Advanced QE school: Hubbard and Koopmans functionals from linear response

Carrier mobility and superconductivity with EPW

Hands-on Session (Tue.2)

Hands-on based on `Quantum ESPRESSO` (v7.2) and `EPW` v5.7
Tutorial based on the 2023 Virtual School on Many-Body Calculations using EPW and BerkeleyGW

Introduction

In this tutorial, we will show in **Exercise 1** how to compute the intrinsic electron and hole low-field drift and Hall mobility of the polar cubic semiconductor BN using the linearised iterative Boltzmann transport equation (IBTE) and the self-energy relaxation time approximation (SERTA), with or without external magnetic field. In **Exercise 2** we will compute the superconducting properties of MgB_2 by solving the anisotropic Migdal-Eliashberg equations.
For the description for all input flags please follow the link:

`https://docs.epw-code.org/doc/Inputs.html`

Exercise 1

1.1 Theory

In this example we are going to calculate the drift and Hall hole carrier mobility of c-BN. The drift mobility is obtained with:

$$\mu^{\mathrm{d}}_{\alpha\beta} = \frac{-1}{V^{\mathrm{uc}} n^{\mathrm{c}}} \sum_n \int \frac{\mathrm{d}^3 k}{\Omega^{\mathrm{BZ}}} \, v_{n\mathbf{k}\alpha} \partial_{E_\beta} f_{n\mathbf{k}} \tag{1}$$

where the out of equilibrium occupations are obtained by solving the BTE:

$$\partial_{E_\beta} f_{n\mathbf{k}} = e v_{n\mathbf{k}\beta} \frac{\partial f^0_{n\mathbf{k}}}{\partial \varepsilon_{n\mathbf{k}}} \tau_{n\mathbf{k}} + \frac{2\pi \tau_{n\mathbf{k}}}{\hbar} \sum_{m\nu} \int \frac{\mathrm{d}^3 q}{\Omega_{\mathrm{BZ}}} |g_{mn\nu}(\mathbf{k},\mathbf{q})|^2$$
$$\times \Big[(n_{\mathbf{q}\nu} + 1 - f^0_{n\mathbf{k}}) \delta(\varepsilon_{n\mathbf{k}} - \varepsilon_{m\mathbf{k}+\mathbf{q}} + \hbar\omega_{\mathbf{q}\nu}) + (n_{\mathbf{q}\nu} + f^0_{n\mathbf{k}}) \delta(\varepsilon_{n\mathbf{k}} - \varepsilon_{m\mathbf{k}+\mathbf{q}} - \hbar\omega_{\mathbf{q}\nu}) \Big] \partial_{E_\beta} f_{m\mathbf{k}+\mathbf{q}}. \tag{2}$$

The scattering rate in Eq. (2) is defined as:

$$\tau^{-1}_{n\mathbf{k}} \equiv \frac{2\pi}{\hbar} \sum_{m\nu} \int \frac{d^3 q}{\Omega_{\mathrm{BZ}}} |g_{mn\nu}(\mathbf{k},\mathbf{q})|^2 \big[(n_{\mathbf{q}\nu} + 1 - f^0_{m\mathbf{k}+\mathbf{q}})$$
$$\times \delta(\varepsilon_{n\mathbf{k}} - \varepsilon_{m\mathbf{k}+\mathbf{q}} - \hbar\omega_{\mathbf{q}\nu}) + (n_{\mathbf{q}\nu} + f^0_{m\mathbf{k}+\mathbf{q}}) \delta(\varepsilon_{n\mathbf{k}} - \varepsilon_{m\mathbf{k}+\mathbf{q}} + \hbar\omega_{\mathbf{q}\nu}) \big]. \tag{3}$$

A common approximation to Eq. (2) is called the self-energy relaxation time approximation (SERTA) and consists in neglecting the second term in the right-hand of the equation which gives:

$$\mu^{\mathrm{SERTA}}_{\alpha\beta} = \frac{-e}{V^{\mathrm{uc}} n^{\mathrm{c}}} \sum_n \int \frac{\mathrm{d}^3 k}{\Omega^{\mathrm{BZ}}} \frac{\partial f^0_{n\mathbf{k}}}{\partial \varepsilon_{n\mathbf{k}}} v_{n\mathbf{k}\alpha} v_{n\mathbf{k}\beta} \tau_{n\mathbf{k}}. \tag{4}$$

The the low-field phonon-limited carrier mobility in the presence of a small finite magnetic field **B** is given by:

$$\mu_{\alpha\beta}(B_\gamma) = \frac{-1}{V^{\mathrm{uc}} n^{\mathrm{c}}} \sum_n \int \frac{\mathrm{d}^3 k}{\Omega^{\mathrm{BZ}}} v_{n\mathbf{k}\alpha} [\partial_{E_\beta} f_{n\mathbf{k}}(B_\gamma) - \partial_{E_\beta} f_{n\mathbf{k}}], \tag{5}$$

again solving the BTE with finite (small) magnetic field:

$$\left[1 - \frac{e}{\hbar}\tau_{n\mathbf{k}}(\mathbf{v}_{n\mathbf{k}} \times \mathbf{B}) \cdot \nabla_{\mathbf{k}}\right] \partial_{E_\beta} f_{n\mathbf{k}}(B_\gamma) = e v_{n\mathbf{k}\beta} \frac{\partial f^0_{n\mathbf{k}}}{\partial \varepsilon_{n\mathbf{k}}} \tau_{n\mathbf{k}} + \frac{2\pi\tau_{n\mathbf{k}}}{\hbar} \sum_{m\nu} \int \frac{\mathrm{d}^3 q}{\Omega^{\mathrm{BZ}}} |g_{mn\nu}(\mathbf{k},\mathbf{q})|^2$$
$$\times \left[(n_{\mathbf{q}\nu} + 1 - f^0_{n\mathbf{k}}) \delta(\varepsilon_{n\mathbf{k}} - \varepsilon_{m\mathbf{k}+\mathbf{q}} + \hbar\omega_{\mathbf{q}\nu}) + (n_{\mathbf{q}\nu} + f^0_{n\mathbf{k}}) \delta(\varepsilon_{n\mathbf{k}} - \varepsilon_{m\mathbf{k}+\mathbf{q}} - \hbar\omega_{\mathbf{q}\nu})\right] \partial_{E_\beta} f_{m\mathbf{k}+\mathbf{q}}(B_\gamma). \tag{6}$$

The Hall factor and Hall mobility are then obtained as:

$$r_{\alpha\beta}(\hat{\mathbf{B}}) \equiv \lim_{\mathbf{B}\to 0} \sum_{\delta\epsilon} \frac{[\mu^{\mathrm{d}}_{\alpha\delta}]^{-1} \mu_{\delta\epsilon}(\mathbf{B}) [\mu^{\mathrm{d}}_{\epsilon\beta}]^{-1}}{|\mathbf{B}|} \tag{7}$$

$$\mu^{\mathrm{Hall}}_{\alpha\beta}(\hat{\mathbf{B}}) = \sum_\gamma \mu^{\mathrm{d}}_{\alpha\gamma} r_{\gamma\beta}(\hat{\mathbf{B}}), \tag{8}$$

where $\hat{\mathbf{B}}$ is the direction of the magnetic field. More information can be found in the review Rep. Prog. Phys. **83**, 036501 (2020).

Ionized impurity scattering is an important mobility limiting mechanism alongside thermal lattice vibrations in doped materials or materials with native ionized point defects. For 3D bulk crystals containing randomly distributed ionized impurities, we can exploit the Kohn-Luttinger ensemble average to obtain the partial transition rates of carriers by the ionized impurities.

$$\tau^{-1,\mathrm{ii}}_{n\mathbf{k}\to m,\mathbf{k}+\mathbf{q}} = n_{\mathrm{ii}} \frac{2\pi}{\hbar} \left[\frac{e^2}{4\pi\varepsilon_0} \frac{4\pi Z}{\Omega_{\mathrm{u.c}}}\right]^2 \sum_{\mathbf{G}\neq -\mathbf{q}} \frac{|\langle u_{m\mathbf{k}+\mathbf{q}} | e^{i\mathbf{G}\cdot\mathbf{r}} | u_{n\mathbf{k}} \rangle_{\mathrm{u.c}}|^2}{|(\mathbf{q}+\mathbf{G}) \cdot \varepsilon^0 \cdot (\mathbf{q}+\mathbf{G})|^2} \delta(\epsilon_{n\mathbf{k}} - \epsilon_{m\mathbf{k}+\mathbf{q}}), \tag{9}$$

Here, n_{ii} is the concentration of ionized impurities in the system, Z is the charge of the ionized impurities, and ε^0 is the low-frequency dielectric tensor which includes contributions from both electronic and ionic polarizability. More information can be found in the article Phys. Rev. B **107**, 125207 (2023).

1.2 Preliminary calculations with Quantum Espresso

First download the exercise files:

```
$ wget https://github.com/materialscloud-org/QuantumESPRESSO-school-2023/tree/main/Day2/Day2.Ponce.tar
$ tar -xvf Day2.Ponce.tar
$ cd exercise1/
```

▶ Make a self-consistent calculation for c-BN.

```
--                                                                    scf.in
 &control
    calculation     = 'scf'
    prefix          = 'bn'
    restart_mode    = 'from_scratch'
    pseudo_dir      = './'
    outdir          = './'
 /
```

```
 &system
    ibrav           = 2
    celldm(1)       = 6.833
    nat             = 2
    ntyp            = 2
    ecutwfc         = 40
 /
 &electrons
    diagonalization = 'david'
    mixing_beta     = 0.7
    conv_thr        = 1.0d-13
 /
ATOMIC_SPECIES
  B  10.811   B-PBE.upf
  N  14.0067  N-PBE.upf
ATOMIC_POSITIONS {crystal}
 B    0.00  0.00   0.00
 N   -0.25  0.75  -0.25
K_POINTS automatic
8 8 8 0 0 0
```

Note: In practice the **k**-point grid needs to be fairly large in order to get converged dielectric function and Born effective charges during the following phonon calculation.

```
$ mpirun -np 2 pw.x -in scf.in | tee scf.out
```

▶ Compute the vibrational properties of c-BN on a coarse 4x4x4 $\mathbf{q}$-point grid.

```
--                                                                                ph.in
&inputph
  tr2_ph=1.0d-17,
  prefix='bn',
  amass(1)=10.811,
  amass(2)=14.0067,
  outdir='./',
  fildyn='bn.dyn.xml',
  fildvscf='dvscf'
  ldisp=.true.,
  epsil=.true.,
  nq1 = 4,
  nq2 = 4,
  nq3 = 4
 /
```

Note: We have the input variable `epsil=.true.` which computes the macroscopic dielectric constant in non-metallic systems. If you add `.xml` after the name of the dynamical matrix file, it will produce the data in XML format (preferred).
Note 2: The input variable responsible to produce the electron-phonon matrix element is `fildvscf`. Always make sure that this variable is present.
Note 3: Notice the very tight `tr2_ph` threshold parameter on the self-consistent first-order perturbed wavefunction. This is crucial to obtain good vibrational properties.

```
$ mpirun -np 2 ph.x -in ph.in | tee ph.out
```

The calculation should take about 5 min on 4 cores. During the run, notice the IBZ q-point grid:

```
     Dynamical matrices for ( 4, 4, 4)  uniform grid of q-points
     (   8 q-points):
       N         xq(1)         xq(2)         xq(3)
       1   0.000000000   0.000000000   0.000000000
       2  -0.250000000   0.250000000  -0.250000000
       3   0.500000000  -0.500000000   0.500000000
```

```
4   0.000000000   0.500000000   0.000000000
5   0.750000000  -0.250000000   0.750000000
6   0.500000000   0.000000000   0.500000000
7   0.000000000  -1.000000000   0.000000000
8  -0.500000000  -1.000000000   0.000000000
```

as well as the dielectric function and Born effective charge tensor:

```
     Dielectric constant in cartesian axis

     (       4.597197252      -0.000000000       0.000000000 )
     (      -0.000000000       4.597197252       0.000000000 )
     (      -0.000000000       0.000000000       4.597197252 )

     Effective charges (d Force / dE) in cartesian axis with asr applied:
      atom      1   B  Mean Z*:        1.89277
E*x (        1.89277       -0.00000       -0.00000 )
E*y (       -0.00000        1.89277       -0.00000 )
E*z (        0.00000       -0.00000        1.89277 )
      atom      2   N  Mean Z*:       -1.89277
E*x (       -1.89277        0.00000        0.00000 )
E*y (        0.00000       -1.89277        0.00000 )
E*z (       -0.00000        0.00000       -1.89277 )
```

The experimental dielectric constant in c-BN is about 4.46. More accurate values can be obtained with larger $\mathbf{k}$-point grids. c-BN is a polar material and has a Born effective charge of 1.89 which is very close to theoretical value of 1.91.

Finally, we need to post-process some of the data to make it ready for EPW. To do so, we can use a python script (usually provided in `QE/EPW/bin/pp.py` but copied here for convenience).

▶ Run the python post-processing to create the `save` folder

```
$ python3 pp.py
```

The script will ask you to enter the prefix used for the calculation. In this case enter "bn". The script will create a new folder called "save" that contains the dvscf potential files, pattern files, and dynamical matrices on the IBZ.

▶ Compute the low-frequency dielectric constant including ionic polarization for c-BN.
We need one final ingredient before moving on to running EPW if we intend to examine ionized impurity scattering, and that is to determine the zone-center dielectric polarizability of c-BN including ionic polarization. We have the electronic contribution from `ph.x`, but we will use `dynmat.x` to determine ε^0. Below is input for `dynmat.x`, `dynmat.in`.

```
--                                                                    dynmat.in
&input
  fildyn='bn.dyn1.xml',
  amass(1)=10.811,
  amass(2)=14.0067,
  asr='simple',
  lperm=.true. ! calculate ionic polarizability
  filout='bn.dynmat.out',
  fileig='bn.eig'
 /
```

To do this, run:

```
$ dynmat.x -in dynmat.in | tee dynmat.out
```

We can find the following lines in the `dynmat.out` file:

```
 Electronic dielectric permittivity tensor (F/m units)
         4.597197     0.000000     0.000000
        -0.000000     4.597197    -0.000000
         0.000000     0.000000     4.597197

 ... with zone-center polar mode contributions
         6.629671     0.000000    -0.000000
         0.000000     6.629671    -0.000000
        -0.000000    -0.000000     6.629671
```

We will need the value of ≈ 6.63 for ϵ_0 to include ionized impurity scattering later on.

1.3 Interpolation of the electron-phonon matrix element in real-space with EPW

▶ Do a non self-consistent calculation on a 4x4x4 uniform and Γ-centered $\mathbf{k}$-point grid with crystal coordinates in the interval [0,1[
Such a grid can be for example generated with the `wannier90` utility with `kmesh.pl 4 4 4`.
The `nscf.in` file is as follow:

```
--                                                                                                  nscf.in
 &control
    calculation     = 'nscf'
    prefix          = 'bn'
    restart_mode    = 'from_scratch'
    pseudo_dir      = './'
    outdir          = './'
 /
 &system
    ibrav           = 2
    celldm(1)       = 6.833
    nat             = 2
    ntyp            = 2
    ecutwfc         = 40
    nbnd            = 20
 /
 &electrons
    diagonalization = 'david'
    mixing_beta     = 0.7
    conv_thr        = 1.0d-13
 /
ATOMIC_SPECIES
  B  10.811   B-PBE.upf
  N  14.0067  N-PBE.upf
ATOMIC_POSITIONS {crystal}
 B    0.00  0.00  0.00
 N   -0.25  0.75 -0.25
K_POINTS crystal
64
  0.00000000  0.00000000  0.00000000  1.562500e-02
  0.00000000  0.00000000  0.25000000  1.562500e-02
  0.00000000  0.00000000  0.50000000  1.562500e-02
...
```

```
$ mpirun -np 2 pw.x -in nscf.in | tee nscf.out
```

The reason for the non-self consistent calculation is that EPW needs the wavefunctions on the full BZ on a grid between 0 and 1.

Note: Since we are also interested in electron mobility, we will need the conduction bands. Notice that we added the input `nbnd = 20` in `nscf.in`

▶ Perform an `EPW` calculation to Fourier-transform the electron-phonon matrix element from a coarse 4x4x4 $\mathbf{k}$ and $\mathbf{q}$-point grids to real space and then interpolate the electronic band structure and phononic dispersion along the $L-\Gamma-X-K-\Gamma$ high symmetry line by reading the file `LGXKG.txt`.

```
--                                                                                epw1.in
&inputepw
  prefix      = 'bn'
  outdir      = './'

  elph        = .true.
  epbwrite    = .true.
  epbread     = .false.
  epwwrite    = .true.
  epwread     = .false.
  etf_mem     =  1
  lpolar      = .true.   ! polar material
  vme         = 'dipole'


  nbndsub     =  3
  bands_skipped = 'exclude_bands = 1, 5-20'

  wannierize   = .true.
  num_iter     = 50000
  iprint       = 2
  dis_win_max  =  12.0
  dis_win_min  = -1.0

  proj(1)     = 'N:p'

  wdata(1) = 'bands_plot = .true.'
  wdata(2) = 'begin kpoint_path'
  wdata(3) = ' L  0.500  0.500  0.500   G  0.000  0.000  0.000 '
  wdata(4) = ' G  0.000  0.000  0.000   X  0.500  0.000  0.500 '
  wdata(5) = ' X  0.500  0.000  0.500   K  0.375  0.375  0.750 '
  wdata(6) = ' K  0.375  0.375  0.750   G  0.000  0.000  0.000 '
  wdata(7) = 'end kpoint_path'
  wdata(8) = 'bands_plot_format = gnuplot'
  wdata(9) = 'guiding_centres = .true.'
  wdata(10) = 'dis_num_iter      = 5000'
  wdata(11) = 'num_print_cycles  = 10'
  wdata(12) = 'dis_mix_ratio     = 1.0'
  wdata(13) = 'conv_tol = 1E-12'
  wdata(14) = 'conv_window = 4'
  wdata(15) = 'use_ws_distance = T'

  fsthick     = 100
  degaussw    = 0.001

  dvscf_dir   = './save'

  band_plot   = .true.

  filkf       = './LGXKG.txt'
  filqf       = './LGXKG.txt'

   nk1        = 4
   nk2        = 4
   nk3        = 4
   nq1        = 4
   nq2        = 4
   nq3        = 4
```

```
/
```

```
$ mpirun -np 2 epw.x -npool 2 -input epw1.in | tee epw1.out
```

Note: The number of pool `-npool` has to be the same as the total number of core `-np` since **k**-point parallelization is (almost) the only parallelization level allowed. **G**-vector parallelization will be introduced in EPW v6.0.

The calculation should take less than 2 min. Note that the code should have detected the presence of the `quadrupole.fmt` file and correclty read the quadrupole tensor. Look in the output for the line `Quadrupole tensor is correctly read:`. In this hands-on we will not cover how to obtain the quadrupole tensor and they are simply given here. There are two ways to obtain them:

- Using perturbation theory. This is implemented in a recent version of the Abinit software.
- Fitting the perturbed density or the electron-phonon matrix elements in the long wavelength limit obtained by direct DFPT calculations.

More information can be found in Phys. Rev. Research **3**, 043022 (2021)

At the end of the calculation, because of the keyword `band_plot = .true.`, the code should produce the `band.eig` and `phband.freq` files that contain the electronic band structure and phononic dispersion along a path given in the `filkf` and `filqf` files.
If you want to have files in an easy gnuplot format, you can use the `plotband.x` tool by doing

```
$ plotband.x
```

and follow the instructions. You should check that both plots look reasonable.

▶ Do a restart calculation (restarting from the `bn.epmatwp1` file) and compute the hole mobility of c-BN.

```
$ mpirun -np 2 epw.x -npool 2 -input epw2.in | tee epw2.out
```

The input file is as follow:

```
--                                                                                epw2.in
  &inputepw
  prefix      = 'bn'
  outdir      = './'

  elph        = .true.
  epwwrite    = .false.
  epwread     = .true.
  etf_mem     =  3      ! generate k-points within fsthick
  lpolar      = .true.
  vme         = 'dipole'
  mp_mesh_k   = .true.

  nbndsub     =  3
  bands_skipped = 'exclude_bands = 1, 5-20'

  scattering  = .true.
  scattering_serta = .true.
  int_mob     = .false.
  carrier     = .true.
  ncarrier    =  -1E13
  iterative_bte = .true.
  epmatkqread = .false.
  mob_maxiter = 300
  broyden_beta= 1.0
```

```
 bfieldx     = 0.0d0
 bfieldy     = 0.0d0
 bfieldz     = 1.0d-10  ! Apply a magnetic field along Cart. z

 nstemp      = 1
 temps       = 300

 restart      = .true.
 restart_step = 1000

 wannierize  = .false.
 num_iter    = 50000
 iprint      = 2
 dis_win_max = 12.0
 dis_win_min = -1.0

 proj(1)     = 'N:p'

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

 fsthick     = 0.4 ! 0.3 eV
 degaussw    = 0.0

 efermi_read  = .true
 fermi_energy = 11.246840

 dvscf_dir   = './save'

 nkf1        = 30
 nkf2        = 30
 nkf3        = 30
 nqf1        = 30
 nqf2        = 30
 nqf3        = 30

  nk1         = 4
  nk2         = 4
  nk3         = 4
  nq1         = 4
  nq2         = 4
  nq3         = 4
/
```

Notes:

- The value of `fermi_energy` was obtained from the output of the previous calculation `epw1.in`
- `epwread` allows for the restart from the `bn.epmatwp1` file
- `int_mob` allows to perform both electron and hole calculations at the same time but is not recommanded.
- `carrier` and `ncarrier` define the carrier concentration. If `carrier = .true.` then the intrinsic mobility with `ncarrier` concentration (in cm^{-3}) is computed. If `ncarrier` is positive it will compute the electron mobility and if it is negative it will compute the hole mobility. The resulting mobility should be independent of the choice of carrier concentration in reasonable ranges 10^{10}- 10^{16} cm^{-3}.
- `iterative_bte` asks for the iterative solution of the BTE in addition to SERTA.
- `nstemp` and `temps` define the lattice temperature at which the mobility is evaluated.
- `restart` and `restart_step` will create restart point every (in this case) 1000 $\mathbf{q}$-points. You can try breaking the run after a restart point and restart to test this feature.
- `bfieldz` adds a (small) finite magnetic field along the Cartesian z direction (in unit of Tesla). This will automatically trigger the calculation of the Hall factor.
- `mob_maxiter` is the maximum number of iterations for the BTE solution.
- `degaussw = 0.0` means that adaptive smearing is used. Positive values give Gaussian smearing.

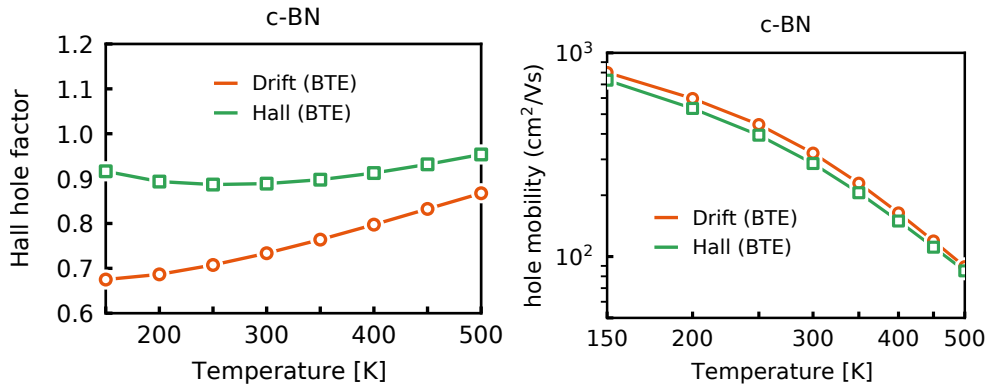
The run should take about 4 min. The fine $\mathbf{k}$ and $\mathbf{q}$ point grids need to be much denser for real calculations. However, we can already get relatively decent results.

▶ Re-run the code with multiple temperatures (using `nstemp = 4` and `temps = 100, 200, 400, 500`). You should remove the `restart.fmt` file before doing so.

Try filling the table below for the hole mobility:

T (K)	hole ε_F (eV)	drift SERTA μ (cm^2/Vs)	drift BTE μ (cm^2/Vs)	Hall BTE μ (cm^2/Vs)
100				
200				
300				
400				
500				

At convergence you should get [1]:



where the room temperature values with SOC should be around 319 cm^2/Vs for the drift BTE and 281 cm^2/Vs for the Hall mobility with a Hall factor of 0.88.

▶ Try to increase the fine grids and add a few more temperatures and see if you can get a result closer to convergence.
▶ Try adding SOC
▶ Try removing or renaming the file `quadrupole.fmt` to do the interpolation with dipole only and see the impact on the results.

1.4 Compute the spectral decomposition

▶ Do a restart calculation (restarting from the `bn.epmatwp1` file) and compute the hole spectral decomposition of c-BN.
▶ You should remove the `restart.fmt` file.

```
$ mpirun -np 2 epw.x -npool 2 -input epw3.in | tee epw3.out
```

The input file is as follow (we show only the difference wrt `epw2.in`):

```
--                                                                    epw3.in
  &inputepw

  iverbosity  = 3
  mob_maxfreq = 160
```

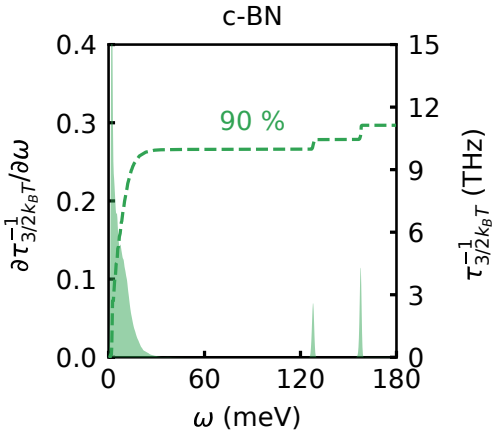
[1] The figure is from Phys. Rev. Research **3**, 043022 (2021)

```
  mob_nfreq   = 640 ! To have 0.25 meV intervals

  nkf1        = 60
  nkf2        = 60
  nkf3        = 60
  nqf1        = 60
  nqf2        = 60
  nqf3        = 60
 /
```

At the end of the calculation, the code should have produced a file named `inv_tau_freq.fmt`. You should open the file and look for the maximum number of **k**-points within the `fsthick`, here you should have 272. You should also look for the number of bands, here you should find 3.
▶ Edit the `gaussian-h.py` python script to correspond to the calculation you have been doing. Then run it.

```
$ python3 gaussian-h.py
```

The script should produce a file named `inv_tau_freq.fmt-gaussian1.0` which you can plot with your favorite software. Note that the results are not converged but it should be clear that acoustic scattering is dominating in c-BN.
At convergence you should get [2]:



1.5 Include carrier-ionized impurity scattering

▶ You will now run a calculations that includes the influence of ionized impurity scattering on the hole mobility of c-BN.
▶ You should once again remove the restart.fmt file:

```
$ rm restart.fmt
```

Then, run EPW:

```
$ mpirun -np 2 epw.x -npool 2 -input epw4.in | tee epw4.out
```

The input file for `epw4.in` is as follow (we show only the difference wrt `epw2.in`):

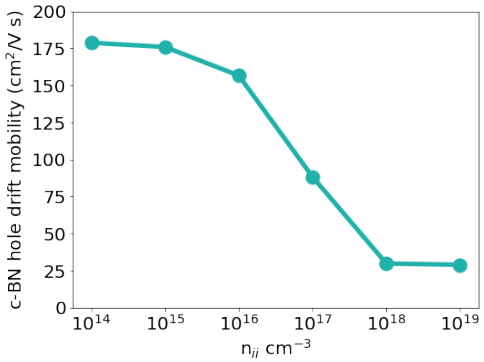
[2] The figure is from Phys. Rev. Research **3**, 043022 (2021)

```
--                                                                                    epw4.in
  &inputepw

  ncarrier = -1E17 ! absolute value MUST match ii_n, negative sign for holes.

  ii_g = .true.
  ii_charge = 1.0d0
  ii_n = 1.0d17  ! must match ncarrier, keep positive for both holes and electrons
  ii_scattering = .true.
  ii_only = .false.
  ii_eps0 = 6.62967d0
 /
```

Notes:

- `ii_g` = .true. tells EPW to calculate ionized impurity matrix elements
- `ii_charge` is the charge of the ionized impurities in units of the electron charge.
- `ii_n` is the density of ionized impurities in units of cm^{-3}. For bulk, typical doping values range from 10^{15} to 10^{18} cm^{-3}, higher would result in degenerate doping.
- `ii_scattering` = .true. tells EPW to compute the ionized impurity scattering rates.
- `ii_only`: if `ii_only` were set to `.true.`, this would tell EPW to only include ionized impurity scattering and omit phonon scattering. Since we want phonon and ionized impurity scattering, we set `ii_only = .false.`.
- `ii_eps0` is the low-frequency dielectric constant, determined by `dynmat.x`.

After running `epw.x` with input `epw4.in`, we should find a hole mobility at 300 K of 88.25 cm^2/V s, reduced from a phonon-only limited mobility of 178.87 cm^2/V s previously calculated. The presence of ionized impurities strongly influences the drift mobilities. As an exercise, you can modify `ii_n` (and `ncarrier` accordingly), between 1.0d15 and 1.0d19 to see how the hole mobility at 300 K changes as the concentration of ionized impurities increases. Here is the plot of the expected results for this example.



NOTE: Ionized impurity scattering and Hall mobilities have not been tested together. The use of the ionized impurity functionality in EPW should be currently limited to drift mobilities only.

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 $\mathbf{q}$-point grid using the following jobscript (`job.ph`) and input files (`scf.in` and `ph.in`) for MgB_2:

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

```
$ cd phonon
$ mpirun -np 2 pw.x -in scf.in | tee scf.out
$ mpirun -np 2 ph.x -in ph.in | tee ph.out
```

scf.in

```
--
&control
 calculation  = 'scf'
 restart_mode = 'from_scratch',
 prefix       = 'mgb2',
 pseudo_dir   = '../../pseudo/',
 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 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 jollowing jobscript (`job.epw1`) and input files (`nscf.in` and `epw1.in`):

```
$ cd ../epw1-FSR
$ mpirun -np 2 pw.x -in scf.in > scf.out
$ mpirun -np 2 pw.x -in nscf.in > nscf.out
$ mpirun -np 2 epw.x -in epw1.in > epw1.out
```

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

```
$ ./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`.

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`.

Note 4: The Migdal-Eliashberg equations are solved in the standard, Fermi surface restriction (FSR), approximation.

nscf.in

```
&control
  calculation  = 'nscf'
  prefix       = 'mgb2',
  pseudo_dir   = '../../pseudo/',
  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',
  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 irreducible 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` windown. 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            100
Number selected, total            200            204
......
Number selected, total           7800           7954
We only need to compute     7846 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 $\mathbf{k}$-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 =       446 out of      3234

Progression iq (fine) =        100/      7846
Progression iq (fine) =        200/      7846
....
....
Progression iq (fine) =       7800/      7846
             Fermi level (eV) =    0.746936938273072D+01
DOS(states/spin/eV/Unit Cell) =    0.324589885287221D+00
       Electron smearing (eV) =    0.500000000000000D-01
            Fermi window (eV) =    0.200000000000000D+00

Finish writing .ephmat files
```

- Write the Fermi surface files `mgb2.fs_YY.cube` (YY = band index within the `fsthick`) and `mgb2.fs.frmsf` by setting `fermi_plot = .true.`. The *.cube files can be visualized with VESTA and the *.frmsf file can be visualized with FermiSurfer.

```
 Fermi surface calculation on fine mesh
              Fermi level (eV) =   7.469369
       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:

$$\lambda_{n\mathbf{k},m\mathbf{k+q}}(\omega_j) = N_\mathrm{F} \sum_\nu \frac{2\omega_{\mathbf{q}\nu}}{\omega_j^2 + \omega_{\mathbf{q}\nu}^2} |g_{mn\nu}(\mathbf{k},\mathbf{q})|^2 \tag{10}$$

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

$$\lambda_{n\mathbf{k}}(\omega_j) = \sum_m \int \frac{d\mathbf{q}}{\Omega_\mathrm{BZ}} \frac{\delta(\epsilon_{m\mathbf{k+q}} - \epsilon_\mathrm{F})}{N_\mathrm{F}} \lambda_{n\mathbf{k},m\mathbf{k+q}}(\omega_j) \tag{11}$$

  The isotropic electron-phonon coupling strength takes the form:

$$\lambda(\omega_j) = \sum_n \int \frac{d\mathbf{k}}{\Omega_\mathrm{BZ}} \frac{\delta(\epsilon_{n\mathbf{k}} - \epsilon_\mathrm{F})}{N_\mathrm{F}} \lambda_{n\mathbf{k}}(\omega_j) \tag{12}$$

  The standard electron-phonon coupling strength λ found in the literature corresponds to setting $\omega_j = 0$ in Eq. (12).

  The isotropic Eliashberg spectral function takes the following form:

$$\alpha^2F(\omega) = \frac{1}{N_\mathrm{F}} \sum_{nm\nu} \int \frac{d\mathbf{k}}{\Omega_\mathrm{BZ}} \int \frac{d\mathbf{q}}{\Omega_\mathrm{BZ}} |g_{mn\nu}(\mathbf{k},\mathbf{q})|^2 \, \delta(\omega - \omega_{\mathbf{q}\nu}) \delta(\epsilon_{n\mathbf{k}} - \epsilon_\mathrm{F}) \delta(\epsilon_{m\mathbf{k+q}} - \epsilon_\mathrm{F}) \tag{13}$$

- Solve the anisotropic FSR 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 FSR Migdal-Eliashberg equations take the following form:

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

$$Z_{n\mathbf{k}}(i\omega_j)\Delta_{n\mathbf{k}}(i\omega_j) = \frac{\pi T}{N_\mathrm{F}}\sum_{mj'}\int\frac{d\mathbf{q}}{\Omega_\mathrm{BZ}}\frac{\Delta_{m\mathbf{k}+\mathbf{q}}(i\omega_{j'})}{\sqrt{\omega_{j'}^2+\Delta_{m\mathbf{k}+\mathbf{q}}^2(i\omega_{j'})}} \times\left[\lambda_{n\mathbf{k},m\mathbf{k}+\mathbf{q}}(\omega_j-\omega_{j'})-\mu_\mathrm{c}^*\right]\delta(\epsilon_{m\mathbf{k}+\mathbf{q}}-\epsilon_\mathrm{F}), \quad (14)$$

where $\lambda_{n\mathbf{k},m\mathbf{k}+\mathbf{q}}(\omega_j-\omega_{j'})$ 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 =    0.7115964

Estimated Allen-Dynes Tc =    35.325420 K for muc =    0.05000

Estimated w_log in Allen-Dynes Tc =    59.079741 meV

Estimated BCS superconducting gap =     5.357632 meV

Estimated Tc from machine learning model =    37.738498 K

WARNING WARNING WARNING

The code may crash since tempsmax =   55.000 K is larger than Allen-Dynes Tc =    35.325 K

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
broyden mixing factor =      0.70000
mixing factor = 0.2 is used for the first three iterations.

Actual number of frequency points (     1) =     92 for uniform sampling

Size of allocated memory per pool: ~=    0.0664 Gb
   iter      ethr          znormi      deltai [meV]
     1   2.676074E+00   1.669496E+00   6.099397E+00
     2   2.057049E-02   1.668907E+00   6.186415E+00
     ....
    13   4.794415E-04   1.662258E+00   6.916128E+00
Convergence was reached in nsiter =     13

Chemical potential (itemp =   1) =     7.4693693827E+00 eV

Temp (itemp =   1) =   10.000 K  Free energy =    -0.006786 meV

Min. / Max. values of superconducting gap =     0.000000   12.162768 meV
```

- 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]    Re[delta] [meV]
     82    1.692066E+00    6.378369E+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 $\lambda_{n\mathbf{q},m\mathbf{k}pq}(0)$ 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).

Note: First, put # in front of the 1st line and the last 7 lines of mgb2.a2f, otherwise gnuplot does not work.

You can use the `gnuplot` script `fig5.plt` to plot. You should get something similar to Fig. 1.

```
$ gnuplot fig5.plt
$ evince fig5.pdf
```

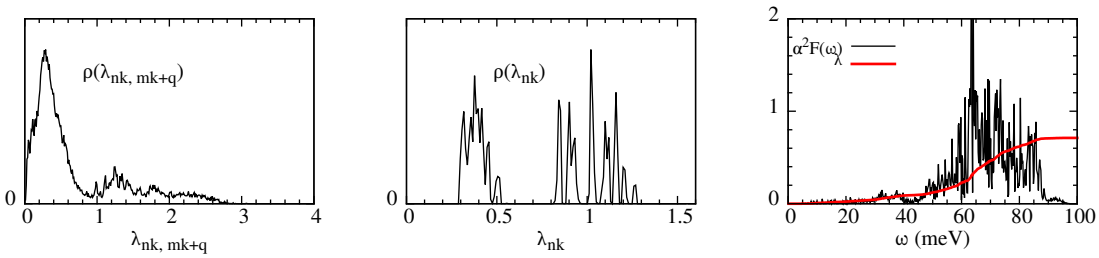


Fig. 5 Left: The anisotropic electron-phonon coupling strength $\lambda_{n\mathbf{q},m\mathbf{k}pq}(0)$ (from `mgb2.lambda_pairs`). Middle: The anisotropic electron-phonon coupling strength $\lambda_{n\mathbf{k}}(0)$ on the Fermi surface (from `mgb2.lambda_k_pairs`). Right: The isotropic Eliashberg spectral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

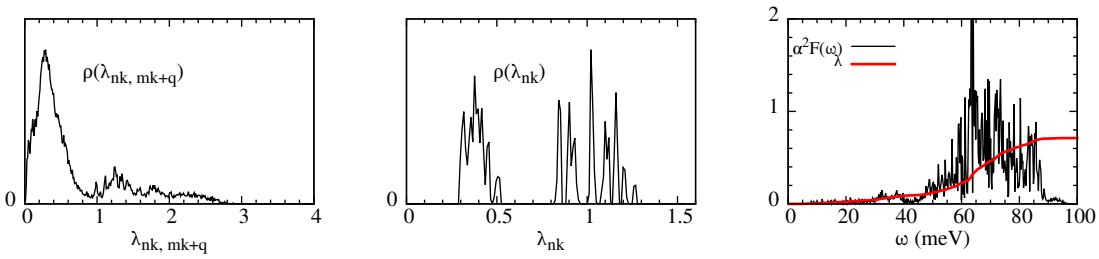


Fig. 6 At convergence you should get something close to this figure (see Phys. Rev. B **87**, 024505 (2013) for fully converged calculation parameters).

▶ 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 4 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 $Z_{n\mathbf{k}}(i\omega_j)$, and the superconducting gap $\Delta_{n\mathbf{k}}(i\omega_j)$ (eV).

`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 `gnuplot` script `fig7.plt` to plot. You should get something similar to Fig. 3 at 10 K. The file `fig7.pdf` is too large (10MB) to open while connecting to a remote server: To avoid opening it directly, you can use `pdftopng` command to show the plot.

```
$ gnuplot fig7.plt
$ pdftopng fig7.pdf fig7
$ display fig7-000001.png
```

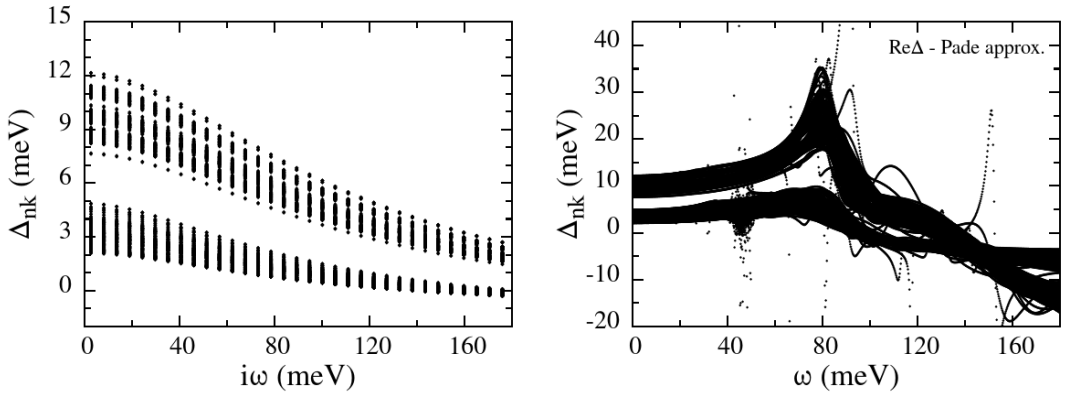


Fig. 7 Left: Superconducting gap along the imaginary axis (columns 1:4 from `mgb2.imag_aniso_010.00`). Right: Superconducting gap along the real axis (columns 1:5 from `mgb2.pade_aniso_010.00` - this file is about 70MB).

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

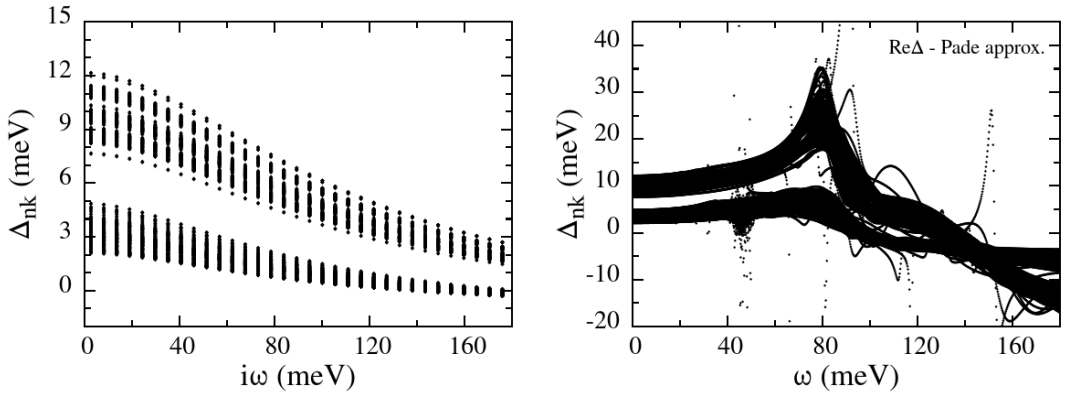


Fig. 8 At convergence you should get something close to this figure (see Phys. Rev. B **87**, 024505 (2013) for fully converged calculation parameters). (Note: Only about half of the points are shown.)

▶ 6th 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 `gnuplot` script `fig9.plt`.

```
$ gnuplot fig9.plt
$ evince fig9.pdf
```

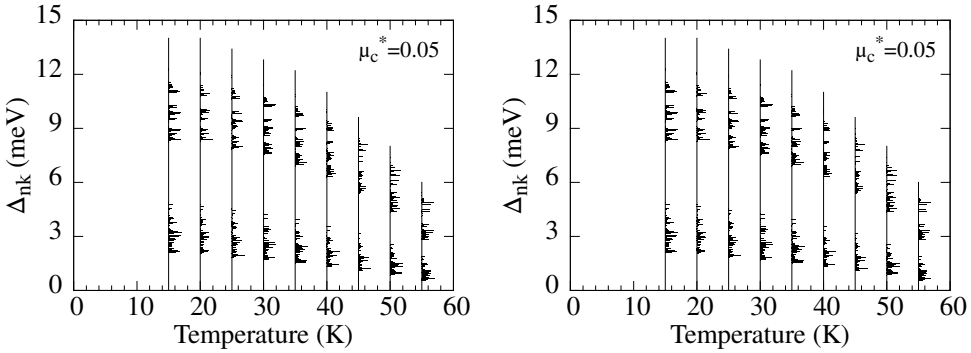


Fig. 9 Calculated anisotropic superconducting gap of MgB_2 on the Fermi surface as a function of temperature. At convergence you should get the right hand-side figure (see Phys. Rev. B **87**, 024505 (2013) for fully converged calculation parameters). (Note: the heights of the histograms are multiplied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

▶ 7th 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:

$$\frac{N_S(\omega)}{N_\mathrm{F}} = \sum_n \int \frac{d\mathbf{k}}{\Omega_\mathrm{BZ}} \frac{\delta(\epsilon_{n\mathbf{k}} - \epsilon_\mathrm{F})}{N_\mathrm{F}} \mathrm{Re}\left[\omega / \sqrt{\omega^2 - \Delta_{n\mathbf{k}}^2(\omega)}\right] \tag{15}$$

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

You can use the gnuplot script fig10.plt to plot mgb2.qdos_010.00. Edit fig10.plt to add the value of DOS in the normal state and run gnuplot.

fig10.plt

```
set terminal pdfcairo color dashed enhanced font "Times,25" fontscale 0.4 size 4,3 lw 2
...
set xrange [0:15]
set yrange [0:2.2]
set out "fig10.pdf"
set key at graph 0.9, 0.9
NF= 0.324589885287221           # DOS in the normal state
plot "mgb2.qdos_010.00" u ($1*1000):($2/NF) w l lw 2 lt 1 lc rgb "black" notitle
reset
```

```
$ gnuplot fig10.plt
$ evince fig10.pdf
```

You should get something similar to Fig. 6 (left) at 10 K:

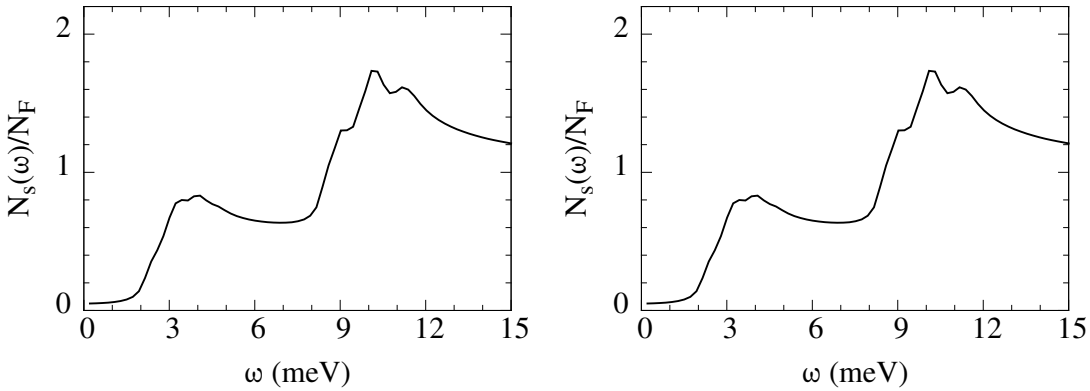


Fig. 10 Calculated $N_S(\omega)/N_F$ as a function of frequency at 10 K. At convergence you should get something closer to the right hand-side figure (see Phys. Rev. B **87**, 024505 (2013) for fully converged calculation parameters). (Note: the second column of mgb2.qdos_XX should be divided by the value of DOS from the epw1.out).

▶ 8th step: (Optional due to time limit) 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).

▶ 9th step: (Optional due to time limit) 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.

▶ 10th step: Solve the anisotropic full-bandwidth (FBW) Migdal-Eliashberg equations. The self-consistent and non self-consistent calculations are the same as for the standard FSR approximation, you can either copy the '../epw1-FSR/mgb2.save' directory or rerun the self-consistent and non self-consistent calculations. After this, do an EPW calculation using the following jobscript (job.epw2)

and input file (epw2.in; only differences with respect to ../epw1-FSR/epw1.in file are shown below):

Note: Here, we have fixed the chemical potential at the Fermi level. If you want to update the chemical potential at every temperature, set muchem = .true. in your EPW input file.

```
$ cd ../epw2-FBW
$ mpirun -np 2 pw.x -in scf.in | tee scf.out
$ mpirun -np 2 pw.x -in nscf.in | tee nscf.out
$ mpirun -np 2 epw.x -in epw2.in | tee epw2.out
```

```
--                                                                                    epw2.in
fbw       = .true.
```

The anisotropic FBW Migdal-Eliashberg equations are solved self-consistently on the imaginary frequency axis by setting the keywords fbw = .true., eliashberg = .true., laniso = .true., and limag = .true. in the EPW input file.

The anisotropic FBR Migdal-Eliashberg equations take the following form:

$$Z_{n\mathbf{k}}(i\omega_j) = 1 + \frac{T}{\omega_j N_\mathrm{F}} \sum_{mj'} \int \frac{d\mathbf{q}}{\Omega_\mathrm{BZ}} \frac{\omega_{j'} Z_{m\mathbf{k}+\mathbf{q}}(i\omega_{j'})}{\theta_{m\mathbf{k}+\mathbf{q}}(i\omega_{j'})} \lambda_{n\mathbf{k},m\mathbf{k}+\mathbf{q}}(\omega_j - \omega_{j'})$$

$$\chi_{n\mathbf{k}}(i\omega_j) = \frac{-T}{N_\mathrm{F}} \sum_{mj'} \int \frac{d\mathbf{q}}{\Omega_\mathrm{BZ}} \frac{\varepsilon_{m\mathbf{k}'} - \mu_\mathrm{F} + \chi_{m\mathbf{k}+\mathbf{q}}(i\omega_{j'})}{\theta_{m\mathbf{k}+\mathbf{q}}(i\omega_{j'})} \lambda_{n\mathbf{k},m\mathbf{k}+\mathbf{q}}(\omega_j - \omega_{j'})$$

$$\phi_{n\mathbf{k}}(i\omega_j) = \frac{T}{N_\mathrm{F}} \sum_{mj'} \int \frac{d\mathbf{q}}{\Omega_\mathrm{BZ}} \frac{\phi_{m\mathbf{k}'}(i\omega_{j'})}{\theta_{m\mathbf{k}+\mathbf{q}}(i\omega_{j'})} \left[\lambda_{n\mathbf{k},m\mathbf{k}+\mathbf{q}}(\omega_j - \omega_{j'}) - \mu_\mathrm{c}^*\right] \tag{16}$$

where $\lambda_{n\mathbf{k},m\mathbf{k}+\mathbf{q}}(\omega_j - \omega_{j'})$ is the anisotropic electron-phonon coupling strength. The superconducting gap is defined in terms of the renormalization function and the order parameter as: $\Delta_{n\mathbf{k}}(i\omega_j) = \phi_{n\mathbf{k}}(i\omega_j)/Z_{n\mathbf{k}}(i\omega_j)$. The semiempirical Coulomb parameter μ_c^* is provided as an input varible muc.

This set of equations is supplemented with an equation for the electron number N_e which determines the chemical potential μ_F if muchem = .true. is set in the EPW calculation.

$$N_\mathrm{e} = \sum_n \int \frac{d\mathbf{k}}{\Omega_\mathrm{BZ}} \left(1 - 2T \sum_j \frac{\varepsilon_{n\mathbf{k}} - \mu_\mathrm{F} + \chi_{n\mathbf{k}}(i\omega_j)}{\theta_{n\mathbf{k}}(i\omega_j)}\right) \tag{17}$$

Here, N_e is the number of electrons per unit cell.

```
 ================================================================
 Solve full-bandwidth anisotropic Eliashberg equations
 ================================================================
 ......
 temp(  1) =     10.00000 K

 Solve full-bandwidth anisotropic Eliashberg equations on imaginary-axis

 Total number of frequency points nsiw(     1) =     92
 Cutoff frequency wscut =     0.5008
 broyden mixing factor =      0.70000
 mixing factor = 0.2 is used for the first three iterations.
```

```
Actual number of frequency points (     1) =     92 for uniform sampling

Size of allocated memory per pool: ~=    0.0680 Gb
   iter      ethr          znormi      deltai [meV]   shifti [meV]       mu [eV]
     1   2.863885E+00   1.503340E+00   6.227374E+00   7.559550E-02   7.469369E+00
     2   2.182520E-02   1.493895E+00   6.362738E+00   2.284481E-01   7.469369E+00
     ......
     9   8.355396E-04   1.459439E+00   6.803335E+00   4.346811E-01   7.469369E+00
Convergence was reached in nsiter =      9

Chemical potential (itemp =   1) =     7.4693693827E+00 eV

Temp (itemp =   1) =   10.000 K  Free energy =    -0.010175 meV

Min. / Max. values of superconducting gap =     0.000000   13.062462 meV
```

▶ 11th step: To compare the results of the superconducting gap with those from the previous FSR calculation, follow the steps 5 and 6 above. You can use the `gnuplot` scripts `fig7-FBW.plt` and `fig9-FBW.plt`.

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 anisotropic Migdal-Eliashberg equations.
- If the anisotropic 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.

- muchem solve the anisotropic FBW ME eqs. with variable chemical potential.
- gridsamp = 0 generates a uniform Matsubara frequency grid (default).
- gridsamp = 1 generates a sparse Matsubara frequency grid.
- 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 anisotropic 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 theanisotropic 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.
```