements from a coarse 6x6x6 to a dense 24x24x24 $\mathbf{k}$-point grid and from a coarse 2x2x2 to a dense 12x12x12 $\mathbf{q}$-point grid.

```
Using uniform q-mesh:    12   12   12
Size of q point mesh for interpolation:       1728
Using uniform MP k-mesh:    24   24   24
Size of k point mesh for interpolation:       1586
Max number of k points per pool:              794
```

- 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 `mgb2.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.

```
Finish writing .ikmap file

Finish mapping k+sign*q onto the fine irreducibe k-mesh

Nr irreducible k-points within the Fermi shell =       118 out of       793
Progression iq (fine) =         100/      1728
Progression iq (fine) =         200/      1728
...
...
Progression iq (fine) =        1700/      1728
```

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

  The anisotropic Migdal-Eliashberg equations take the following form:

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

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

The anisotropic electron-phonon coupling strength entering in Eqs. (1) is defined as:

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

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 =    0.4766640

Estimated Allen-Dynes Tc =    14.714941 K for muc =    0.05000

Estimated BCS superconducting gap =     0.002232 eV

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: ~=    0.1094 Gb
iter       ethr          znormi  [eV]     deltai  [eV]
1   3.189420E+00   1.458415E+00   2.368222E-03
2   7.367752E-02   1.458167E+00   2.441382E-03
...
15   3.205864E-04   1.457147E+00   2.726953E-03
Convergence was reached in nsiter =     15
```

- 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).

```
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   1.429292E+00   2.557589E-03

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

▶ 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 $Z^N_{n\mathbf{k}}(i\omega_j)$ 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.

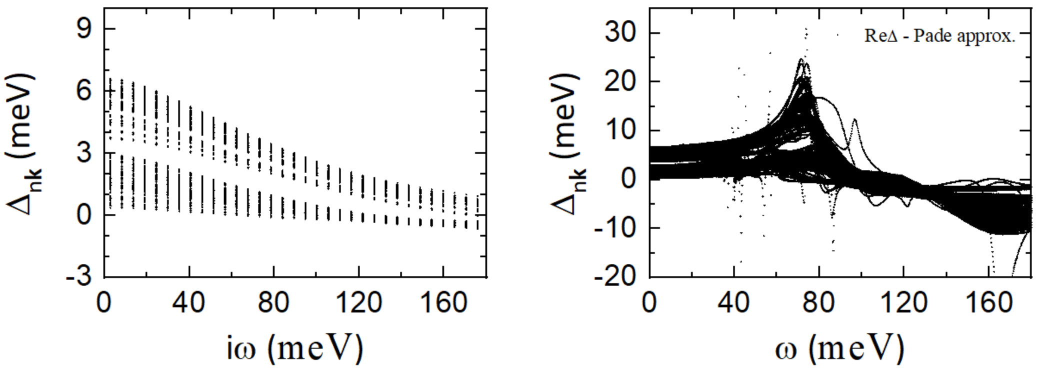

Fig. 1 The plot on the left is the superconducting gap along the imaginary axis (columns 1:4 from `mgb2.imag_aniso_010.00`). The plot on the right is the superconducting gap along the real axis (columns 1:5 and 1:6 from `mgb2.pade_aniso_010.00` - this file is about 70MB).

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:

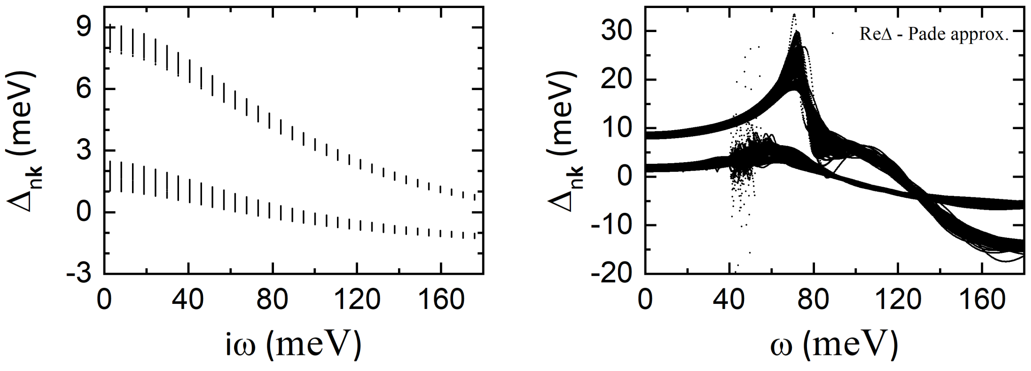

Fig. 2 Calculated energy-dependent superconducting gap of MgB_2 at T=10 K. The gap is obtained by solving the fully anisotropic Migdal-Eliashberg equations with μ^*= 0.16. (left) Superconducting gap along the imaginary energy axis (black dots). (right) Superconducting gap along the real energy axis, obtained from the approximate analytic continuation using Padé functions (black dots). Figure adapted from Phys. Rev. B **87**, 024505 (2013).

▶ 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.       ! read from file the superdconducting gap

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

Notes:

- `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.
- `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 `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 2 epw.x -npool 2 < epw2.in > epw2.out
```

The run should take about 3 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.

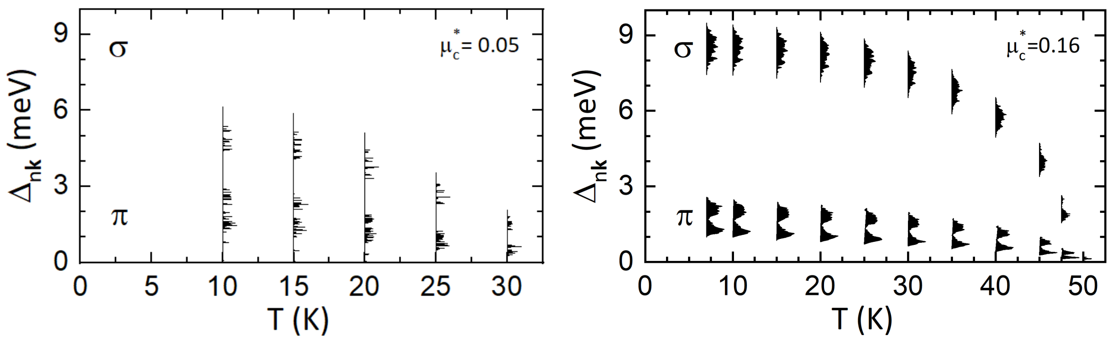

Fig. 3 Calculated anisotropic superconducting gaps of MgB_2 on the Fermi surface as a function of temperature. At convergence you should get the right-side figure adapted from Phys. Rev. B **87**, 024505 (2013).

▶ 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`.

Solving the isotropic Migdal-Eliashberg equations with EPW

▶ 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 = 'mgb2.a2f_iso'`.

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.

The input file is as follow (only the differences w.r.t. `epw2.in` are shown):

```
--                                                                        epw3.in
fila2f    = 'mgb2.a2f_iso'

liso      = .true.          ! solve the isotropic ME eqs.
limag     = .true.

nstemp    = 17
tempsmin  = 1.00
tempsmax  = 17.00
```

```
$ mpirun -np 1 epw.x -npool 1 < epw3.in > epw3.out
```

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

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

To obtain the `mgb2.imag_iso_gap0` file use the `script_gap0_imag` shell script:

```
$  ./script_gap0_imag
```

You should get the following graph:

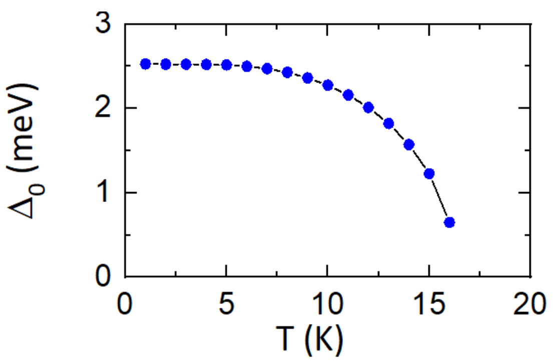

Fig. 3 Calculated isotropic superconducting gap of MgB_2 at the Fermi level as a function of temperature.