2024 School on Electron-Phonon Physics, Many-Body Perturbation Theory, and Computational Workflows

Wannier function perturbation theory and phonon spectral function

Hands-on Session (Sat.5)

Hands-on based on Quantum ESPRESSO (v7.3.1) and EPW v5.9a

Introduction

In this tutorial, we show how to compute the phonon-induced renormalization of electron band structure of diamond within the Allen–Heine–Cardona theory[1] using Wannier Function perturbation theory (WFPT)[2], and the electron and phonon spectral functions of MgB_2.
You are advised to prepare the following script file, e.g. `run.sh`:

```
--                                                                                                    run.sh
#!/bin/bash
#SBATCH -J job.ph # Job name
#SBATCH -N 1 # Total # of nodes
#SBATCH --ntasks-per-node 8
#SBATCH -t 00:15:00 # Run time (hh:mm:ss)
#SBATCH -A DMR23030
#SBATCH -p skx
#SBATCH --reservation=NSF_Summer_School_Sat

# Launch MPI code...
export PATHQE=/work2/05193/sabyadk/stampede3/EPWSchool2024/q-e

ibrun -n 8 $PATHQE/bin/pw.x -nk 4 -in scf.in > scf.out
ibrun -n 8 $PATHQE/bin/ph.x -nk 4 -in ph.in > ph.out
```

Exercise 1: Zero-point renormalization and temperature-dependent band gap of diamond

1.1 Theory

In this example we calculate the phonon-induced band gap renormalization of diamond. In the Allen–Heine–Cardona (AHC) formalism[1], the phonon-induced renormalization of electron eigenvalue is determined by the real part of the electron self-energy, which is the sum of the Fan and the Debye–Waller (DW) terms:

$$\Sigma_{n\mathbf{k}} = \Sigma^{\text{Fan}}_{n\mathbf{k}} + \Sigma^{\text{DW}}_{n\mathbf{k}}. \tag{1}$$

The non-adiabatic Fan term comes from the second-order perturbative correction due to the linear electron-phonon coupling parameter and is given by[3]:

$$\Sigma^{\text{Fan}}_{n\mathbf{k}} = \int \frac{d^3q}{\Omega_{\text{BZ}}} \sum_{m\nu} |g_{mn\nu}(\mathbf{k},\mathbf{q})|^2 \sum_{\pm} \frac{n_{\mathbf{q}\nu} + f^{\pm}_{m\mathbf{k}+\mathbf{q}}}{\varepsilon_{n\mathbf{k}} - \varepsilon_{m\mathbf{k}+\mathbf{q}} \pm \omega_{\mathbf{q}\nu} + i\eta}, \tag{2}$$

[1] P. B. Allen and V. Heine, J. Phys. C, **9**, 2305 (1976), P. B. Allen and M. Cardona, Phys. Rev. B, **23**, 1495 (1981)
[2] J.-M. Lihm and C.-H. Park, Phys. Rev. X **11**, 041053 (2021)
[3] S. Poncé *et al.*, J. Chem. Phys. **143**, 102813 (2015)

where $f^{+}_{m\mathbf{k}+\mathbf{q}}$ is the Fermi–Dirac occupation factor of the state with energy $\varepsilon_{m\mathbf{k}+\mathbf{q}}$, $f^{-}_{m\mathbf{k}+\mathbf{q}} = 1 - f^{+}_{m\mathbf{k}+\mathbf{q}}$, and η a positive infinitesimal value (`degaussw`). This expression contains an infinite sum over bands m which cannot be computed using Wannier interpolation of low-energy bands.
To solve this problem, we use Wannier function perturbation theory (WFPT). We begin by splitting the Fan term into the low-energy active space contribution ($\Sigma^{\mathrm{A-Fan}}$) and the rest space contribution ($\Sigma^{\mathrm{R-Fan}}$) separated by the cutoff energy E^{A}:

$$\Sigma^{\mathrm{A-Fan}}_{n\mathbf{k}} = \int \frac{d^3q}{\Omega_{\mathrm{BZ}}} \sum_{m}^{\varepsilon_{m\mathbf{k}+\mathbf{q}} \leq E^{\mathrm{A}}} \sum_{\nu} |g_{mn\nu}(\mathbf{k},\mathbf{q})|^2 \sum_{\pm} \frac{n_{\mathbf{q}\nu} + f^{\pm}_{m\mathbf{k}+\mathbf{q}}}{\varepsilon_{n\mathbf{k}} - \varepsilon_{m\mathbf{k}+\mathbf{q}} \pm \omega_{\mathbf{q}\nu} + i\eta}, \tag{3}$$

$$\Sigma^{\mathrm{R-Fan}}_{n\mathbf{k}} = \int \frac{d^3q}{\Omega_{\mathrm{BZ}}} \sum_{m}^{\varepsilon_{m\mathbf{k}+\mathbf{q}} > E^{\mathrm{A}}} \sum_{\nu} |g_{mn\nu}(\mathbf{k},\mathbf{q})|^2 \frac{2n_{\mathbf{q}\nu} + 1}{\varepsilon_{n\mathbf{k}} - \varepsilon_{m\mathbf{k}+\mathbf{q}} \pm \omega_{\mathbf{q}\nu} + i\eta}. \tag{4}$$

In Eq. (4), we assumed that the cutoff energy is much higher than the Fermi level such that the electron occupation is always zero above the window. For states $n\mathbf{k}$ within the active space, $\varepsilon_{n\mathbf{k}} - \varepsilon_{m\mathbf{k}+\mathbf{q}}$ in the denominator of Eq. (4) is never zero. Assuming this electronic energy difference is much larger than the phonon frequency, we can drop the phonon frequency and approximate the rest term as

$$\Sigma^{\mathrm{R-Fan}}_{n\mathbf{k}} \approx \int \frac{d^3q}{\Omega_{\mathrm{BZ}}} \sum_{\nu} (2n_{\mathbf{q}\nu} + 1) \sum_{m}^{\varepsilon_{m\mathbf{k}+\mathbf{q}} > E^{\mathrm{A}}} \frac{|g_{mn\nu}(\mathbf{k},\mathbf{q})|^2}{\varepsilon_{n\mathbf{k}} - \varepsilon_{m\mathbf{k}+\mathbf{q}}}. \tag{5}$$

In practice, E^{A} (`ahc_win_max`) is a convergence parameter that needs to be increased until the result is converged. Usually, 1 eV above the highest band of interest is sufficient.
The infinite sum over the high-energy bands in Eq. (5) can be rewritten as[4]

$$\sum_{m}^{\varepsilon_{m\mathbf{k}+\mathbf{q}} > E^{\mathrm{A}}} \frac{|g_{mn\nu}(\mathbf{k},\mathbf{q})|^2}{\varepsilon_{n\mathbf{k}} - \varepsilon_{m\mathbf{k}+\mathbf{q}}} = \langle \partial^{\mathrm{R}}_{\mathbf{q}\nu} u_{n\mathbf{k}} | \partial^{\mathrm{R}}_{\mathbf{q}\nu} u_{n\mathbf{k}} \rangle, \tag{6}$$

where $|\partial^{\mathrm{R}}_{\mathbf{q}\nu} u_{n\mathbf{k}}\rangle$ is the first-order perturbation of the electron wavefunction projected onto the high-energy subspace, which can be computed by solving the Sternheimer equation

$$(\varepsilon_{n\mathbf{k}} - \hat{H}_{\mathbf{k}+\mathbf{q}}) |\partial^{\mathrm{R}}_{\mathbf{q}\nu} u_{n\mathbf{k}}\rangle = \hat{P}^{\mathrm{R}}_{\mathbf{k}+\mathbf{q}} \, \partial_{\mathbf{q}\nu} \hat{V} |u_{n\mathbf{k}}\rangle. \tag{7}$$

Here, $\hat{P}^{\mathrm{R}}_{\mathbf{k}+\mathbf{q}}$ is the projector onto the high-energy bands.
Using WFPT, the wavefunction perturbation $|\partial^{\mathrm{R}}_{\mathbf{q}\nu} u_{n\mathbf{k}}\rangle$ can be represented in terms of localized objects. These objects are called "Wannier function perturbations", as they are the change of Wannier functions due to a phonon perturbation. Correspoingly, the matrix element on the right-hand side of Eq. (6) can be Wannier interpolated from a coarse $\mathbf{k}$-point grid to real space, and then to a dense $\mathbf{k}$-point grid. More information about WFPT can be found in the article J. M. Lihm and C. H. Park, Phys. Rev. X **11**, 041053 (2021).

The DW term comes from the first-order perturbative correction due to the quadratic electron-phonon coupling parameter $D_{n\nu}(\mathbf{k},\mathbf{q})$:

$$\Sigma^{\mathrm{DW}}_{n\mathbf{k}} = \sum_{m\nu} \int \frac{d^3q}{\Omega_{\mathrm{BZ}}} \frac{n_{\mathbf{q}\nu} + 1/2}{2\omega_{\mathbf{q}\nu}} D_{n\nu}(\mathbf{k},\mathbf{q}). \tag{8}$$

The $D_{n\nu}(\mathbf{k},\mathbf{q})$ term, which is called the DW matrix element, is defined as:

$$D_{n\nu}(\mathbf{k},\mathbf{q}) = \sum_{\kappa\kappa'\alpha\alpha'} e^{*}_{\kappa\alpha\nu}(\mathbf{q}) e_{\kappa'\alpha'\nu}(\mathbf{q}) \langle u_{n\mathbf{k}} | \partial_{-\mathbf{q}\kappa\alpha} \partial_{\mathbf{q}\kappa'\alpha'} \hat{V} | u_{n\mathbf{k}} \rangle. \tag{9}$$

[4] X. Gonze *et al.*, Ann. Phys. 523, 168 (2011)

This term cannot be directly evaluated within first-order DFPT because it involves the second derivative of the electronic potential with respect to atomic displacements. Using the rigid-ion approximation which neglects the cross-derivatives between different atoms, and exploiting the translational covariance of the electron potential, the DW matrix element can be expressed as[5]

$$\langle u_{n\mathbf{k}}|\partial_{-\mathbf{q}\kappa\alpha}\partial_{\mathbf{q}\kappa'\alpha'}\hat{V}|u_{n\mathbf{k}}\rangle \overset{\text{RIA}}{\approx} i\delta_{\kappa\kappa'}\langle u_{n\mathbf{k}}|[\partial_{\mathbf{0}\kappa\alpha}\hat{V},\hat{p}_{\alpha'}]|u_{n\mathbf{k}}\rangle, \tag{10}$$

where $\hat{p}_{\alpha'} = -i\partial/\partial r_{\alpha'}$ is the momentum operator in the direction α'. Therefore, in the RIA, the DW self-energy takes the following explicit form:

$$\Sigma^{\text{DW}}_{n\mathbf{k}} \overset{\text{RIA}}{\approx} \sum_{m\nu}\int \frac{d^3q}{\Omega_{\text{BZ}}} \frac{n_{\mathbf{q}\nu}+1/2}{2\omega_{\mathbf{q}\nu}} \sum_{\kappa\alpha\alpha'} e^*_{\kappa\alpha\nu}(\mathbf{q})e_{\kappa\alpha'\nu}(\mathbf{q})\langle u_{n\mathbf{k}}|[\partial_{\mathbf{0}\kappa\alpha}\hat{V},\hat{p}_{\alpha'}]|u_{n\mathbf{k}}\rangle\,. \tag{11}$$

Note the κ, the atomic index, is the same for the two eigenmodes as the RIA neglects inter-atomic cross-derivatives.

1.2 Preliminary calculations with Quantum Espresso

First download the exercise files:

```
$ cd $SCRATCH
$ cp /work2/05193/sabyadk/stampede3/EPWSchool2024/tutorials/Sat.5.Lihm.tar .
$ tar -xvf Sat.5.Lihm.tar
$ cd Sat.5.Lihm/exercise1/
```

▶ Make a self-consistent calculation for diamond.

scf.in

```
--
&control
   calculation  = 'scf'
   prefix       = 'diamond'
   pseudo_dir   = './'
   outdir       = './'
/
&system
   ibrav     = 2
   celldm(1) = 6.74
   nat       = 2
   ntyp      = 1
   ecutwfc   = 40.0
/
&electrons
   diagonalization = 'david'
   mixing_beta     = 0.7
   conv_thr        = 1.0d-13
/
ATOMIC_SPECIES
  C  12.011   C.pz-vbc.UPF
ATOMIC_POSITIONS crystal
  C  0.125   0.125   0.125
  C -0.125  -0.125  -0.125
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. (Here, the Born effective charges are 0 as diamond is infrared inactive.)

[5] J.-M. Lihm and C.-H. Park, Phys. Rev. B **101**, 121102(R) (2020)

```
$ ibrun -n 8 $PATHQE/bin/pw.x -nk 4 -in scf.in > scf.out
```

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

```
--                                                                                ph.in
&inputph
   prefix    = 'diamond'
   reduce_io = .true.
   epsil     = .true.
   fildyn    = 'diamond.dyn.xml'
   ldisp     = .true.
   fildvscf  = 'dvscf'
   tr2_ph    = 1.d-20
   nmix_ph   = 12
   nq1 = 4
   nq2 = 4
   nq3 = 4
   /
```

Note: We have the input variable `reduce_io = .true.` which reduces the I/O operations at the cost of higher memory usage.
Note 2: If you add `.xml` after the name of the dynamical matrix file, it will produce the data in XML format (preferred).
Note 3: The input variable responsible to produce the electron-phonon matrix element is `fildvscf`. Always make sure that this variable is present.
Note 4: Notice the very tight `tr2_ph` threshold parameter on the self-consistent first-order perturbed wavefunction. This is crucial to obtain good vibrational properties.
Note 5: The input variable `nmix_ph = 12` improves the convergence of the self-consistent DFPT at the cost of a small increase in memory usage, compared to the default of `nmix_ph = 4`.

```
$ ibrun -n 8 $PATHQE/bin/ph.x -nk 4 ph.in > ph.out
```

The calculation should take about 1 min on 8 cores.

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

```
$ python3 $PATHQE/EPW/bin/pp.py
```

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

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 follows:

```
--                                                                                nscf.in
&control
      calculation  = 'bands'
      prefix       = 'diamond'
      pseudo_dir   = './'
      outdir       = './'
/
&system
      ibrav     = 2
      celldm(1) = 6.74
```

```
      nat       = 2
      ntyp      = 1
      ecutwfc   = 40.0
      nbnd      = 8
/
&electrons
      conv_thr = 1.d-12
/
ATOMIC_SPECIES
   C  12.011   C.pz-vbc.UPF
ATOMIC_POSITIONS crystal
   C  0.125   0.125   0.125
   C -0.125  -0.125  -0.125
K_POINTS crystal
64
   0.000000000000  0.000000000000  0.000000000000  1.56250000e-02
   0.000000000000  0.000000000000  0.250000000000  1.56250000e-02
   0.000000000000  0.000000000000  0.500000000000  1.56250000e-02
...
```

```
$ ibrun -n 8 $PATHQE/bin/pw.x -nk 4 -in nscf.in > 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 1: Since we are interested in the band gap, we will need the conduction bands. Notice that we added the input `nbnd = 8` in `nscf.in`

Note 2: We use `calculation = 'bands'` instead of `calculation = 'nscf'`. This is because the latter option may expand the set of $\mathbf{k}$-points according to the symmetry, while the former always computes the eigenenergies and wavefunctions for the given set of $\mathbf{k}$-points.

▶ Compute the matrix elements for the DW and rest-space Fan self-energy on the coarse $\mathbf{k}$-point grid

For WFPT, we need matrix elements for the DW and rest-space Fan self-energy on the coarse $\mathbf{k}$-point grid. To compute these, we use the `electron_phonon = 'ahc'` option of the `ph.x` code.
The `ahc.in` file is as follows:

```
--                                                                                ahc.in
&inputph
  prefix    = 'diamond'
  reduce_io = .true.
  epsil     = .true.
  fildyn    = 'diamond_ahc.dyn.xml'
  ldisp     = .true.
  fildvscf  = 'dvscf'
  nq1 = 4
  nq2 = 4
  nq3 = 4

  ! Input variables for Allen-Heine-Cardona calculation
  electron_phonon = 'ahc'
  trans           = .false.
  ahc_nbnd        = 8
  ahc_dir         = './save/ahc_dir/'
 /
```

There are four new input variables specific to the AHC calculation:
Note 1: `electron_phonon = 'ahc'` enables the calculation of the Sternheimer and DW matrix elements.
Note 2: `trans = .false.` disables the computation of phonons, as the phonon potential will be read from file.
Note 3: `ahc_nbnd = 8` specifies the number of bands to be considered in the calculation of the matrix elements.
Note 4: `ahc_dir = './save/ahc_dir/'` specifies the directory where the matrix elements will be stored.

```
$ ibrun -n 8 $PATHQE/bin/ph.x -nk 4 -in ahc.in > ahc.out
```

▶ 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`.

```
--                                                                                                    epw.in
&inputepw
  prefix      = 'diamond'
  amass(1)    = 12.011
  outdir      = './'
  dvscf_dir   = './save/'
  elph        = .true.
  etf_mem     = 0
  lopt_w2b    = .true.

  ! --- Input variables for WFPT ---
  lwfpt       = .true.
  ahc_nbnd    = 8
  ahc_win_max = 23.0
  ahc_win_min = -100.0

  ! --- Input variables for electron self-energy ---
  elecselfen = .true.
  filkf      = './kpt.txt'
  nqf1       = 10
  nqf2       = 10
  nqf3       = 10
  degaussw   = 0.02  ! eV
  temps      = 0.0   ! Kelvin
  ! ------------------------------------------------

  efermi_read = .true.
  fermi_energy = 15.0

  use_ws = .true.
  lpolar = .true.
  vme    = 'wannier'

  ! For the first-time, perform Wannierization of the electrons
  ! and the electron-phonon matrix elements
  wannierize = .true.
  epbwrite = .true.
  epwwrite = .true.

  ! To restart from the real-space matrix elements
  !wannierize = .false.
  !epwread = .true.

  nbndsub     = 8
  num_iter    = 200
  proj(1)     = 'C:sp3'
  wdata(1)    = 'conv_tol = 1.d-10'
  wdata(2)    = 'conv_window = 5'

  nk1         = 4
  nk2         = 4
  nk3         = 4

  nq1         = 4
  nq2         = 4
  nq3         = 4
 /
```

```
$ ibrun -n 8 $PATHQE/bin/epw.x -nk 8 -in epw.in > epw.out
```

Notes:

- We set `etf_mem = 0` to minimize the number of IO operations. For larger systems where memory becomes a concern, you can set `etf_mem = 1`.
- `lopt_w2b = .true.` enables the optimization of the Wannier interpolation that reduces a 3D Fourier transformation to a nested 1D Fourier transformation with cached intermediate results. This option can significantly improve performance *without* compromising any accuracy when using homogeneous $\mathbf{q}$-point grids. See Eqs. (8,9) of J. Kaye *et al.*, SciPost Phys. 15, 062 (2023).
- `lwfpt` enables WFPT.
- `ahc_nbnd` specifies the number of bands for which the Sternheimer matrix element is computed. This number should be the same as in the `ahc.in` file.
- `ahc_win_min` and `ahc_win_max` specifies the lower and upper boundaries of the active space.
- `elecselfen` asks for the electron self-energy to be computed.
- `degaussw = 0.02` specifies the smearing parameter η in the units of eV.
- `temps` define the lattice temperature at which the self-energy is evaluated.
- 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`.
- If you run the code for the second time, you can skip Wannierization and the calculation of the real-space electron-phonon matrix elements by uncommenting the lines `wannierize = .false.` and `epwread = .true.`, and commenting out the three lines above them, `wannierize = .true.`, `epbwrite = .true.`, and `epwwrite = .true.`.

The calculation should take less than 2 min. The fine $\mathbf{q}$ point grids need to be much denser for real calculations. However, we can already get relatively decent results.

In the output you will see the line '`ik = 1 coord.: 0.0000000 0.0000000 0.0000000`'. The data following this line show the real and imaginary parts of the electron self-energy. (Note that below we have trimmed some digits for brevity. The actual output has more digits printed.) The real part is the total electron band structure renormalization, which is the sum of the active space Fan, rest space Fan, and DW terms. From the underlined numbers, we find that the renormalization of the valence band maxima (VBM), lowest conduction band at $\mathbf{k} = \Gamma$, and conduction band minima (CBM) are 124.73, -350.68, and -167.00 meV, respectively.

```
ik =       1 coord.:    0.0000000   0.0000000   0.0000000
-------------------------------------------------------------------
E(   1 )=  -23.097318 eV   Re[Sigma]= -0.64319...E+02 meV Im[Sigma]=  0.59261...E+00 meV     Z= ...
E(   2 )=   -1.641768 eV   Re[Sigma]=  0.12473...E+03 meV Im[Sigma]=  0.13815...E+01 meV     Z= ...
E(   3 )=   -1.641768 eV   Re[Sigma]=  0.12473...E+03 meV Im[Sigma]=  0.13815...E+01 meV     Z= ...
E(   4 )=   -1.641768 eV   Re[Sigma]=  0.12473...E+03 meV Im[Sigma]=  0.13815...E+01 meV     Z= ...
E(   5 )=    4.020640 eV   Re[Sigma]= -0.35068...E+03 meV Im[Sigma]=  0.91491...E+02 meV     Z= ...
E(   6 )=    4.020640 eV   Re[Sigma]= -0.35068...E+03 meV Im[Sigma]=  0.91491...E+02 meV     Z= ...
E(   7 )=    4.020640 eV   Re[Sigma]= -0.35068...E+03 meV Im[Sigma]=  0.91491...E+02 meV     Z= ...
E(   8 )=   11.517629 eV   Re[Sigma]=  0.00000...E+00 meV Im[Sigma]=  0.24628...E+00 meV     Z= ...
-------------------------------------------------------------------


ik =       2 coord.:    0.3500000   0.0000000   0.3500000
-------------------------------------------------------------------
E(   1 )=  -18.740849 eV   Re[Sigma]= -0.74350...E+01 meV Im[Sigma]=  0.33020...E+02 meV     Z=  ...
E(   2 )=   -9.321123 eV   Re[Sigma]= -0.38989...E+01 meV Im[Sigma]=  0.71024...E+02 meV     Z=  ...
E(   3 )=   -7.102523 eV   Re[Sigma]= -0.55960...E+01 meV Im[Sigma]=  0.13225...E+03 meV     Z=  ...
E(   4 )=   -7.102523 eV   Re[Sigma]= -0.55960...E+01 meV Im[Sigma]=  0.13225...E+03 meV     Z=  ...
E(   5 )=    2.239900 eV   Re[Sigma]= -0.16700...E+03 meV Im[Sigma]=  0.11050...E+01 meV     Z=  ...
E(   6 )=    5.097206 eV   Re[Sigma]= -0.14679...E+03 meV Im[Sigma]=  0.21412...E+03 meV     Z=  ...
E(   7 )=   13.403539 eV   Re[Sigma]=  0.00000...E+00 meV Im[Sigma]=  0.21549...E+00 meV     Z=  ...
E(   8 )=   13.403539 eV   Re[Sigma]=  0.00000...E+00 meV Im[Sigma]=  0.21549...E+00 meV     Z=  ...
-------------------------------------------------------------------
```

A more detailed analysis of the electron self-energy can be found in the file `elself_wfpt_sup.0.000K`. Each of the column shows the contribution of each electron self-energy term: active-space Fan

(Eq. (3)), active-space DW, rest-space Fan (Eq. (4)), and rest-space DW. One can find that the rest-space Fan and DW terms have large values with opposite signs. (Again, we have removed some digits for brevity.)

```
# Electron self-energy (meV) in the Allen-Heine-Cardona formalism at T = 0.000K
#   ik  ibnd    E_nk (eV)  Re[Active_Fan]  Active_DW  Rest_Fan    Rest_DW      Im[Active_Fan] (meV)
     1     1 -0.80973E+01 -0.71604E+02  0.87182E+01 -0.10433E+03  0.10289E+03  0.59261E+00
     2     1 -0.37408E+01 -0.50846E+01  0.58749E+01 -0.19203E+03  0.18380E+03  0.33020E+02

     1     2  0.13358E+02  0.97221E+02  0.28803E+02 -0.10153E+04  0.10140E+04  0.13815E+01
     2     2  0.56788E+01 -0.10493E+02  0.16287E+02 -0.48840E+03  0.47871E+03  0.71024E+02

     1     3  0.13358E+02  0.97221E+02  0.28803E+02 -0.10153E+04  0.10140E+04  0.13815E+01
     2     3  0.78974E+01 -0.30455E+02  0.84617E+02 -0.80246E+03  0.74271E+03  0.13225E+03

     1     4  0.13358E+02  0.97221E+02  0.28803E+02 -0.10153E+04  0.10140E+04  0.13815E+01
     2     4  0.78974E+01 -0.30455E+02  0.84617E+02 -0.80246E+03  0.74271E+03  0.13225E+03

     1     5  0.19020E+02 -0.25361E+03 -0.31710E+02 -0.99682E+03  0.93145E+03  0.91491E+02
     2     5  0.17239E+02 -0.31152E+02 -0.81005E+02 -0.41128E+03  0.35643E+03  0.11050E+01

     1     6  0.19020E+02 -0.25361E+03 -0.31710E+02 -0.99682E+03  0.93145E+03  0.91491E+02
     2     6  0.20097E+02  0.62796E+02 -0.11039E+03 -0.31412E+03  0.21492E+03  0.21412E+03

     1     7  0.19020E+02 -0.25361E+03 -0.31710E+02 -0.99682E+03  0.93145E+03  0.91491E+02
     2     7  0.28403E+02  0.87698E+02 -0.17944E+02  0.00000E+00  0.00000E+00  0.21549E+00

     1     8  0.26517E+02  0.83752E+02 -0.15407E+03  0.00000E+00  0.00000E+00  0.24628E+00
     2     8  0.28403E+02  0.87698E+02 -0.17944E+02  0.00000E+00  0.00000E+00  0.21549E+00
```

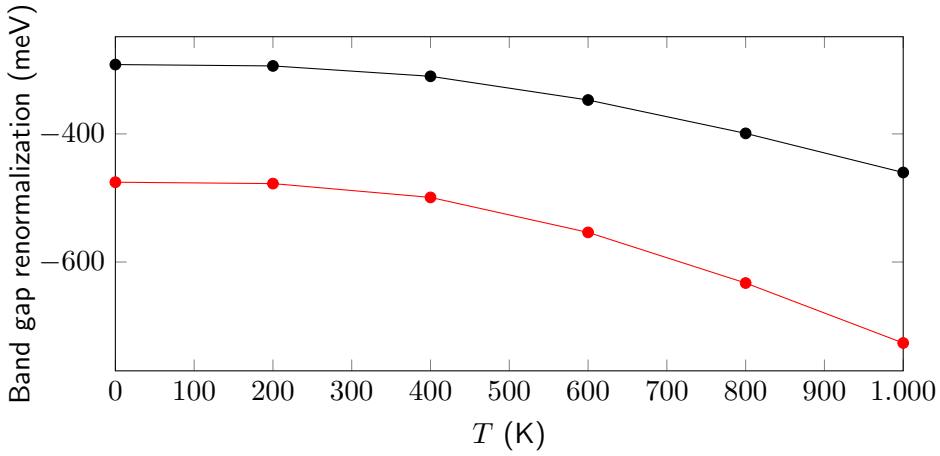
▶ Verify that the sum of the four terms equals the total electron self-energy shown in `epw.out`.

▶ Re-run the code with multiple temperatures (using `temps = 0.0 200 400 600 800 1000`). You should remove the `restart.fmt` file before doing so.

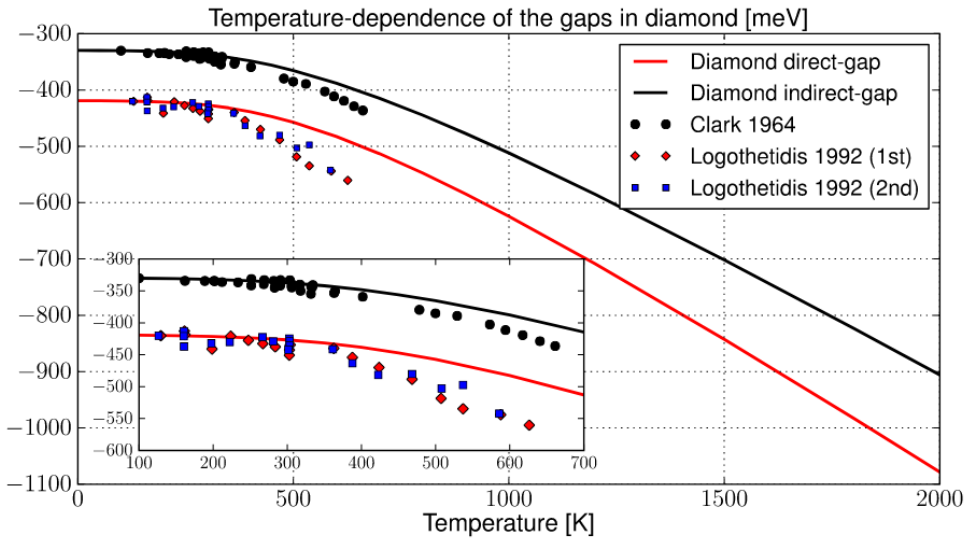
Try filling the table below for the band gap renormalization:

T (K)	VBM (meV)	CBM (meV)	CB-Γ (meV)	Direct gap (meV)	Indirect gap (meV)
0					
200					
400					
600					
800					
1000					

With our settings, you should get the following figure (black: indirect gap, red: direct gap):



At convergence you should get [6]:



▶ Try to decompose the energy and band gap renormalization into the Fan and DW terms.
▶ Try to increase the fine grids and add a few more temperatures and see if you can get a result closer to convergence.
▶ Try to change `ahc_win_max` and see the impact on the results.
▶ Try removing or renaming the file `quadrupole.fmt` to do the interpolation without quadrupoles and see the impact on the results.
▶ Try removing the `lopt_w2b = .true.` option and compare the performance. In particular, compare the time spent in the `ephW2Bp_opt` (run with optimization) and `ephW2Bp` (run without optimization) steps reported in the end of the `epw.out` file.

[6] The figure is from S. Poncé *et al.*, J. Chem. Phys. **143**, 102813 (2015)

Exercise 2: Electronic spectral function of MgB_2

In this example, we calculate the electron spectral function of MgB_2 using WFPT. We mostly highlight the specificity of spectral function calculations and it is therefore advised to first do exercise 1.

First go in the second exercise:

```
$ cd exercise2
```

▶ Make a self-consistent calculation for MgB_2 and a phonon calculation on a homogeneous 3×3×3 **q**-point grid.

```
$ ibrun -n 8 $PATHQE/bin/pw.x -nk 4 -in scf.in > scf.out
$ ibrun -n 8 $PATHQE/bin/ph.x -nk 4 -in ph.in > ph.out
```

scf.in

```
--
&control
   calculation  = 'scf'
   restart_mode = 'from_scratch'
   prefix       = 'mgb2'
   pseudo_dir   = './'
   outdir       = './'
/
&system
   ibrav        = 4
   celldm(1)    = 5.8260252227888
   celldm(3)    = 1.1420694129095
   nat          = 3
   ntyp         = 2
   ecutwfc      = 40
   smearing     = 'mp'
   occupations  = 'smearing'
   degauss      = 0.05
/
&electrons
   diagonalization = 'david'
   mixing_beta     = 0.7
   conv_thr        = 1.0d-12
/
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
```

ph.in

```
--
&inputph
    prefix    = 'mgb2'
    fildyn    = 'mgb2.dyn.xml'
    fildvscf  = 'dvscf'
    tr2_ph    =  1.0d-20
    ldisp     = .true.
    reduce_io = .true.
    nmix_ph   = 12
    nq1 = 3
    nq2 = 3
    nq3 = 3
/
```

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
(   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
```

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 $PATHQE/EPW/bin/pp.py
```

The script will ask you to enter the prefix used for the calculation. In this case enter "mgb2". The script will create a new folder called "save" that contains the dvscf potential files, pattern files, and dynamical matrices on the IBZ.

▶ Do a non self-consistent calculation on a homogeneous $6\times6\times6$ **uniform and Γ-centered grid between [0,1) in crystal coordinates**.

nscf.in

```
--
&control
    calculation = 'bands'
    prefix      = 'mgb2'
    pseudo_dir  = './'
    outdir      = './'
    verbosity   = 'high'
/
&system
    ibrav       = 4
    celldm(1)   = 5.8260252227888
    celldm(3)   = 1.1420694129095
    nat         = 3
    ntyp        = 2
    ecutwfc     = 40
    smearing    = 'mp'
    occupations = 'smearing'
    degauss     = 0.05
    nbnd        = 12
/
&electrons
    diagonalization = 'david'
    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
0.00000000  0.00000000  0.33333333  4.629630e-03
...
```

```
$ ibrun -n 8 $PATHQE/bin/pw.x -nk 4 -in nscf.in > nscf.out
```

▶ Compute the matrix elements for the DW and rest-space Fan self-energy on the coarse $\mathbf{k}$-point grid

```
--                                                                                ahc.in
&inputph
  prefix    = 'mgb2'
  reduce_io = .true.
  fildyn    = 'mgb2_ahc.dyn.xml'
  ldisp     = .true.
  fildvscf  = 'dvscf'
  nq1 = 3
  nq2 = 3
  nq3 = 3

  ! Input variables for Allen-Heine-Cardona calculation
  electron_phonon = 'ahc'
  trans           = .false.
  ahc_nbnd        = 9
  ahc_dir         = './save/ahc_dir/'
 /
```

Note: We set `ahc_nbnd = 9`, although we compute `nbnd = 12` bands in `nscf.in`. This means that we compute the Sternheimer matrix elements for bands 1 to 9, and skip the calculation for bands 10 to 12. Bands 10 to 12 will be excluded from Wannierization via the input variable `bands_skipped` in `epw.in`.

```
$ ibrun -n 8 $PATHQE/bin/ph.x -nk 4 -in ahc.in > ahc.out
```

▶ Perform an EPW calculation to transform electron-phonon matrix elements from a coarse $3\times3\times3$ $\mathbf{q}$-grid in momentum space to real space in the Wannier function basis.

```
--                                                                                epw1.in
&inputepw
  prefix      = 'mgb2'
  outdir      = './'
  dvscf_dir   = './save'
  etf_mem     =  0

  elph        = .true.
  use_ws      = .true.
  epbwrite    = .true.
  epwwrite    = .true.

  ! --- Input variables for WFPT ---
  lwfpt       = .true.
  ahc_nbnd    = 9
  ahc_win_max = 8.8
  ahc_win_min = -100.0
  ! ---------------------------------

  wannierize  = .true.
  nbndsub     =  5

  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'
  bands_skipped = 'exclude_bands = 10-12'
```

```
  nk1         = 6
  nk2         = 6
  nk3         = 6

  nq1         = 3
  nq2         = 3
  nq3         = 3

  nkf1        = 1
  nkf2        = 1
  nkf3        = 1

  nqf1        = 1
  nqf2        = 1
  nqf3        = 1
/
```

```
$ ibrun -n 8 $PATHQE/bin/epw.x -nk 8 -in epw.in > epw.out
```

Notes:

- `ahc_nbnd = 9` is the same as in `ahc.in`.
- `ahc_win_max` should be equal to or smaller than `dis_froz_max`, the upper bound of the frozen window for Wannierization.

▶ Perform an EPW restart calculation to obtain the electron self-energy and spectral function for **k** along the same high-symmetry path

```
--                                                                                    epw2.in
&inputepw
  prefix      = 'mgb2'
  outdir      = './'
  dvscf_dir   = './save'
  etf_mem     =  0
  lopt_w2b    = .true.

  elph        = .true.
  use_ws      = .true.
  epwread     = .true.

  ! --- Input variables for WFPT ---
  lwfpt       = .true.
  ahc_nbnd    = 9
  ahc_win_max = 8.8
  ahc_win_min = -100.0

  ! --- Input variables for electron self-energy ---
  elecselfen = .true.
  filkf      = './kpt.txt'
  nqf1       = 10
  nqf2       = 10
  nqf3       = 10
  degaussw   = 0.10  ! eV
  temps      = 300.0 ! Kelvin

  ! --- Input variables for electron spectral function ---
  specfun_el   = .true.
  wmax_specfun =   1.5
  wmin_specfun = -13.0
  nw_specfun   = 200
  ! ------------------------------------------------------

  nbndsub     =  5

  efermi_read  = .true.
  fermi_energy = 7.4502
```

```
  nk1         = 6
  nk2         = 6
  nk3         = 6

  nq1         = 3
  nq2         = 3
  nq3         = 3
/
```

```
$ ibrun -n 8 $PATHQE/bin/epw.x -nk 8 -in epw2.in > epw2.out
```

Notes:

- We use `efermi_read = .true.` and provide the Fermi energy as `fermi_energy = 7.4502` (eV) since we are using $\mathbf{k}$-points along a high-symmetry path. Otherwise, the code will compute the Fermi energy using the $\mathbf{k}$ points given by the `filkf` file. Since this file does not sample the Brillouin zone uniformly, the Fermi energy is incorrect.
- `elecselfen` asks for the electron self-energy to be computed.
- `specfun_el` asks for the electron spectral function to be computed.
- The spectral function is computed for a uniform grid of frequencies between `wmin_specfun = -13.0` (eV) and `wmax_specfun = 1.5` (eV) with `nw_specfun = 200` points. The energy zero is set to `fermi_energy = 7.4502` (eV).

The code will produce two additional output files, `specfun.elself.300.000K` and `specfun_sup.elself.300.000K`.
The file `specfun.elself.300.000K` contains the spectral function data:

```
#Electronic spectral function (meV)


#K-point     Energy[eV]     A(k,w)[meV^-1]

     1   -13.00000   0.60715E-06
     1   -12.92714   0.83532E-06
     1   -12.85427   0.12018E-05
      ...
# Integrated spectral function    0.12028E+00
      ...
```

The first column is the $\mathbf{k}$-point index, the second column is the energy ω in meV (with the reference zero at the Fermi level), and the third column is the total spectral function $\sum_n A_{n\mathbf{k}}(\omega)$ in meV^{-1}. After all the frequency points are printed for a given $\mathbf{k}$-point, the integrated spectral function, which is the number of occupied electrons *within the frequency window* is printed. Importantly, this does not count the electrons below `wmin_specfun`.

The file `specfun_sup.elself.300.000K` contains the frequency-dependent self-energy:

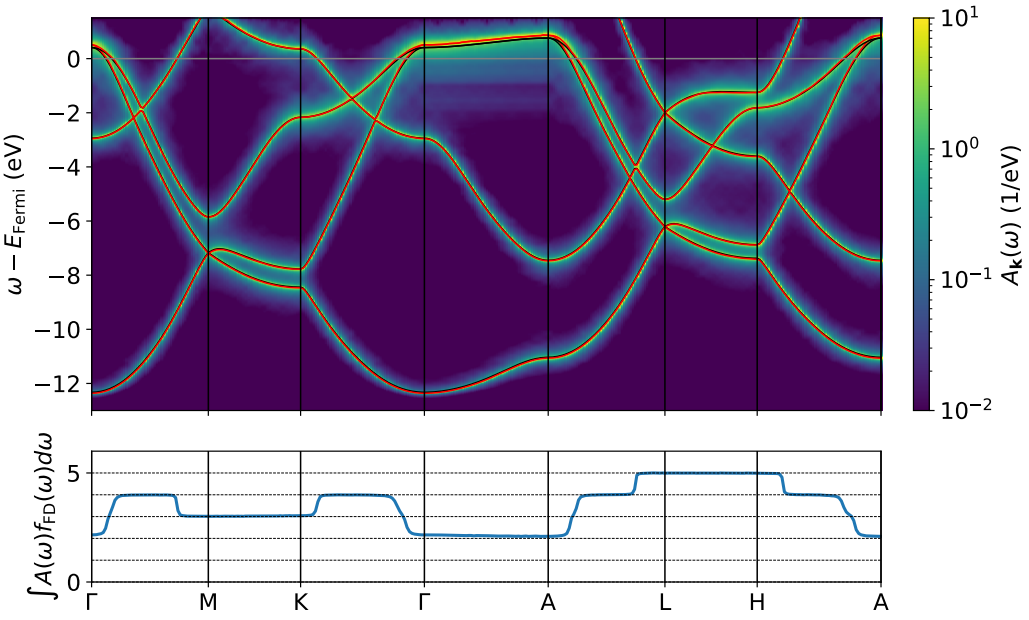
```
#KS eigenenergies + real and im part of electronic self-energy (meV)


#K-point    Band   e_nk[eV]   w[eV]      Real Sigma[meV]  Im Sigma[meV]

      1        1      -12.3573      -13.0000      -29.7716        0.7026
      1        1      -12.3573      -12.9271      -30.2972        0.7516
      ...
      2        1      -12.3551      -13.0000      -29.7596        0.7026
      2        1      -12.3551      -12.9271      -30.2852        0.7516
      ...
```

For each bands and $\mathbf{k}$ points, the diagonal part of the frequency-dependent self-energy $\Sigma_{n\mathbf{k}}(\omega)$ is printed.

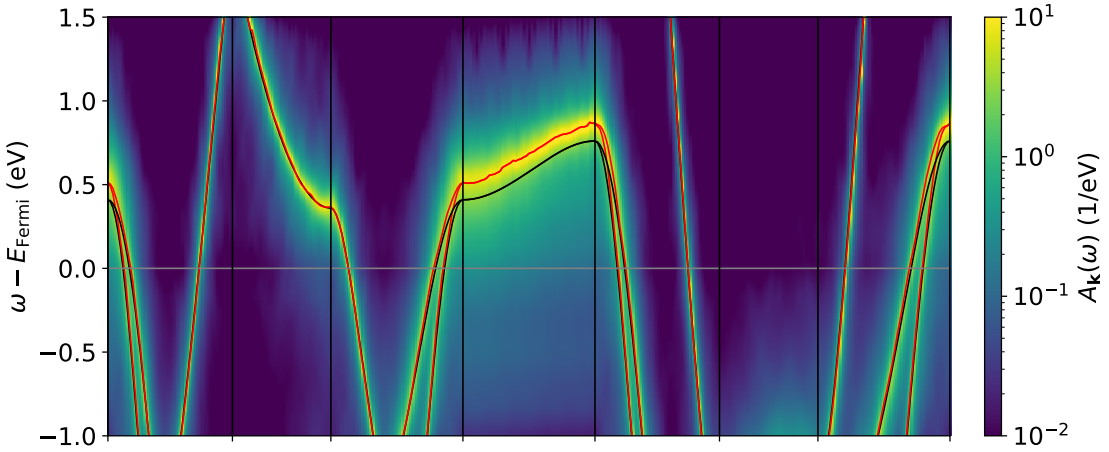
The computed spectral function can be visualized using the `plot.py` script. Since the spectral function has sharp peaks at the quasiparticle energy, we linearly interpolate the self-energy to a denser frequency grid before computing the spectral function. (See Appendix 1 for the full script.) To use matplotlib, we should first load the module and install the matplotlib package.

```
$ module load python/3.9.18
$ python3 -m pip install matplotlib --user
$ python3 plot.py
```

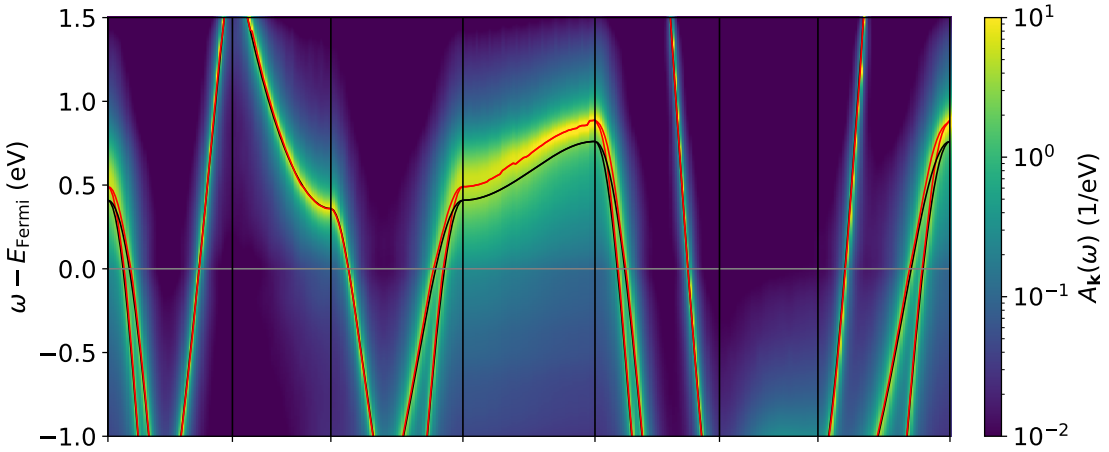
The script will create two pdf files, `mgb2_spectral.pd` and `mgb2_spectral_zoom.pdf`. You may view them using evince or by downloading them to your laptop.



The upper panel shows the electron spectral function as the background color. The black and red curves show the bare and renormalized electron dispersion.
The lower panel shows the electron occupation at each $\mathbf{k}$-point, which is the integral of the spectral function multiplied by the Fermi–Dirac function $f_{\mathrm{FD}}(\omega) = \frac{1}{e^{(\omega-\mu)/k_{\mathrm{B}}T}+1}$. This value changes when the dispersion crosses the Fermi level.
Zooming in near the Fermi level, we can see the phonon induced renormalization of the electron dispersion: the red curve is the renormalized band structure, and the black curve is the bare band structure.



The calculation is not fully converged and the spectral function is not yet smooth. Using a denser $20\times20\times20$ $\mathbf{q}$-point grid and 500 frequency points, we have:



Exercise 3: Phononic spectral function of MgB_2

In this example, we calculate the phonon spectral function of MgB_2. We mostly highlight the specificity of phonon spectral function calculations and it is therefore advised to first do exercise 2.
The 1-loop (1L) contribution to the phonon self-energy is given by[7]

$$\Pi^{\mathrm{1L}}_{\mathbf{q}\nu}(\omega;T) = \frac{g_\mathrm{s}}{N_k}\sum_{mn\mathbf{k}}\frac{f_{n\mathbf{k}}(T)-f_{m\mathbf{k}+\mathbf{q}}(T)}{\varepsilon_{n\mathbf{k}}-\varepsilon_{m\mathbf{k}+\mathbf{q}}+\omega+i\eta}|g_{mn\nu}(\mathbf{k},\mathbf{q})|^2. \tag{12}$$

Here, g_s is the spin degeneracy factor. In principle, the self-energy $\Pi^{\mathrm{1L}}_{\mathbf{q}\mu\nu}(\omega;T)$ also has off-diagonal terms that couple different phonon modes and change the phonon eigenmodes[8], but we neglect them here: $\Pi^{\mathrm{1L}}_{\mathbf{q}\mu\nu} \triangleq \Pi^{\mathrm{1L}}_{\mathbf{q}\nu}\delta_{\mu\nu}$. Calculation of the off-diagonal self-energy will be made available in the EPW code in the near future.

Since the DFPT calculation already includes static screening, we have to subtract the static DFPT contribution from the phonon self-energy to avoid double counting. Also, in this "unscreening" process, we have to take into account that the DFPT calculation is done usually at a high temperature (e.g., $T^{\mathrm{high}} = 8000$ K) and the phonon self-energy is calculated at a lower temperature (e.g., $T^{\mathrm{low}} = 300$ K). Hence, we subtract the static self-energy at T^{high}, and then add the dynamical self-energy at T^{low}:

$$\Delta\Pi_{\mathbf{q}\nu}(\omega;T^{\mathrm{low}}) = \Pi^{\mathrm{1L}}_{\mathbf{q}\nu}(\omega;T^{\mathrm{low}}) - \Pi^{\mathrm{1L}}_{\mathbf{q}\nu}(0;T^{\mathrm{high}}) \tag{13}$$

More information about the phonon self-energy and the unscreening procedure can be found in the article J. Berges *et al.*, Phys. Rev. X **13**, 041009 (2023).

The phonon spectral function reads[9]

$$A_{\mathbf{q}}(\omega;T) = -\frac{2\omega}{\pi}\sum_{\nu}\mathrm{Im}\frac{1}{(\omega+i\eta)^2-\omega^2_{\mathbf{q}\nu}-2\omega_{\mathbf{q}\nu}\,\Delta\Pi_{\mathbf{q}\nu}(\omega;T)} \tag{14}$$

▶ Make a self-consistent calculation, a phonon calculation, and an EPW calculation to transform electron-phonon matrix elements to real space.

```
$ ibrun -n 8 $PATHQE/bin/pw.x -nk 4 -in scf.in > scf.out
$ ibrun -n 8 $PATHQE/bin/ph.x -nk 4 -in ph.in > ph.out
$ ibrun -n 8 $PATHQE/bin/pw.x -nk 4 -in nscf.in > nscf.out
$ python3 $PATHQE/EPW/bin/pp.py < pp.in
$ ibrun -n 8 $PATHQE/bin/epw.x -nk 8 epw1.in > epw1.out
```

scf.in

```
--
&control
   calculation  = 'scf'
   restart_mode = 'from_scratch'
   prefix       = 'mgb2'
   pseudo_dir   = './'
   outdir       = './'
/
&system
   ibrav        = 4
   celldm(1)    = 5.8260252227888
```

[7] M. Calandra *et al.*, Phys. Rev. B **82**, 165111 (2010), J. Berges *et al.*, Phys. Rev. X **13**, 041009 (2023)
[8] J. Berges *et al.*, Phys. Rev. B **101**, 155107 (2020), see Fig. 2(c).
[9] P. B. Allen and R. Silberglitt, Phys. Rev. B **9**, 4733 (1974). See also Eqs. (2,3) of J. Berges *et al.*, Phys. Rev. X **13**, 041009 (2023). We have an additional $2\omega_{\mathbf{q}\nu}$ factor multiplied to the self-energy because of the difference in the definition of the electron-phonon matrix element $g_{mn\nu}(\mathbf{k},\mathbf{q})$. Compare Eq. (2) of F. Giustino, Rev. Mod. Phys. **89**, 015003 (2017) and Eq. (12) of J. Berges *et al.*, Phys. Rev. X **13**, 041009 (2023).

```
   celldm(3)    = 1.1420694129095
   nat          = 3
   ntyp         = 2
   ecutwfc      = 40
   smearing     = 'fermi-dirac'
   occupations  = 'smearing
   degauss      = 0.02
/
&electrons
   diagonalization = 'david'
   mixing_beta     = 0.7
   conv_thr        = 1.0d-12
/
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 8 0 0 0
```

ph.in

```
--
&inputph
    prefix    = 'mgb2'
    fildyn    = 'mgb2.dyn.xml'
    fildvscf  = 'dvscf'
    tr2_ph    =  1.0d-20
    ldisp     = .true.
    reduce_io = .true.
    nmix_ph   = 12
    nq1 = 3
    nq2 = 3
    nq3 = 2
/
```

pp.in

```
--
mgb2
```

epw1.in

```
--
&inputepw
   prefix      = 'mgb2'
   outdir      = './'
   dvscf_dir   = './save'
   etf_mem     =  0

   elph        = .true.
   use_ws      = .true.
   epbwrite    = .true.
   epwwrite    = .true.

   wannierize  = .true.
   nbndsub     =  5

   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'
   bands_skipped = 'exclude_bands = 10-12'

   nk1         = 6
   nk2         = 6
   nk3         = 6

   nq1         = 3
```

```
   nq2         = 3
   nq3         = 2

   nkf1        = 1
   nkf2        = 1
   nkf3        = 1

   nqf1        = 1
   nqf2        = 1
   nqf3        = 1
/
```

The input files are similar to those used in Exercise 2, except for the following changes:

- For `scf.in`, we use `smearing = 'fermi-dirac'` instead of `smearing = 'mp'`. The reason is that EPW assumes Fermi–Dirac smearing in the phonon self-energy calculation.[10]
- To deal with the change in the smearing, we increased the $\mathbf{k}$ point grid to $12 \times 12 \times 8$ in `scf.in`.
- To make the phonon calculation run faster, we reduced `nq3` to 2 in `ph.in` and `epw1.in`. This parameter should be larger ($\sim$10) in a real calculation.
- For the `epw1.in` file, we remove input arguments related to WFPT.

▶ Perform an EPW restart calculation to obtain the electron self-energy and spectral function for $\mathbf{k}$ along the same high-symmetry path

```
$ ibrun -n 8 $PATHQE/bin/epw.x -nk 8 epw2.in > epw2.out
```

```
--                                                                                    epw2.in
&inputepw
  prefix      = 'mgb2'
  outdir      = './'
  dvscf_dir   = './save'
  etf_mem     =  1
  lopt_w2b    = .false.

  elph        = .true.
  use_ws      = .true.
  epwread     = .true.

  ! --- Input variables for phonon spectral function ---
  specfun_ph   = .true.

  filqf      = './qpt.txt'
  nkf1       = 10
  nkf2       = 10
  nkf3       = 10

  wmin_specfun = 0.00 ! eV
  wmax_specfun = 0.12 ! eV
  nw_specfun   = 200
  degaussw   = 0.05   ! eV
  temps      = 300.0  3157.75  ! Kelvin (0.02 Ry = 3157.75 K)
  ! ---------------------------------------------------

  nbndsub     =  5

  efermi_read  = .true.
  fermi_energy = 7.5014

  nk1         = 6
  nk2         = 6
  nk3         = 6
```

[10] The current version of EPW only supports Fermi–Dirac smearing in the phonon self-energy calculation. In the next release of EPW, we will support other smearing functions. We will also enable an option to automatize the "unscreening" of the phonon self-energy.

```
  nq1         = 3
  nq2         = 3
  nq3         = 2
/
```

Notes:

- We set `lopt_w2b = .false.` because the $\mathbf{q}$-point is not from a homogeneous grid, so the optimization does not work. We can then use the memory-optimized version with `etf_mem = 1`.
- `specfun_ph` asks for the frequency-dependent phonon self-energy and spectral function to be computed.
- `filqf` is the file containing the $\mathbf{q}$-points for the phonon spectral function calculation.
- `nkf1`, `nkf2`, and `nkf3` are the number of $\mathbf{k}$-points in the $\mathbf{k}$-point grid for the phonon spectral function calculation.
- The spectral function is computed for a uniform grid of frequencies between `wmin_specfun = 0.00` (eV) and `wmax_specfun = 0.12` (eV) with `nw_specfun = 200` points.
- `degaussw = 0.05` (eV) is the smearing parameter η in the self-energy formula Eq. (12).
- We compute the self-energy at two temperatures `temps = 300.0, 3157.75` (K) to account for the unscreening procedure. 3157.75 K corresponds to 0.02 Ry smearing value used in the SCF calculation.

In addition to the standard output, the code will produce additional output files. One can obtain a complete information on the frequency-dependent self-energy from the file `specfun_sup.phon` (again, we trimmed some digits):

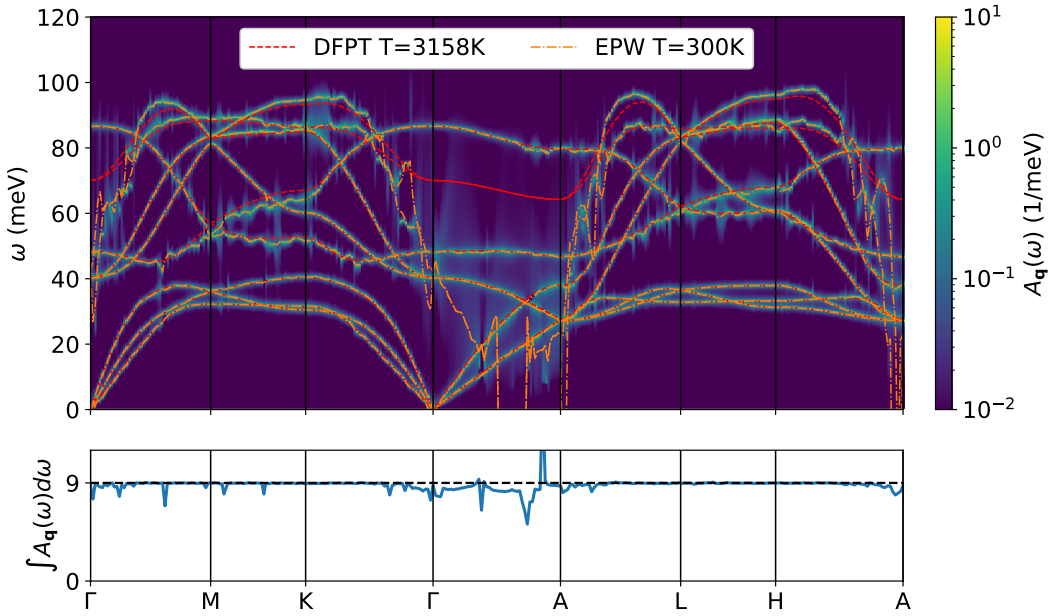
```
#Phonon eigenenergies + real and im part of phonon self-energy (meV)
#Q-point    Mode      Temp.[K]       w_q[eV]        w[eV]    Real Sigma(w)[meV]
   Real Sigma(w=0)[meV]     Im Sigma(w)[meV]
      1        1   300.000      -0.00000       0.00000    0.00000E+00    0.00000E+00    0.00000E+00
      1        2   300.000      -0.00000       0.00000    0.00000E+00    0.00000E+00    0.00000E+00
      ...
```

The static self-energy $\Pi^{1\mathrm{L}}_{\mathbf{q}\nu}(0;T^{\mathrm{high}})$ in Eq. (13) can be found in the 7th column, and the dynamical self-energy $\Pi^{1\mathrm{L}}_{\mathbf{q}\nu}(\omega;T^{\mathrm{low}})$ is in the 6th (real part) and 8th (imaginary part) columns, for the corresponding temperatures in the 3rd row.

The output can be parsed and visualized using the `plot.py` script. The script implements Eq. (14) for the calculation of the self-energy. As in the electronic case, we linearly interpolate the self-energy in frequency. (See Appendix 2 for the full script.) (If you get `ModuleNotFoundError`, make sure you have loaded the python module and install the matplotlib package (see page 15).)

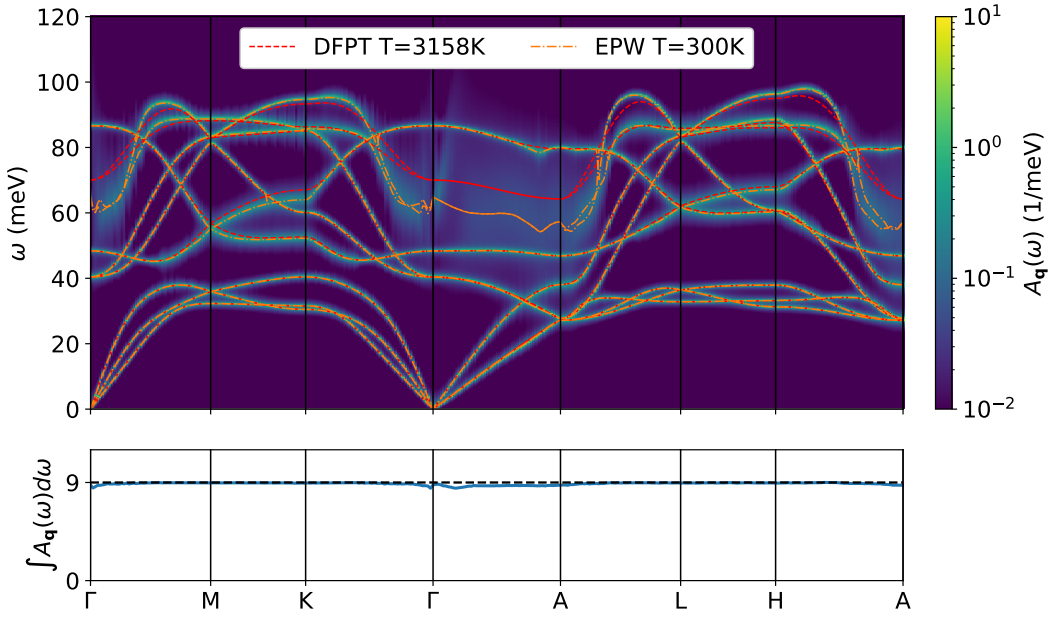
```
$ python3 plot.py
```

The result looks as follows:



The red dashed curves show the high-temperature phonon dispersion computed using DFPT, the orange dash-dotted curves the low-temperature static phonon dispersion, and the background color the low-temperature spectral function. One finds that the phonon spectral function is broadened and shifted due to the electron-phonon interaction. The integral of the spectral function should give the number of phonon modes in the system. Here, the frequency and $\mathbf{k}$-point mesh is too coarse, so the integral is not exactly 9 but oscillates around 9.

Using a denser $60 \times 60 \times 60$ $\mathbf{k}$-point grid and 500 frequency points, we have:



▶ Try to increase the number of frequency grid points and the number of $\mathbf{k}$ points to converge the spectral function and the integral.

▶ Try to change the temperature and see the temperature dependence of the spectral function.

Appendix 1: Plotting script for the electron self-energy

```
#!/usr/bin/env python3
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.image import NonUniformImage

plt.rcParams.update({'font.size': 15})

def parse_epw_selfen(filename, nk, nbnd):
    xks = np.zeros((3, nk))
    energy = np.zeros((nbnd, nk))
    sigma = np.zeros((nbnd, nk), dtype=complex)

    with open(filename, 'r') as f:
        for line in f:
            if "Electron Self-Energy using Wannier function perturbation theory" in line:
                break
        print(f.readline())
        f.readline()
        f.readline()
        for ik in range(nk):
            xks[:, ik] = [float(x) for x in f.readline().split()[-3:]]
            f.readline()
            for ib in range(nbnd):
                data = f.readline().split()
                energy[ib, ik] = float(data[3])
                sigma[ib, ik] = float(data[6]) + 1j * float(data[9])
            f.readline()
            f.readline()
            f.readline()

    sigma /= 1000 # meV to eV
    return xks, energy, sigma


def parse_epw_specfun_sup(filename):
    data = np.loadtxt(filename)
    nk = int(data[-1, 0])
    nbnd = int(data[-1, 1])

    data = data.reshape(nbnd, nk, -1, 6)
    ws = data[0, 0, :, 3]
    es = data[:, :, 0, 2]
    sigma = (data[:, :, :, 4] + 1j * data[:, :, :, 5]) / 1000 # meV to eV
    return ws, es, sigma

def parse_epw_specfun(filename):
    data = np.loadtxt(filename)
    nk = int(data[-1, 0])

    data = data.reshape(nk, -1, 3)
    ws = data[0, :, 1]
    As = data[:, :, 2] * 1000  # 1/meV to 1/eV

    n_int = []
    with open(filename, "r") as f:
        for line in f:
            if "Integrated spectral function" in line:
                n_int += [float(line.split()[-1])]
    n_int = np.array(n_int)

    return ws, As, n_int

# -------------------------------------
# Parse EPW output
use_interpolation = True
T = 300.0

ws, es, sigma = parse_epw_specfun_sup(f"specfun_sup.elself.{T:.3f}K")
```

```
ws, As, nocc = parse_epw_specfun(f"specfun.elself.{T:.3f}K")
nbnd, nk, nfreq = sigma.shape

xks, _, sigma_ahc = parse_epw_selfen("epw2.out", nk, nbnd)


# -------------------------------------
# Compute spectral function on a denser frequency grid using linear interpolation of the self-energy
# A = 1 / (w - e - sigma)
ws_itp = np.linspace(ws.min(), ws.max(), 10_000, True)
dw = ws_itp[1] - ws_itp[0]
fermi_dirac = 1 / (np.exp(ws_itp / (T * 8.61732814974056E-05)) + 1)

nocc_itp = np.zeros((nk,))
As_itp = np.zeros((nk, len(ws_itp)))
for ik in range(nk):
    for ib in range(nbnd):
        sigma_itp = np.interp(ws_itp, ws, sigma[ib, ik, :])
        As_itp[ik, :] += (1 / (ws_itp - es[ib, ik] - sigma_itp)).imag / np.pi
    nocc_itp[ik] += np.sum(As_itp[ik, :] * fermi_dirac) * dw

# -------------------------------------
if use_interpolation:
    # With interpolation
    As_plot = As_itp
    ws_plot = ws_itp
    nocc_plot = nocc_itp

else:
    # Without interpolation
    As_plot = As
    ws_plot = ws
    nocc_plot = nocc


# Set x axis
dks = np.linalg.norm(xks[:, 1:] - xks[:, :-1], axis=0)
xs = np.concatenate(([0.], np.cumsum(dks)))

fig, axes = plt.subplots(2, 2, figsize=(10, 6), sharex="col",
    gridspec_kw={'height_ratios': [3, 1], "width_ratios":[1,0.02]})

axes[1, 1].set_axis_off()

# -------------------------------------
# Plot spectral function
plt.sca(axes[0, 0])
im = NonUniformImage(axes[0, 0], interpolation='nearest',
    extent=[xs[0], xs[-1], ws_plot.min(), ws_plot.max()],
    cmap="viridis", norm=plt.matplotlib.colors.LogNorm(vmin=0.01, vmax=10))
im.set_data(xs, ws_plot, As_plot.T)
axes[0, 0].add_image(im)
cbar = plt.colorbar(im, cax=axes[0, 1])
cbar.set_label("$A_{\mathbf{k}}(\omega)$ (1/eV)")

# Plot bands
for ibnd in range(nbnd):
    plt.plot(xs, es[ibnd, :], "k-", lw=1)
    plt.plot(xs, es[ibnd, :] + sigma_ahc[ibnd, :].real, "r-", lw=1)

plt.axhline(0, c="grey", lw=1)
plt.ylabel("$\omega - E_\mathrm{Fermi}$ (eV)")
plt.ylim([ws_plot.min(), ws_plot.max()])

# -------------------------------------
# Plot integrated spectral function
plt.sca(axes[1, 0])
dw = ws_plot[1] - ws_plot[0]
# plt.plot(xs, np.sum(As_plot, axis=1) * dw, label=r"$A(\omega)$")
plt.plot(xs, nocc_plot, "-", label=r"$A(\omega) f_{\rm FD}(\omega)$", lw=2)
plt.ylabel(r"$\int A(\omega) f_{\rm FD}(\omega) d\omega$")
```

```
# plt.legend()

plt.ylim([0, 6])
for i in range(6):
    plt.axhline(i, c="k", lw=0.5, ls="--")

xs_highsym = xs[np.arange(0, 351, 50)]
for x in xs_highsym:
    for ax in axes[:, 0]:
        ax.axvline(x, c="k", lw=1)
plt.xticks(xs_highsym, ["$\Gamma$", "M", "K", "$\Gamma$", "A", "L", "H", "A"])
plt.xlim([xs[0], xs[-1]])


plt.tight_layout()
fig.savefig("mgb2_spectral.pdf")

axes[0, 0].set_ylim([-1.0, 1.5])
fig.savefig("mgb2_spectral_zoom.pdf")
#plt.show()
```

Appendix 2: Plotting script for the phonon self-energy

```
#!/usr/bin/env python3
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.image import NonUniformImage

plt.rcParams.update({'font.size': 15})

filename = "specfun_sup.phon"

# ------------------------------------
# Parse specfun_sup file
data = np.loadtxt(filename)
nq = int(data[-1, 0])
nmodes = int(data[-1, 1])

# [iq, iT, iw, imode, idata]
data = data.reshape(nq, 2, -1, nmodes, 8)
ws = data[0, 0, :, 0, 4] * 1e3  # eV to meV
nw = len(ws)

T_low = data[0, 0, 0, 0, 2]
T_high = data[0, 1, 0, 0, 2]
w_ph = data[:, 0, 0, :, 3] * 1e3  # eV to meV
Pi_low = data[:, 0, :, :, 5] + 1j * data[:, 0, :, :, 7]
Pi_high_0 = data[:, 1, :, :, 6]
Pi = Pi_low - Pi_high_0

xks = np.loadtxt("qpt.txt", skiprows=1)[:, :3].T
dks = np.linalg.norm(xks[:, 1:] - xks[:, :-1], axis=0)
xs = np.concatenate(([0.], np.cumsum(dks)))

ws_plot = np.linspace(ws.min(), ws.max(), 1_000, True)

# ------------------------------------
# Compute self-energy
A = np.zeros((nq, len(ws_plot)))
w_ph_low = np.zeros_like(w_ph)
eta = 0.1 # meV

for iq in range(nq):
   for imode in range(nmodes):
      wq = w_ph[iq, imode]

      w2 = wq**2 + 2 * wq * Pi[iq, 0, imode]
      w_ph_low[iq, imode] = np.sqrt(abs(w2).real) * np.sign(w2.real)

      Pi_itp = np.interp(ws_plot, ws, Pi[iq, :, imode])

      for iw in range(len(ws_plot)):
         w = ws_plot[iw]
         Pi_w = Pi_itp[iw]
         A[iq, iw] += -np.imag(2 * w / ((w + 1j * eta)**2 - wq**2 - 2 * wq * Pi_w)) / np.pi

A[A < 0] = 1e-10

fig, axes = plt.subplots(2, 2, figsize=(10, 6), sharex="col",
   gridspec_kw={'height_ratios': [3, 1], "width_ratios":[1,0.02]})
axes[1, 1].set_axis_off()

# ------------------------------------
# Plot spectral function
plt.sca(axes[0, 0])
im = NonUniformImage(axes[0, 0], interpolation='nearest',
   extent=[xs[0], xs[-1], ws_plot.min(), ws_plot.max()], cmap="viridis",
   norm=plt.matplotlib.colors.LogNorm(vmin=0.01, vmax=10))
im.set_data(xs, ws_plot, A.T)
axes[0, 0].add_image(im)
cbar = plt.colorbar(im, cax=axes[0, 1])
```

```
cbar.set_label("$A_{\mathbf{q}}(\omega)$ (1/meV)")

# Plot bands
for i in range(nmodes):
   plt.plot(xs, w_ph[:, i], "r--", lw=1, label=f"DFPT T={T_high:.0f}K" if i == 0 else None)
   plt.plot(xs, w_ph_low[:, i], "-.", c="C1", lw=1, label=f"EPW T={T_low:.0f}K" if i == 0 else None)
plt.legend(framealpha=1.0, loc="upper center", ncol=2)

plt.axhline(0, c="grey", lw=1)
plt.ylabel("$\omega$ (meV)")
plt.ylim([ws_plot.min(), ws_plot.max()])

# -------------------------------------
# Plot integrated spectral function
plt.sca(axes[1, 0])
dw = ws_plot[1] - ws_plot[0]
plt.plot(xs, np.sum(A, axis=1) * dw, "-", label=r"$A(\omega)$", lw=2)
plt.ylabel(r"$\int A_{\mathbf{q}}(\omega) d\omega$")

plt.ylim([0, 12])
plt.yticks([0, 9])
plt.axhline(9, c="k", ls="--")

xs_highsym = xs[np.arange(0, 351, 50)]
for x in xs_highsym:
      for ax in axes[:, 0]:
         ax.axvline(x, c="k", lw=1)
plt.xticks(xs_highsym, ["$\Gamma$", "M", "K", "$\Gamma$", "A", "L", "H", "A"])
plt.xlim([xs[0], xs[-1]])


plt.tight_layout()
fig.savefig("mgb2_phonon.pdf")
#plt.show()
```