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

Calculations of superconducting Tc

Hands-on session (Wed.9)

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

Exercise 1

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

```
$ cd $SCRATCH
$ mkdir EP-SCHOOL && cd EP-SCHOOL
$ cp /work2/05193/sabyadk/stampede3/EPWSchool2024/tutorials/Wed.9.Mori.tar .
$ tar -xvf Wed.9.Mori.tar && cd Wed.9.Mori/exercise1
```

▶ 1st step: Run a self-consistent calculation on a homogeneous 12x12x12 $\mathbf{k}$-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`):

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

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

job.ph

```
#!/bin/bash
#SBATCH -J job.ph             # Job name
#SBATCH -N 1                  # Total # of nodes
#SBATCH --ntasks-per-node 8
#SBATCH -t 00:30:00           # Run time (hh:mm:ss)
#SBATCH -A DMR23030
#SBATCH -p skx
#SBATCH --reservation=NSF_Summer_School_Wed

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

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

scf.in

```
&control
   calculation  = 'scf'
   restart_mode = 'from_scratch',
   prefix       = 'pb',
   pseudo_dir   = '../../pseudo/',
   outdir       = './'
/
&system
   ibrav        = 2,
   celldm(1)    = 9.27,
```

```
   nat          = 1,
   ntyp         = 1,
   ecutwfc      = 30.0
   occupations  = 'smearing',
   smearing     = 'mp',
   degauss      = 0.025
/
&electrons
   diagonalization = 'david'
   mixing_beta  = 0.7
   conv_thr     = 1.0d-12
/
ATOMIC_SPECIES
 Pb 207.2 pb_s.UPF
ATOMIC_POSITIONS
 Pb 0.00 0.00 0.00
K_POINTS {automatic}
 12 12 12 0 0 0
```

ph.in

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

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

```
 Dynamical matrices for ( 3, 3, 3)  uniform grid of q-points
 (   4 q-points):
   N         xq(1)         xq(2)         xq(3)
   1   0.000000000   0.000000000   0.000000000
   2  -0.333333333   0.333333333  -0.333333333
   3   0.000000000   0.666666667   0.000000000
   4   0.666666667  -0.000000000   0.666666667
```

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

```
$ python3 /work2/05193/sabyadk/stampede3/EPWSchool2024/q-e/EPW/bin/pp.py
```

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

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

```
$ cd ../epw1-2
$ sbatch job.epw1
```

job.epw1

```
#!/bin/bash
#SBATCH -J job.epw1           # Job name
#SBATCH -N 1                  # Total # of nodes
#SBATCH --ntasks-per-node 8
#SBATCH -t 01:00:00           # Run time (hh:mm:ss)
#SBATCH -A DMR23030
```

```
#SBATCH -p skx
#SBATCH --reservation=NSF_Summer_School_Wed

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

ibrun $PATHQE/bin/pw.x -nk 8 -in scf.in > scf.out
#alternatively to re-run a scf calculation copy files from ../phonon/pb.save
#mkdir pb.save
#cp ../phonon/pb.save/charge-density.dat pb.save/
#cp ../phonon/pb.save/data-file-schema.xml pb.save/

ibrun $PATHQE/bin/pw.x -nk 8 -in nscf.in > nscf.out
ibrun $PATHQE/bin/epw.x -nk 8 -in epw1.in > epw1.out
```

nscf.in

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

epw1.in

```
--
&inputepw
  prefix      = 'pb',
  outdir      = './'
  dvscf_dir   = '../phonon/save'  ! directory where .dyn, .dvscf and prefix.phsave/patterns.xx.yy
                                  ! files obtained from phonon calculation are stored

  ep_coupling = .true.          ! run e-ph coupling calculation
  elph        = .true.          ! calculate e-ph coefficients
  epwwrite    = .true.          ! write e-ph matrices in Wann representation
  epwread     = .false.         ! read e-ph matrices from 'prefix.epmatwp' file
  etf_mem     =  0              ! less I/O but more memory is required

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

  num_iter    = 300
  dis_froz_min= -3
```

```
  dis_froz_max= 13.5
  proj(1)   = 'Pb:sp3'
  wdata(1)  = 'bands_plot = .true.'
  wdata(2)  = 'begin kpoint_path'
  wdata(3)  = 'G 0.00 0.00 0.00 X 0.00 0.50 0.50'
  wdata(4)  = 'X 0.00 0.50 0.50 W 0.25 0.50 0.75'
  wdata(5)  = 'W 0.25 0.50 0.75 L 0.50 0.50 0.50'
  wdata(6)  = 'L 0.50 0.50 0.50 K 0.375 0.375 0.75'
  wdata(7)  = 'K 0.375 0.375 0.75 G 0.00 0.00 0.00'
  wdata(8)  = 'G 0.00 0.00 0.00 L 0.50 0.50 0.50'
  wdata(9)  = 'end kpoint_path'
  wdata(10) = 'bands_plot_format = gnuplot'

  fsthick     = 0.4             ! Fermi window thickness [eV]
  degaussw    = 0.1             ! smearing in energy-conserving delta functions in [eV]
  degaussq    = 0.5             ! smearing for sum over q in the e-ph coupling in [meV]

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

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

  nsiter  = 500                 ! number of self-consistent iterations when solving ME eqs.
  npade   = 22                  ! percentage of Matsubara points used in Pade continuation.
  conv_thr_iaxis = 1.0d-3       ! convergence threshold for solving ME eqs. on imaginary axis
  conv_thr_racon = 1.0d-3       ! convergence threshold for solving ME eqs. on real axis

  wscut   = 0.1                 ! upper limit over Matsubara freq. summation in ME eqs on imag.axis in [eV]
  muc     = 0.1                 ! effective Coulomb potential used in the ME eqs.
  temps   = 0.3 0.9 1.5 2.1 2.7 3.3 4.0 4.2 4.4 4.5 4.6   ! temperatures at which ME eqs.
            ! are solved [equally spaced temperature points can also be used: see epw2.in]

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

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

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

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

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

```
$ /work2/05193/sabyadk/stampede3/EPWSchool2024/q-e/external/wannier90/utility/kmesh.pl 3 3 3
```

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

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

and uncomment the following three lines

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

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

Note 4: In some cases, `pw.x` calculates additional $\mathbf{k}$-points which are not provided in the $\mathbf{k}$-point list of the input. If this happens, you need to use the keyword of `calculation='bands'` instead of `calculation='nscf'`.

With the above input, we are instructing EPW to:

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

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

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

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

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

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

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

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

Progression iq (fine) =        100/      5579
Progression iq (fine) =        200/      5579
......
Progression iq (fine) =       5500/      5579
             Fermi level (eV) =    0.117577170439420D+02
DOS(states/spin/eV/Unit Cell) =    0.296296405432580D+00
       Electron smearing (eV) =    0.100000000000000D+00
            Fermi window (eV) =    0.400000000000000D+00

Finish writing .ephmat files
```

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

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

  Note 2: Because the electron-phonon matrix elements do not depend on the temperature at which the Migdal-Eliashberg equations are solved, they can be reused in subsequent EPW calculations at different temperatures.

This is the reason why ephmatXX files are saved in the pb.ephmat directory.

The isotropic Migdal-Eliashberg equations take the following form:

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

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

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

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

```
=============================================================
Solve isotropic Eliashberg equations
=============================================================

Finish reading freq file
......

      1 bands within the Fermi window

Finish reading egnv file

Max nr of q-points =       956

Finish reading ikmap files

Start reading .ephmat files

Finish reading .ephmat files

a2f file is not found to estimate initial gap: calculating a2f files

Finish reading a2f file

Electron-phonon coupling strength =    1.4242792

Estimated Allen-Dynes Tc =     3.168499 K for muc =    0.10000

Estimated w_log in Allen-Dynes Tc =     2.519536 meV

Estimated BCS superconducting gap =     0.480551 meV

Estimated Tc from machine learning model =     4.159446 K

WARNING WARNING WARNING

The code may crash since tempsmax =    4.600 K is larger than Allen-Dynes Tc =     3.168 K
Actual number of frequency points (     1) =    616 for uniform       sampling

temp(  1) =      0.30000 K
```

```
Solve isotropic Eliashberg equations on imaginary-axis

Total number of frequency points nsiw(     1) =    616
Cutoff frequency wscut =     0.1000 eV
Maximum frequency =         0.1000 eV
broyden mixing factor =      0.70000

   iter      ethr          znormi      deltai [meV]
     1   3.130497E+00   2.278915E+00   6.137465E-01
     2   1.477095E-01   2.261020E+00   6.728962E-01
     ......
     8   4.102383E-04   2.209124E+00   8.364763E-01
Convergence was reached in nsiter =      8
```

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

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

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

   pade    Re[znorm]   Re[delta] [meV]
   136   2.209523E+00   8.366739E-01
Convergence was reached for N =    136 Pade approximants

raxis_pade   :      0.01s CPU      0.06s WALL (       1 calls)

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

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

   iter      ethr      Re[znorm]      Re[delta] [meV]
     1   5.087807E-02   2.209517E+00   8.364269E-01
     2   2.288703E-02   2.209517E+00   8.364269E-01
     ......
     8   3.793260E-04   2.209517E+00   8.364268E-01
Convergence was reached in nsiter =      8
```

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

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

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

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

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

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

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

  pb.imag_iso_XX files are generated by setting `eliashberg = .true.`, `liso = .true.`, and `limag = .true.`. Each file contains 3 columns: the Matsubara frequency $i\omega_j$ (eV) along the imaginary axis, the quasiparticle renormalization function $Z(i\omega_j)$, and the superconducting gap $\Delta(i\omega_j)$ (eV).

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

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

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

You can use the gnuplot script fig1.plt to plot pb.imag_iso_000.30, pb.pade_iso_000.30, and pb.acon_iso_000.30. You should get something similar to Fig. 1 at 0.3 K.

```
$ gnuplot fig1.plt
$ evince fig1.pdf
```

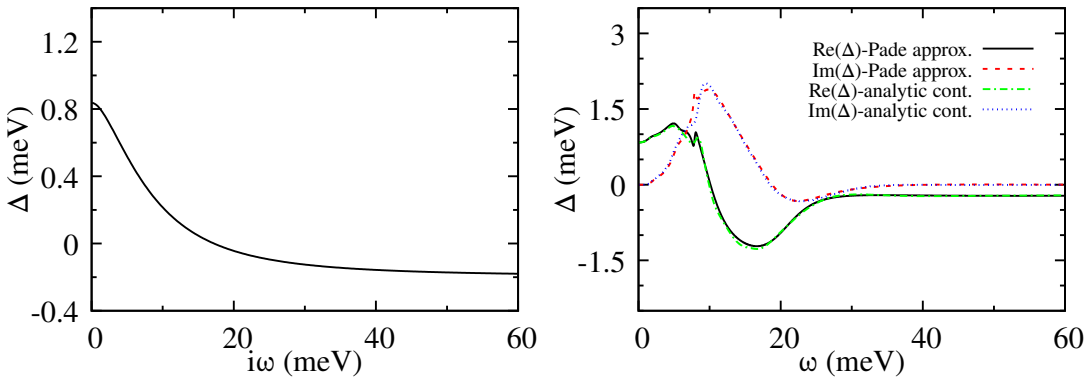

Fig. 1 Left: Superconducting gap along the imaginary axis (columns 1:3 from pb.imag_iso_000.30). Right: Superconducting gap on the real axis (columns 1:4 and 1:5 from pb.pade_iso_000.30 and pb.acon_iso_000.30).

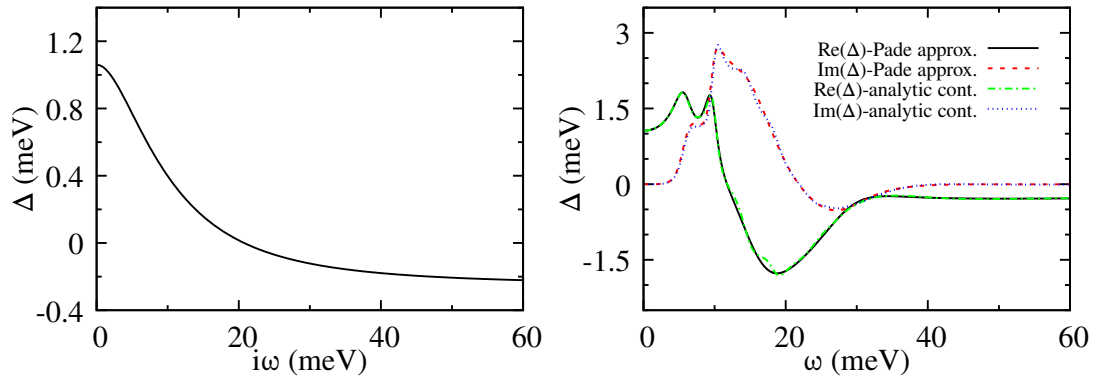

Fig. 1(R) At convergence you should get something close to this figure (see Phys. Rev. B **87**, 024505 (2013) for fully converged calculation parameters).

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

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

script_gap0_imag

```
#!/bin/tcsh

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

```
$ ./script_gap0_imag
```

You can use the `gnuplot` script `fig3.plt` to plot `pb.imag_iso_gap0`. You should get something similar to Fig. 2.

```
$ gnuplot fig2.plt
$ evince fig2.pdf
```

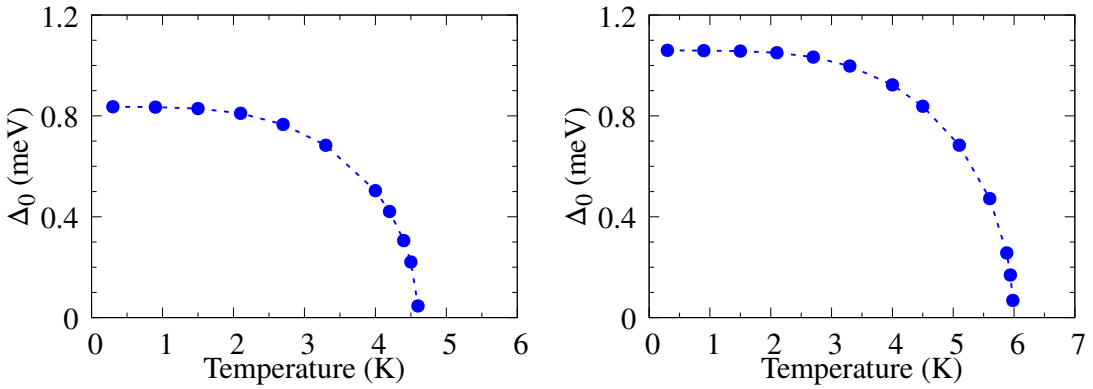

Fig. 2 Calculated isotropic gap of Pb as a function of temperature. At convergence you should get something close to the figure on the right (see Phys. Rev. B **87**, 024505 (2013) for fully converged calculation parameters).

You can further extract the leading edge of the superconducting gap as a function of temperature from the calculations on the real axis and compare it with the one obtained on the imaginary axis shown in Fig. 2. You can use the shell scripts `script_gap0_pade` and `script_gap0_acon` to get the `pb.pade_iso_gap0` and `pb.acon_iso_gap0` files. Next plot these files using gnuplot as was done above for the `pb.imag_iso_gap0`.

```
$ ./script_gap0_pade
```

script_gap0_pade

```
#!/bin/tcsh

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

and

```
$ ./script_gap0_acon
```

script_gap0_acon

```
#!/bin/tcsh

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

You can compare the leading edges on the real axis with the one obtained on the imaginary axis. You should get something similar to Fig. 3.

```
$ gnuplot fig3.plt
$ evince fig3.pdf
```

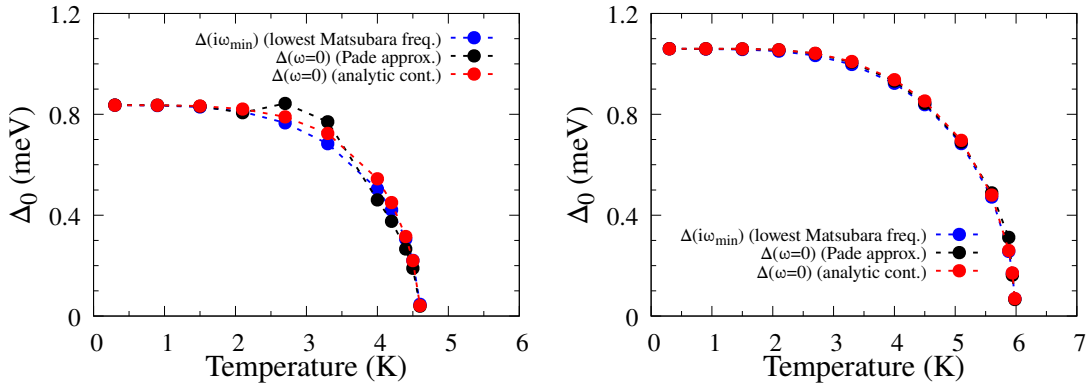

Fig. 3 Calculated isotropic gap of Pb as a function of temperature. At convergence you should get something close to the figure on the right (see Phys. Rev. B **87**, 024505 (2013) for fully converged calculation parameters).

▶ 6th step: Solve the linearized isotropic Migdal-Eliashberg equation for the critical temperature.

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

$$\Delta(i\omega_j) = \sum_{j'} K_{jj'} \Delta(i\omega_{j'}) \tag{3}$$

where

$$K_{jj'} = \frac{1}{|2j'+1|}\left[\lambda(\omega_j - \omega_{j'}) - \mu_{\rm c}^* - \delta_{jj'}\sum_{j''}\lambda(\omega_j - \omega_{j''})s_j s_{j''}\right] \quad (4)$$

and $s_j = sign(\omega_j)$. The critical temperature $T_{\rm c}$ is defined as the value at which the maximum eigenvalue of $K_{jj'}$ is 1.

This step can be done by starting from a file containing the Eliashberg spectral function (`pb.a2f_iso`) using the following jobscript (`job.epw2`) and input file (`epw2.in`; only differences with respect to `epw1.in` file are shown below):

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

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

```
$ sbatch job.epw2
```

job.epw2

```
#!/bin/bash
#SBATCH -J job.epw2           # Job name
#SBATCH -N 1                  # Total # of nodes
#SBATCH --ntasks-per-node 1
#SBATCH -t 00:30:00           # Run time (hh:mm:ss)
#SBATCH -A DMR23030
#SBATCH -p skx
#SBATCH --reservation=NSF_Summer_School_Wed

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

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

epw2.in

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

 lpade       = .false.
 lacon       = .false.
 fila2f      = 'pb.a2f'        ! read the a2f file
 tc_linear   = .true.          ! solve linearized ME eqn. for Tc
 tc_linear_solver  = 'power'   ! algorithm to solve Tc eigenvalue problem: 'power' OR 'lapack'

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

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

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

 Superconducting transition temp. Tc is the one which has Max. eigenvalue close to 1
```

```
-------------------------------------------------------------
       Temp.       Max.          nsiw      wscut     Nr. of iters
       (K)       eigenvalue     (itemp)   (eV)       to Converge
-------------------------------------------------------------
       0.25      4.2835755       739      0.1001        6
       0.50      3.4413117       369      0.1000        7
       ......
       5.25      0.8811649        35      0.1009       27
-------------------------------------------------------------
 Finish: Solving (isotropic) linearized Eliashberg equation
```

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

```
$ ./script_max_eigenvalue
```

`script_max_eigenvalue`

```
#!/bin/tcsh

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

Plot `data_max_eigenvalue.dat` to obtain the T_c. The critical temperature is defined as the value for which the maximum eigenvalue is close to 1. You can use the `gnuplot` script `fig4.plt` to get the graph shown in Fig. 4.

```
$ gnuplot fig4.plt
$ evince fig4.pdf
```

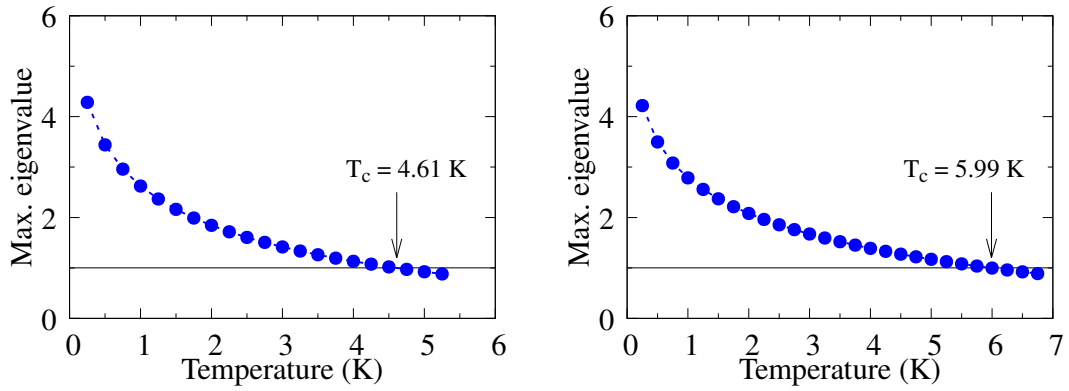

Fig. 4 Calculated maximum eigenvalue as a function of temperature. At convergence you should get something closer to the right hand-side figure.

▶ 7th step: (**Optional due to time limit**) Compute the nesting function $f_{\rm nest}(\mathbf{q})$ as well as the electron-phonon coupling strength of each phonon $\lambda_{\mathbf{q}\nu}$ along the $\mathbf{q}$-point path. Since the electron-phonon interaction in Wannier representation has already been calculated with `epw1.in`, some files can be reused. Create symbolic links for several files, and then do an EPW calculation along the $\mathbf{q}$-point path specified by `filqf` using the following jobscript (`job.epw`) and input file (`epw3.in` and `epw4.in` files are shown below; Only the differences from epw3.in are shown for epw4.in).

```
$ cd ../epw3-4
$ ln -s ../epw1-2/pb.epmatwp .
```

```
$ ln -s ../epw1-2/wigner.fmt .
$ ln -s ../epw1-2/vmedata.fmt .
$ ln -s ../epw1-2/epwdata.fmt .
$ ln -s ../epw1-2/crystal.fmt .
$ ln -s ../epw1-2/pb.ukk .
$ sbatch job.epw
```

job.epw

```
#!/bin/bash
#SBATCH -J job.epw            # Job name
#SBATCH -N 1                  # Total # of nodes
#SBATCH --ntasks-per-node 8
#SBATCH -t 00:30:00           # Run time (hh:mm:ss)
#SBATCH -A DMR23030
#SBATCH -p skx
#SBATCH --reservation=NSF_Summer_School_Wed

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

ibrun $PATHQE/bin/epw.x -nk 8 -in epw3.in > epw3.out
ibrun $PATHQE/bin/epw.x -nk 8 -in epw4.in > epw4.out
```

epw3.in

```
--
&inputepw
  prefix      = 'pb',
  outdir      = './'
  dvscf_dir   = '../phonon/save'

  ep_coupling = .true.          ! run e-ph coupling calculation
  elph        = .true.          ! calculate e-ph coefficients
  epwwrite    = .false.         ! write e-ph matrices in Wann representation
  epwread     = .true.          ! read e-ph matrices from 'prefix.epmatwp' file
  etf_mem     =  0              ! less I/O but more memory is required

  wannierize  = .false.         ! if false, skip the Wannierization
  nbndsub     =  4              ! number of Wannier functions to utilize

  fsthick     = 0.4             ! Fermi window thickness [eV]
  degaussw    = 0.1             ! smearing in energy-conserving delta functions in [eV]
  degaussq    = 0.5             ! smearing for sum over q in the e-ph coupling in [meV]

  phonselfen = .true.           ! calculate the phonon self-energy
  delta_approx= .true.          ! apply the double delta approximation
                                ! for phonon self energy.

  nk1       = 3
  nk2       = 3
  nk3       = 3

  nq1       = 3
  nq2       = 3
  nq3       = 3

  filqf     = 'pb_band.kpt'     ! q points provided in the file
  nkf1      = 18
  nkf2      = 18
  nkf3      = 18
 /
```

epw4.in

```
--
  nest_fn = .true.              ! calculate the nesting function
```

Note: If any of elecselfen, phonselfen, specfun_el, or specfun_ph is true, mp_mesh_k must be false. The default value is false.

Within the double delta approximation (`delta_approx = .true.`), the electron-phonon coupling strength of each phonon $\lambda_{\mathbf{q}\nu}$ is defined as follows:

$$\lambda_{\mathbf{q}\nu} = \frac{2}{N_{\mathrm{F}}} \sum_{nm} \int \frac{d\mathbf{k}}{\Omega_{\mathrm{BZ}}} \frac{|g_{mn\nu}(\mathbf{k},\mathbf{q})|^2}{\omega_{\mathbf{q}\nu}} \delta(\epsilon_{n\mathbf{k}} - \epsilon_{\mathrm{F}}) \delta(\epsilon_{m\mathbf{k}+\mathbf{q}} - \epsilon_{\mathrm{F}}) \tag{5}$$

You should get the following outputs in `epw3.out`:

```
 ================================================================
 Phonon (Imaginary) Self-Energy in the Migdal Approximation
 ================================================================

 Fermi Surface thickness =   0.400000 eV

 Golden Rule strictly enforced with T =   0.025852 eV
 Gaussian Broadening:   0.100000 eV, ngauss=   1
 DOS =  0.296296 states/spin/eV/Unit Cell at Ef= 11.757717 eV

 ismear =     1 iq =       1 coord.:   0.00000  0.00000  0.00000 wt:   1.00000 Temp:  300.000K
 ----------------------------------------------------------------
 lambda___(   1 )=       0.000000   gamma___=       0.000000 meV   omega=      0.0000 meV
 lambda_tr(   1 )=       0.000000   gamma_tr=       0.000000 meV   omega=      0.0000 meV
 lambda___(   2 )=       0.000000   gamma___=       0.000000 meV   omega=      0.0000 meV
 lambda_tr(   2 )=       0.000000   gamma_tr=       0.000000 meV   omega=      0.0000 meV
 lambda___(   3 )=       0.000000   gamma___=       0.000000 meV   omega=      0.0000 meV
 lambda_tr(   3 )=       0.000000   gamma_tr=       0.000000 meV   omega=      0.0000 meV
 lambda___( tot )=       0.000000
 lambda_tr( tot )=       0.000000
 ----------------------------------------------------------------

 Number of (k,k+q) pairs on the Fermi surface:      956 out of     5832
 ......
 ......
 ......
 ismear =     1 iq =      43 coord.:   0.37500  0.37500  0.75000 wt:   1.00000 Temp:  300.000K
 ----------------------------------------------------------------
 lambda___(   1 )=       0.245003   gamma___=       0.006201 meV   omega=      5.2143 meV
 lambda_tr(   1 )=       0.215219   gamma_tr=       0.005447 meV   omega=      5.2143 meV
 lambda___(   2 )=       0.223415   gamma___=       0.008930 meV   omega=      6.5530 meV
 lambda_tr(   2 )=       0.111656   gamma_tr=       0.004463 meV   omega=      6.5530 meV
 lambda___(   3 )=       0.265418   gamma___=       0.016608 meV   omega=      8.1988 meV
 lambda_tr(   3 )=       0.241549   gamma_tr=       0.015114 meV   omega=      8.1988 meV
 lambda___( tot )=       0.733836
 lambda_tr( tot )=       0.568425
 ----------------------------------------------------------------

 Number of (k,k+q) pairs on the Fermi surface:      136 out of     5832
```

The default temperature is 300K, so $\lambda_{\mathbf{q}\nu}$ is output for 300K. However, as long as the double delta approximation is applied, $\lambda_{\mathbf{q}\nu}$ does not depend on temperature. $\lambda_{\mathbf{q}\nu}$ is output in `lambda.phself.300.000K`. Similarly, the nesting function $f_{\mathrm{nest}}(\mathbf{q})$ is defined as follows:

$$f_{\mathrm{nest}}(\mathbf{q}) = \sum_{nm} \int \frac{d\mathbf{k}}{\Omega_{\mathrm{BZ}}} \delta(\epsilon_{n\mathbf{k}} - \epsilon_{\mathrm{F}}) \delta(\epsilon_{m\mathbf{k}+\mathbf{q}} - \epsilon_{\mathrm{F}}) \tag{6}$$

You should get the nesting function in `epw4.out` for each $\mathbf{q}$ point.

```
 ================================================================
 Nesting Function in the double delta approx
 ================================================================

 Fermi Surface thickness =   0.400000 eV
 Gaussian Broadening:   0.100 eV, ngauss=   1
 DOS =  0.296296 states/spin/eV/Unit Cell at Ef= 11.757717 eV

 iq =     1 coord.:   0.00000  0.00000  0.00000 wt:   1.00000
```

```
-----------------------------------------------------------------
Nesting function (q)=   0.417115E+03 [Adimensional]
-----------------------------------------------------------------

Number of (k,k+q) pairs on the Fermi surface:      956 out of     5832
......
```

Extract the data from epw4.out by using the shell script `script_nesting_fn`.

```
$ ./script_nesting_fn
```

script_nesting_fn

```
#!/bin/tcsh
echo "#   iq   Nesting function (q)" > pb.nesting_fn
grep "Nesting function (q)= " epw4.out | awk '{print $4}' | nl >> pb.nesting_fn
```

You can use the gnuplot script fig5.plt to plot the nesting function $f_{\rm nest}(\mathbf{q})$ and the electron-phonon coupling strength of each phonon $\lambda_{\mathbf{q}\nu}$. You should get something similar to Fig. 5.

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

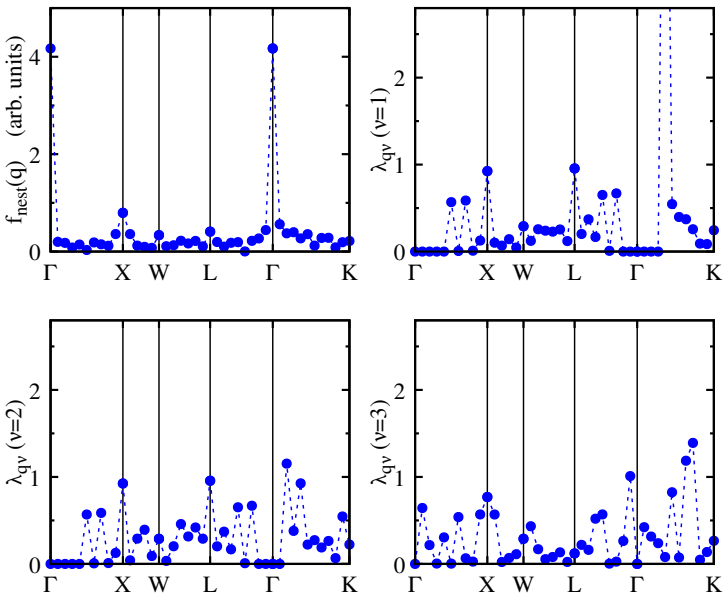

Fig. 5 The $\mathbf{q}$-point dependence of the nesting function $f_{\rm nest}(\mathbf{q})$ and the electron-phonon coupling strength of each phonon $\lambda_{\mathbf{q}\nu}$ (ν=1,2, and 3).

Note: Comparing the nesting function $f_{\rm nest}(\mathbf{q})$ with $\lambda_{\mathbf{q}\nu}$ might be useful for decomposing and analyzing the significant contributions to $\lambda(\omega_j)$.

Exercise 2

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

Go to `exercise2`:

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

▶ 1st step: Run a self-consistent calculation on a homogeneous 12x12x12 **k**-point grid and a phonon calculation on a homogeneous 3x3x3 **q**-point grid using the following 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, but it must be much smaller for actual calculations.

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

job.ph

```
#!/bin/bash
#SBATCH -J job.ph             # Job name
#SBATCH -N 1                  # Total # of nodes
#SBATCH --ntasks-per-node 24
#SBATCH -t 00:30:00           # Run time (hh:mm:ss)
#SBATCH -A DMR23030
#SBATCH -p skx
#SBATCH --reservation=NSF_Summer_School_Wed

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

ibrun $PATHQE/bin/pw.x -nk 6 -in scf.in > scf.out
ibrun $PATHQE/bin/ph.x -nk 6 -in ph.in > 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
 8 8 8 0 0 0
```

ph.in

```
--
&inputph
  prefix   = 'mgb2',
```

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

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

```
Dynamical matrices for ( 3, 3, 3)  uniform grid of q-points
(   6 q-points):
  N         xq(1)         xq(2)         xq(3)
  1   0.000000000   0.000000000   0.000000000
  2   0.000000000   0.000000000   0.291867841
  3   0.000000000   0.384900179   0.000000000
  4   0.000000000   0.384900179   0.291867841
  5   0.333333333   0.577350269   0.000000000
  6   0.333333333   0.577350269   0.291867841
```

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

```
$ python3 /work2/05193/sabyadk/stampede3/EPWSchool2024/q-e/EPW/bin/pp.py
```

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

▶ 3rd step: Do a non self-consistent calculation on a 6×6×6 **uniform** and Γ-**centered grid between [0,1[in crystal coordinates** and an EPW calculation for the anisotropic superconducting properties using the jollowing jobscript (`job.epw1`) and input files (`nscf.in` and `epw1.in`):

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

```
$ /work2/05193/sabyadk/stampede3/EPWSchool2024/q-e/external/wannier90/utility/kmesh.pl 6 6 6
```

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

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

and uncomment the following three lines

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

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

Note 4: By default, the Migdal-Eliashberg equations are solved using the Fermi surface restriction (FSR) approximation.

```
$ cd ../epw1-FSR
$ sbatch job.epw1
```

job.epw1

```
#!/bin/bash
#SBATCH -J job.epw1           # Job name
#SBATCH -N 1                  # Total # of nodes
#SBATCH --ntasks-per-node 48
#SBATCH -t 01:00:00           # Run time (hh:mm:ss)
#SBATCH -A DMR23030
```

```
#SBATCH -p skx
#SBATCH --reservation=NSF_Summer_School_Wed

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

ibrun $PATHQE/bin/pw.x -nk 48 -in scf.in > scf.out
#alternatively to re-run a scf calculation copy files from ../phonon/mgb2.save
#mkdir mgb2.save
#cp ../phonon/mgb2.save/charge-density.dat mgb2.save/
#cp ../phonon/mgb2.save/data-file-schema.xml mgb2.save/

ibrun $PATHQE/bin/pw.x -nk 48 -in nscf.in > nscf.out
ibrun $PATHQE/bin/epw.x -nk 48 -in epw1.in > epw1.out
```

nscf.in

```
&control
 calculation = 'nscf',
 prefix      = 'mgb2',
 pseudo_dir  = '../../pseudo/',
 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        = 16
/
&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     =  0              ! less I/O but more 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       = 0.4           ! Fermi window thickness [eV]
  degaussw      = 0.1           ! smearing in energy-conserving delta functions in [eV]
  degaussq      = 0.5           ! smearing for sum over q in the e-ph coupling in [meV]

  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
  lacon     = .false.           ! not perform the analytic continuation with the iterative procedure

  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    = 10                ! number of temperature points at which the ME eqs. are solved
  temps     = 10 55             ! 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:              136
```

- 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            200
......
Number selected, total           7900           7921
We only need to compute     7979 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 =       808 out of      3234

Progression iq (fine) =        100/      7979
Progression iq (fine) =        200/      7979
......
Progression iq (fine) =       7900/      7979
             Fermi level (eV) =    0.746982272596166D+01
DOS(states/spin/eV/Unit Cell) =    0.340839995969982D+00
       Electron smearing (eV) =    0.100000000000000D+00
            Fermi window (eV) =    0.400000000000000D+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) =   7.469823
       3   bands within the Fermi window
```

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

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

$$\lambda_{n\mathbf{k},m\mathbf{k+q}}(\omega_j) = N_\mathrm{F} \sum_\nu \frac{2\omega_{\mathbf{q}\nu}}{\omega_j^2 + \omega_{\mathbf{q}\nu}^2} |g_{mn\nu}(\mathbf{k},\mathbf{q})|^2 \quad (7)$$

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

$$\lambda_{n\mathbf{k}}(\omega_j) = \sum_m \int \frac{d\mathbf{q}}{\Omega_\mathrm{BZ}} \frac{\delta(\epsilon_{m\mathbf{k+q}} - \epsilon_\mathrm{F})}{N_\mathrm{F}} \lambda_{n\mathbf{k},m\mathbf{k+q}}(\omega_j) \quad (8)$$

  The isotropic electron-phonon coupling strength takes the form:

$$\lambda(\omega_j) = \sum_n \int \frac{d\mathbf{k}}{\Omega_\mathrm{BZ}} \frac{\delta(\epsilon_{n\mathbf{k}} - \epsilon_\mathrm{F})}{N_\mathrm{F}} \lambda_{n\mathbf{k}}(\omega_j) \quad (9)$$

  The standard electron-phonon coupling strength λ found in the literature corresponds to setting $\omega_j = 0$ in Eq. (9).

  The isotropic Eliashberg spectral function takes the following form:

$$\alpha^2F(\omega) = \frac{1}{N_\mathrm{F}} \sum_{mn\nu} \int \frac{d\mathbf{k}}{\Omega_\mathrm{BZ}} \int \frac{d\mathbf{q}}{\Omega_\mathrm{BZ}} |g_{mn\nu}(\mathbf{k},\mathbf{q})|^2 \, \delta(\omega - \omega_{\mathbf{q}\nu}) \delta(\epsilon_{n\mathbf{k}} - \epsilon_\mathrm{F}) \delta(\epsilon_{m\mathbf{k+q}} - \epsilon_\mathrm{F}) \quad (10)$$

- 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:

$$Z_{n\mathbf{k}}(i\omega_j) = 1 + \frac{\pi T}{\omega_j N_\mathrm{F}} \sum_{mj'} \int \frac{d\mathbf{q}}{\Omega_\mathrm{BZ}} \frac{\omega_{j'}}{\sqrt{\omega_{j'}^2 + \Delta_{m\mathbf{k}+\mathbf{q}}^2(i\omega_{j'})}} \times \lambda_{n\mathbf{k},m\mathbf{k}+\mathbf{q}}(\omega_j - \omega_{j'}) \delta(\epsilon_{m\mathbf{k}+\mathbf{q}} - \epsilon_\mathrm{F}) \tag{11}$$

$$Z_{n\mathbf{k}}(i\omega_j)\Delta_{n\mathbf{k}}(i\omega_j) = \frac{\pi T}{N_\mathrm{F}} \sum_{mj'} \int \frac{d\mathbf{q}}{\Omega_\mathrm{BZ}} \frac{\Delta_{m\mathbf{k}+\mathbf{q}}(i\omega_{j'})}{\sqrt{\omega_{j'}^2 + \Delta_{m\mathbf{k}+\mathbf{q}}^2(i\omega_{j'})}} \times \left[\lambda_{n\mathbf{k},m\mathbf{k}+\mathbf{q}}(\omega_j - \omega_{j'}) - \mu_\mathrm{c}^*\right] \delta(\epsilon_{m\mathbf{k}+\mathbf{q}} - \epsilon_\mathrm{F}), \tag{12}$$

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

```
================================================================
Solve anisotropic Eliashberg equations
================================================================
......
Electron-phonon coupling strength =    0.6411965

Estimated Allen-Dynes Tc =    28.703420 K for muc =    0.05000

Estimated w_log in Allen-Dynes Tc =    58.879430 meV

Estimated BCS superconducting gap =     4.353306 meV

Estimated Tc from machine learning model =    30.490706 K

WARNING WARNING WARNING

The code may crash since tempsmax =   55.000 K is larger than Allen-Dynes Tc =    28.703 K
Actual number of frequency points (     1) =     92 for uniform       sampling

temp(  1) =     10.00000 K

Solve anisotropic Eliashberg equations on imaginary-axis

Total number of frequency points nsiw(     1) =     92
Cutoff frequency wscut =     0.5000 eV
Maximum frequency =         0.4954 eV
broyden mixing factor =      0.70000

Size of allocated memory per pool: ~=    0.3804 Gb
   iter      ethr          znormi      deltai [meV]
     1   3.409755E+00   1.619880E+00   4.689818E+00
     2   5.333955E-02   1.618475E+00   4.967665E+00
     ......
    19   8.430125E-04   1.612711E+00   5.904314E+00
Convergence was reached in nsiter =     19
```

- 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   1.611909E+00   5.413260E+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 $\lambda_{n\mathbf{k},m\mathbf{k+q}}(0)$ on the Fermi surface.

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

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

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

The cumulative electron-phonon coupling strength λ takes the following form:

$$\lambda(\omega) = \int_0^{\omega} d\omega' \frac{2\alpha^2F(\omega')}{\omega'} \tag{13}$$

You can use the `gnuplot` script `fig6.plt` to plot. You should get something similar to Fig. 6.

```
$ gnuplot fig6.plt
$ evince fig6.pdf
```

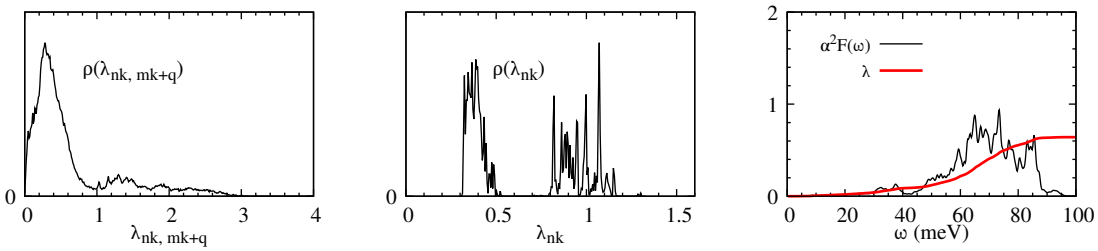

Fig. 6 Left: The anisotropic electron-phonon coupling strength $\lambda_{n\mathbf{k},m\mathbf{k+q}}(0)$ (from `mgb2.lambda_pairs`). Middle: The anisotropic electron-phonon coupling strength $\lambda_{n\mathbf{k}}(0)$ on the Fermi surface (from `mgb2.lambda_k_pairs`). Right: The isotropic Eliashberg spectral function $\alpha^2F(\omega)$ (columns 1:2 from `mgb2.a2f`) and cumulative electron-phonon coupling strength λ (columns 1:12 from `mgb2.a2f`).

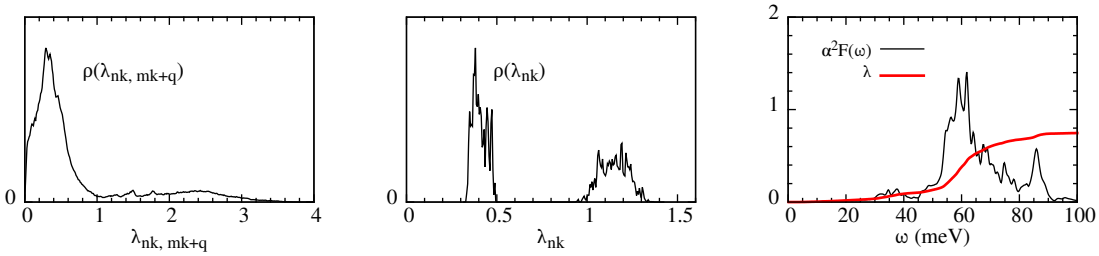

Fig. 6(R) At convergence you should get something close to this figure (see Phys. Rev. B **87**, 024505 (2013) for fully converged calculation parameters).

▶ 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. 7 at 10 K. The file `fig7.pdf` is too large (∼20MB) to open while connecting to a remote server: To avoid opening it directly, you can use `pdftoppm` command to show the plot.

```
$ gnuplot fig7.plt
$ pdftoppm -png -r 100 fig7.pdf fig7
$ display fig7-1.png
```

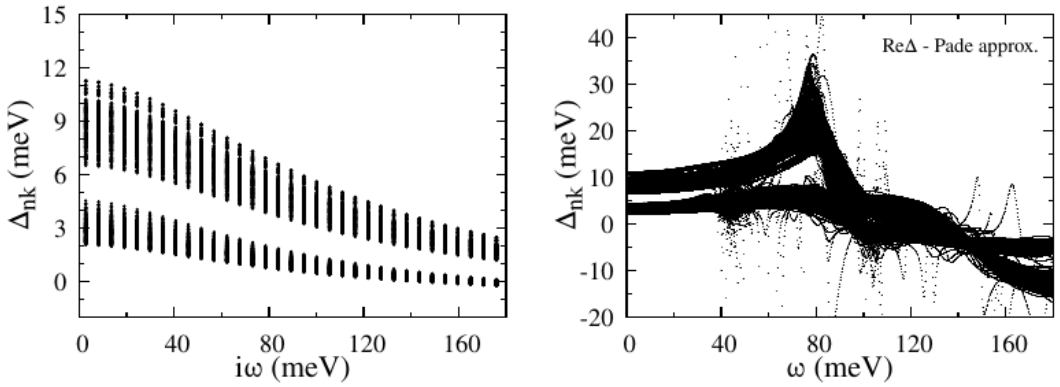

Fig. 7 Left: Superconducting gap along the imaginary axis (columns 1:4 from `mgb2.imag_aniso_010.00`). Right: Superconducting gap along the real axis (columns 1:5 from `mgb2.pade_aniso_010.00` - this file is about 70MB).

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

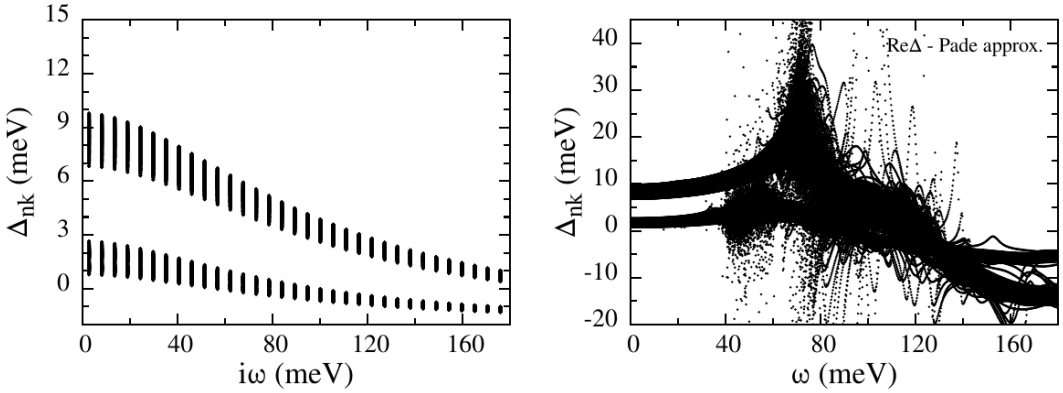

Fig. 7(R) At convergence you should get something close to this figure (see Phys. Rev. B **87**, 024505 (2013) for fully converged calculation parameters).

▶ 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 `fig8.plt`.

```
$ gnuplot fig8.plt
$ evince fig8.pdf
```

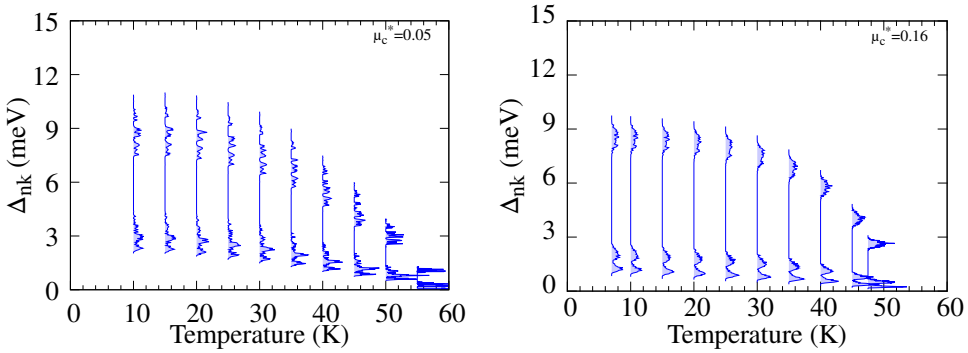

Fig. 8 Calculated anisotropic superconducting gap of MgB_2 on the Fermi surface as a function of temperature. At convergence you should get the right hand-side figure (see Phys. Rev. B **87**, 024505 (2013) for fully converged calculation parameters).

▶ 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:

$$\frac{N_S(\omega)}{N_{\rm F}} = \sum_n \int \frac{d\mathbf{k}}{\Omega_{\rm BZ}} \frac{\delta(\epsilon_{n\mathbf{k}} - \epsilon_{\rm F})}{N_{\rm F}} {\rm Re}\left[\omega / \sqrt{\omega^2 - \Delta^2_{n\mathbf{k}}(\omega)}\right] \tag{14}$$

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

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`.

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

You should get something similar to Fig. 9 (left) at 10 K:

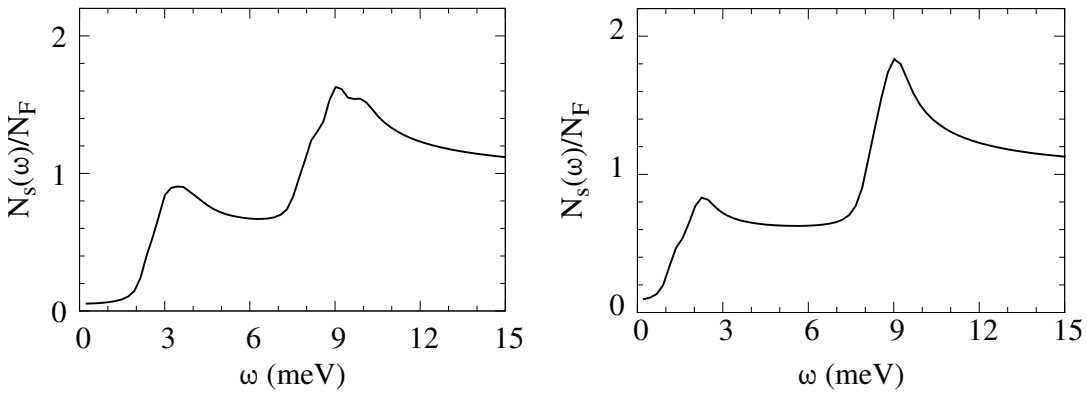

Fig. 9 Calculated $N_S(\omega)/N_{\rm F}$ as a function of frequency at 10 K. At convergence you should get something closer to the right hand-side figure (see Phys. Rev. B **87**, 024505 (2013) for fully converged calculation parameters). (Note: the second column of `mgb2.qdos_XX` should be divided by the value of DOS from the `epw1.out`).

▶ 8th step: Solve the anisotropic full-bandwidth (FBW) Migdal-Eliashberg equations. The process to obtain the electron-phonon matrix elements is exactly the same as the previous FSR calculation, so the `mgb2.ephmat` directory can be reused. Let us create symbolic links for the necessary files, `mgb2.ephmat`, `mgb2.a2f`, and all the `fmt` files and set `ephwrite = .false.`. 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 1: Here, we have fixed the chemical potential at the Fermi level.

Note 2: The a2f file (`prefix.a2f`) is not strictly required, but if it is missing from the current directory, $\alpha^2F(\omega)$ and $\lambda(\omega_j)$ will be recalculated.

```
$ cd ../epw2-FBW
$ ln -s ../epw1-FSR/*fmt .
$ ln -s ../epw1-FSR/mgb2.ephmat .
$ ln -s ../epw1-FSR/mgb2.a2f .
$ sbatch job.epw2
```

job.epw2

```
#!/bin/bash
#SBATCH -J job.epw2           # Job name
#SBATCH -N 1                  # Total # of nodes
#SBATCH --ntasks-per-node 48
#SBATCH -t 00:30:00           # Run time (hh:mm:ss)
#SBATCH -A DMR23030
#SBATCH -p skx
#SBATCH --reservation=NSF_Summer_School_Wed

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

ibrun $PATHQE/bin/epw.x -nk 48 -in epw2.in > epw2.out
```

epw2.in

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

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 FBW Migdal-Eliashberg equations take the following form:

$$Z_{n\mathbf{k}}(i\omega_j) = 1 + \frac{T}{\omega_j N_\mathrm{F}} \sum_{mj'} \int \frac{d\mathbf{q}}{\Omega_\mathrm{BZ}} \frac{\omega_{j'} Z_{m\mathbf{k+q}}(i\omega_{j'})}{\theta_{m\mathbf{k+q}}(i\omega_{j'})} \lambda_{n\mathbf{k},m\mathbf{k+q}}(\omega_j - \omega_{j'}) \tag{15}$$

$$\chi_{n\mathbf{k}}(i\omega_j) = \frac{-T}{N_\mathrm{F}} \sum_{mj'} \int \frac{d\mathbf{q}}{\Omega_\mathrm{BZ}} \frac{\varepsilon_{m\mathbf{k+q}} - \mu + \chi_{m\mathbf{k+q}}(i\omega_{j'})}{\theta_{m\mathbf{k+q}}(i\omega_{j'})} \lambda_{n\mathbf{k},m\mathbf{k+q}}(\omega_j - \omega_{j'}) \tag{16}$$

$$\phi_{n\mathbf{k}}(i\omega_j) = \frac{T}{N_\mathrm{F}} \sum_{mj'} \int \frac{d\mathbf{q}}{\Omega_\mathrm{BZ}} \frac{\phi_{m\mathbf{k+q}}(i\omega_{j'})}{\theta_{m\mathbf{k+q}}(i\omega_{j'})} \left[\lambda_{n\mathbf{k},m\mathbf{k+q}}(\omega_j - \omega_{j'}) - \mu_\mathrm{c}^*\right] \tag{17}$$

where $\lambda_{n\mathbf{k},m\mathbf{k+q}}(\omega_j - \omega_{j'})$ 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 semi-empirical Coulomb parameter μ_c^* is provided as an input variable `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.

$$N_e = \sum_n \int \frac{d\mathbf{k}}{\Omega_{BZ}} \left(1 - 2T \sum_j \frac{\varepsilon_{n\mathbf{k}} - \mu_F + \chi_{n\mathbf{k}}(i\omega_j)}{\theta_{n\mathbf{k}}(i\omega_j)} \right) \tag{18}$$

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 =     0.5000 eV
Maximum frequency =         0.4954 eV
broyden mixing factor =      0.70000

Size of allocated memory per pool: ~=    0.1941 Gb
   iter      ethr          znormi      deltai [meV]   shifti [meV]       mu [eV]
     1   3.361099E+00   1.540525E+00   4.809028E+00   8.161334E-01   7.469823E+00
     2   8.078731E-02   1.518899E+00   5.264979E+00   1.230508E+00   7.469823E+00
     ......
    23   8.622406E-04   1.503663E+00   6.493878E+00   1.247068E+00   7.469823E+00
Convergence was reached in nsiter =     23
```

▶ 9th 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 `fig10.plt` and `fig11.plt`.

```
$ gnuplot fig10.plt
$ pdftoppm -png -r 100 fig10.pdf fig10
$ display fig10-1.png
```

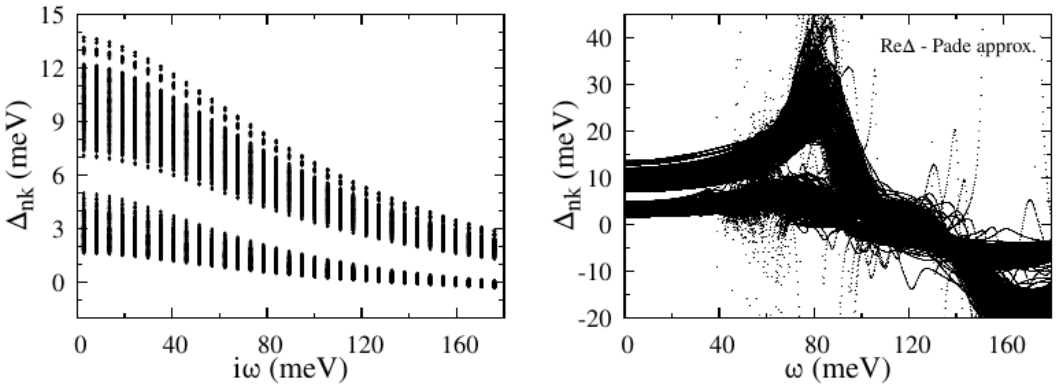

Fig. 10 Left: Superconducting gap along the imaginary axis (columns 1:4 from `mgb2.imag_aniso_010.00`) obtained from the FBW calculations. Right: Superconducting gap along the real axis (columns 1:5 from `mgb2.pade_aniso_010.00`).

```
$ gnuplot fig11.plt
$ evince fig11.pdf
```

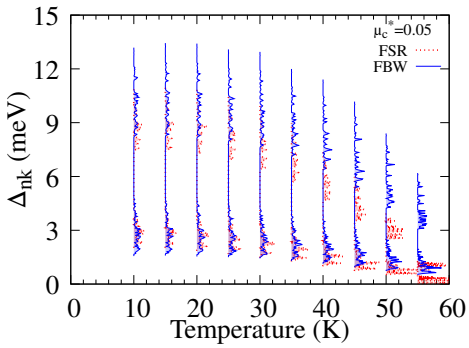

Fig. 11 Calculated anisotropic superconducting gap of MgB_2 on the Fermi surface as a function of temperature obtained from the FSR (red-dashed line) and FBW (blue-solid line) calculations.

▶ 10th step: (**Optional due to time limit**) Check how the results change with updating the chemical potential at each temperature (`muchem = .true.`).

```
$ cd ../epw3-FBW+mu
$ ln -s ../epw1-FSR/*fmt .
$ ln -s ../epw1-FSR/mgb2.ephmat .
$ ln -s ../epw1-FSR/mgb2.a2f .
$ sbatch job.epw3
```

job.epw3

```
#!/bin/bash
#SBATCH -J job.epw3           # Job name
#SBATCH -N 1                  # Total # of nodes
#SBATCH --ntasks-per-node 48
#SBATCH -t 00:30:00           # Run time (hh:mm:ss)
#SBATCH -A DMR23030
#SBATCH -p skx
#SBATCH --reservation=NSF_Summer_School_Wed

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

ibrun $PATHQE/bin/epw.x -nk 48 -in epw3.in > epw3.out
```

epw3.in

```
--
muchem      = .true.
```

You can use the `gnuplot` scripts `fig12.plt` and `fig13.plt` to compare the results of the superconducting gap with those from the previous FBW calculation with the fixed chemical potential.

```
$ gnuplot fig12.plt
$ pdftoppm -png -r 100 fig12.pdf fig12
$ display fig12-1.png
```

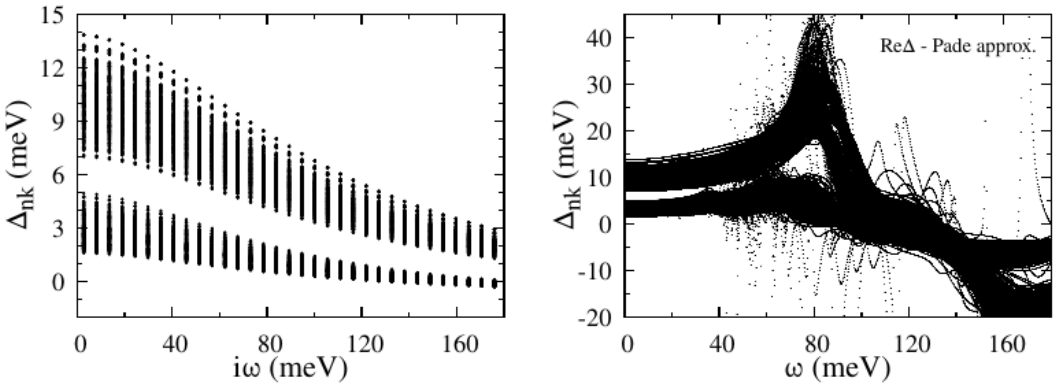

Fig. 12 Left: Superconducting gap along the imaginary axis (columns 1:4 from `mgb2.imag_aniso_010.00`) obtained from the FBW calculations with updating the chemical potential. Right: Superconducting gap along the real axis (columns 1:5 from `mgb2.pade_aniso_010.00`).

```
$ gnuplot fig13.plt
$ evince fig13.pdf
```

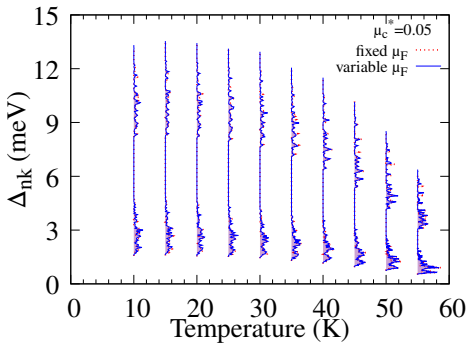

Fig. 13 Calculated anisotropic superconducting gap of MgB_2 on the Fermi surface as a function of temperature obtained with the fixed chemical potential (red-dashed line) and the variable chemical potential (blue-solid line).

Appendix: How to plot the superconducting gap on the Fermi surface with FermiSurfer

To visualize, open `mgb2.imag_aniso_gap0_XX.frmsf` with FermiSurfer.

`mgb2.imag_aniso_gap0_XX.frmsf` (XX = temperature) files were generated by setting `laniso = .true.` and `iverbosity = 2` in `epw1.in`. Each file contains the energy eigenvalues relative to the Fermi level, and can be visualized with FermiSurfer.

Note: FermiSurfer is not installed on Stampede3. Please try it locally.

Exercise 3 (Optional due to time limit)

In this tutorial we are going to calculate the superconducting properties of bcc Nb by solving the anisotropic Migdal-Eliashberg equations using sparse-ir sampling. The theory related to this tutorial can be found in the arXiv: 2404.11528.

Go to exercise3:

```
$ cd ../../exercise3
```

▶ 1st step: Run a self-consistent calculation on a homogeneous 12x12x12 **k**-point grid and a phonon calculation on a homogeneous 4x4x4 **q**-point grid using the following jobscript (`job.ph`) and input files (`scf.in` and `ph.in`) for Nb:

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

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

job.ph

```
#!/bin/bash
#SBATCH -J job.ph             # Job name
#SBATCH -N 1                  # Total # of nodes
#SBATCH --ntasks-per-node 24
#SBATCH -t 00:30:00           # Run time (hh:mm:ss)
#SBATCH -A DMR23030
#SBATCH -p skx
#SBATCH --reservation=NSF_Summer_School_Wed

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

ibrun $PATHQE/bin/pw.x -nk 6 -in scf.in > scf.out
ibrun $PATHQE/bin/ph.x -nk 6 -in ph.in > ph.out
```

scf.in

```
 &control
    calculation     = 'scf'
    prefix          = 'nb'
    restart_mode    = 'from_scratch'
    pseudo_dir   = '../../pseudo/',
    outdir          = './'
 /
 &system
    ibrav           = 3
    celldm(1)       = 6.252854867061436
    nat             = 1
    ntyp            = 1
    nbnd            = 24
    ecutwfc         = 40
    occupations     = 'smearing'
    smearing        = 'm-p'
    degauss         = 0.01
 /
 &electrons
    diagonalization = 'cg'
    mixing_mode = 'plain'
    mixing_beta     = 0.7
    conv_thr        = 1.0d-12
 /
ATOMIC_SPECIES
  Nb   92.91  Nb_ONCV_PBE-1.2.upf
ATOMIC_POSITIONS {crystal}
Nb            0.0000000000        0.0000000000        0.0000000000    0   0   0
K_POINTS {automatic}
 12 12 12 0 0 0
```

ph.in

```
--
&inputph
  prefix   = 'nb'
  fildvscf = 'dvscf'
  outdir   = './',
```

```
 fildyn   = 'Nb.dyn'
 ldisp = .true.
 nq1=4,
 nq2=4,
 nq3=4,
 tr2_ph   =  1.0d-14
/
```

During the run, notice the irreducible (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   0.000000000  -0.250000000   0.250000000
  3   0.000000000   0.500000000  -0.500000000
  4  -0.250000000   0.750000000  -0.500000000
  5   0.000000000   0.000000000   0.500000000
  6   0.000000000   0.750000000  -0.250000000
  7   0.500000000  -0.500000000   0.500000000
  8   0.000000000   0.000000000  -1.000000000
```

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

```
$ python3 /work2/05193/sabyadk/stampede3/EPWSchool2024/q-e/EPW/bin/pp.py
```

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

▶ 3rd step: Do a non self-consistent calculation on a 4×4×4 **uniform** and Γ-**centered grid between [0,1[in crystal coordinates** and an EPW calculation for the electron-phonon coupling strength λ using the following jobscript (`job.epw1`) and input files (`nscf.in` and `epw1.in`):

Note 1: The homogeneous **k** grid for the non self-consistent calculations can be generated using the script `kmesh.pl`
`$ /work2/05193/sabyadk/stampede3/EPWSchool2024/q-e/external/wannier90/utility/kmesh.pl 4 4 4`

```
$ cd ../epw1-lambda
$ sbatch job.epw1
```

job.epw1

```
#!/bin/bash
#SBATCH -J job.epw1           # Job name
#SBATCH -N 1                  # Total # of nodes
#SBATCH --ntasks-per-node 48
#SBATCH -t 01:00:00           # Run time (hh:mm:ss)
#SBATCH -A DMR23030
#SBATCH -p skx
#SBATCH --reservation=NSF_Summer_School_Wed

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

ibrun $PATHQE/bin/pw.x -nk 48 -in scf.in > scf.out
#alternatively to re-run a scf calculation copy files from ../phonon/nb.save
#mkdir nb.save
#cp ../phonon/nb.save/charge-density.dat nb.save/
#cp ../phonon/nb.save/data-file-schema.xml nb.save/

ibrun $PATHQE/bin/pw.x -nk 48 -in nscf.in > nscf.out
ibrun $PATHQE/bin/epw.x -nk 48 -in epw1.in > epw1.out
```

nscf.in

```
&control
   calculation     = 'nscf'
   prefix          = 'nb'
```

```
    restart_mode    = 'from_scratch'
    pseudo_dir   = '../../pseudo/',
    outdir          = './'
 /
 &system
    ibrav           = 3
    celldm(1)       = 6.252854867061436
    nat             = 1
    ntyp            = 1
    nbnd            = 24
    ecutwfc         = 40
    occupations     = 'smearing'
    smearing        = 'm-p'
    degauss         = 0.01
 /
 &electrons
    diagonalization = 'cg'
    mixing_mode = 'plain'
    mixing_beta     = 0.7
    conv_thr        = 1.0d-10
 /
ATOMIC_SPECIES
  Nb   92.91  Nb_ONCV_PBE-1.2.upf
ATOMIC_POSITIONS {crystal}
Nb            0.0000000000        0.0000000000        0.0000000000    0   0   0
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
 ...
```

```
--                                                                                epw1.in
&inputepw
  prefix      = 'nb',
  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     =  0              ! less I/O but more memory is required
  vme         = 'dipole'

  nbndsub     =  9
  bands_skipped = 'exclude_bands = 1:4'

  wannierize  = .true.
  num_iter    = 0
  dis_win_min = 10
  dis_froz_min = 11
  dis_froz_max= 28
  proj(1) = 'Nb:s'
  proj(2) = 'Nb:p'
  proj(3) = 'Nb:d'

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

  fsthick     = 0.5           ! Fermi window thickness [eV]
  degaussw    = 0.1           ! smearing in the energy-conserving delta functions in [eV]
  degaussq    = 0.5           ! smearing for sum over q in the e-ph coupling in [meV]

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

  nk1         = 4             ! dimensions of the coarse electronic grid
  nk2         = 4
  nk3         = 4
```

```
 nq1         = 4             ! dimensions of the coarse phonon grid
 nq2         = 4
 nq3         = 4

 mp_mesh_k   = .true.        ! use irreduciable electronic fine mesh
 nkf1 = 20                   ! dimensions of the fine electronic grid
 nkf2 = 20
 nkf3 = 20
 nqf1 = 20                   ! dimensions of the fine phonon grid
 nqf2 = 20
 nqf3 = 20
/
```

With the above input, we are instructing EPW to:

- Fourier-transform the electron-phonon matrix elements from a coarse 4×4×4 to a dense 20×20×20 $\mathbf{k}$-point grid and from a coarse 4×4×4 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:    20   20   20
Size of k point mesh for interpolation:        512
Max number of k points per pool:                12
```

- Pre-compute the $\mathbf{q}$-points that fall within the `fsthick` window. 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            200
......
Number selected, total           8000           8000
We only need to compute     8000 q-points
```

- Write on disk in the `nb.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:       256

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

Nr irreducible k-points within the Fermi shell =       144 out of       256

Progression iq (fine) =        100/      8000
Progression iq (fine) =        200/      8000
......
Progression iq (fine) =       8000/      8000
             Fermi level (eV) =    0.178752569141474D+02
DOS(states/spin/eV/Unit Cell) =    0.654658687855450D+00
       Electron smearing (eV) =    0.100000000000000D+00
            Fermi window (eV) =    0.500000000000000D+00

Finish writing .ephmat files
```

▶ 4th step: Plot the isotropic and anisotropic electron-phonon coupling strength. You can use the `gnuplot` script `fig14.plt` to plot. You should get something similar to Fig. 14.

```
$ gnuplot fig14.plt
$ evince fig14.pdf
```

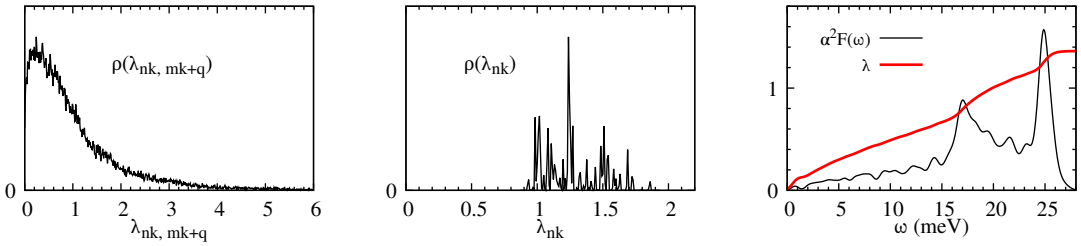

Fig. 14 Left: The anisotropic electron-phonon coupling strength $\lambda_{n\mathbf{k},m\mathbf{k+q}}(0)$ (from `nb.lambda_pairs`). Middle: The anisotropic electron-phonon coupling strength $\lambda_{n\mathbf{k}}(0)$ on the Fermi surface (from `nb.lambda_k_pairs`). Right: The isotropic Eliashberg spectral function $\alpha^2F(\omega)$ (columns 1:2 from `nb.a2f`) and cumulative electron-phonon coupling strength λ (columns 1:12 from `nb.a2f`).

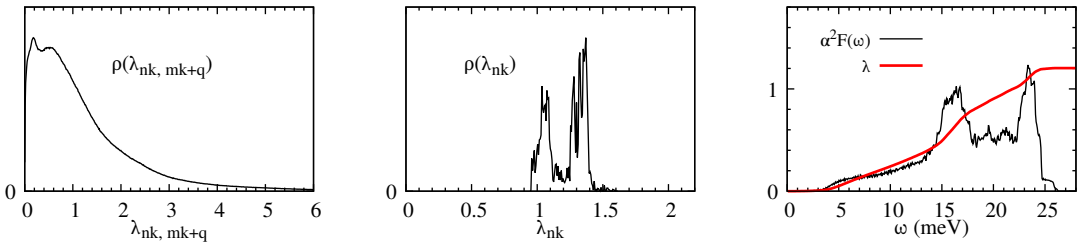

Fig. 14(R) At convergence you should get something close to this figure (see arXiv: 2404.11528 for fully converged calculation parameters).

▶ 5th step: Solve the anisotropic full-bandwidth (FBW) Migdal-Eliashberg equations using IR basis functions. Since the `nb.ephmat` directory has already been obtained, let us create symbolic links for the necessary files, `nb.ephmat` and all the `fmt` files and set `ephwrite = .false.`. After this, do an EPW calculation using the following jobscript (`job.epw2`) and input file (`epw2.in`; only differences with respect to `../epw1-lambda/epw1.in` file are shown below):

```
$ cd ../epw2-mustar
$ ln -s ../epw1-lambda/*fmt .
$ ln -s ../epw1-lambda/nb.ephmat .
$ ln -s ../epw1-lambda/nb.a2f .
$ sbatch job.epw2
```

job.epw2

```
#!/bin/bash
#SBATCH -J job.epw2           # Job name
#SBATCH -N 2                  # Total # of nodes
#SBATCH --ntasks-per-node 48
#SBATCH -t 00:30:00           # Run time (hh:mm:ss)
#SBATCH -A DMR23030
#SBATCH -p skx
#SBATCH --reservation=NSF_Summer_School_Wed
```

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

ibrun $PATHQE/bin/epw.x -nk 96 -in epw2.in > epw2.out
```

```
--                                                                                                    epw2.in
ep_coupling = .false.
elph        = .false.
epwwrite    = .false.
epwread     = .true.
wannierize  = .false.

iverbosity  = 0             ! if you want to get .frmsf files, change this to 2.

ephwrite    = .false.

laniso  = .true.            ! solve anisotropic ME eqs.
limag = .true.              ! solve ME eqs. imaginary axis
fbw    = .true.             ! solve FBW ME eqs.
muchem = .true.             ! update the chemical potential at every temperature
gridsamp = 2                ! If gridsamp = 2, use sparse-ir sampling

nsiter = 500                ! convergence threshold for solving ME eqs. on imaginary axis
filirobj = '../../irobjs/ir_nlambda6_ndigit8.dat'
                            ! The file containing the sparse samping points and the IR functions.
broyden_beta = -0.7         ! If this factor is negative, use the linear mixing instead Broyden mixing.

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

temps  = 0.2 6.0 10.0 12.0 13.0 13.5
                            ! temperature list

nsiter = 500                ! number of self-consistent iterations when solving ME eqs.
muc    = 0.24               ! effective Coulomb potential used in ME eqs.
```

Note 1: Starting from version 5.8, it became possible to read `ephmatXX` files using a different number of pools than the number used when writing them. Due to this change, the recent version cannot read `prefix.ephmat` written with older versions.
Note 2: Especially when solving anisotropic ME equations, linear mixing can sometimes be more efficient than Broyden mixing.
Note 3: $\lambda = 10^{\texttt{nlambda}}$ determines the maximum Matsubara frequency, while $\epsilon_{\rm IR} = 10^{-\texttt{ndigit}}$ controls the sparseness of sampling points. If `nlambda` or `ndigit` is large, the number of sampling points for Matsubara frequency increases.
Note 4: Since the EPW code cannot generate the IR basis functions, the IR basis functions are read from the input file (`filirobj`). If you want to change `nlambda` and `ndigit`, you should specify a different file. To generate this object file by yourself, you can use the Python `sparse-ir` library. For more details, refer to the following URL: https://github.com/SpM-lab/sparse-ir-fortran.
Note 5: In this calculation, the value of `wscut` does not affect the calculation results, as the Matsubara frequencies are read from the file when using sparse-ir sampling (`gridsamp = 2`), making `wscut` ignored. `wscut` is only used to determine the upper limit of the real frequencies when performing analytic continuation.

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 FBW Migdal-Eliashberg equations take the following form:

$$Z_{n\mathbf{k}}(i\omega_j) = 1 + \frac{T}{\omega_j N_{\rm F}} \frac{1}{N_{\rm q}} \sum_{\mathbf{q}m}^{\rm fsthick} \sum_{j'} \frac{\omega_{j'} Z_{m\mathbf{k}+\mathbf{q}}(i\omega_{j'})}{\theta_{m\mathbf{k}+\mathbf{q}}(i\omega_{j'})} \lambda_{n\mathbf{k},m\mathbf{k}+\mathbf{q}}(\omega_j - \omega_{j'}) \tag{19}$$

$$\chi_{n\mathbf{k}}(i\omega_j) = \frac{-T}{N_\mathrm{F}}\frac{1}{N_\mathrm{q}}\sum_{\mathbf{q}m}^{\mathrm{fsthick}}\sum_{j'}\frac{\varepsilon_{m\mathbf{k}+\mathbf{q}} - \mu + \chi_{m\mathbf{k}+\mathbf{q}}(i\omega_{j'})}{\theta_{m\mathbf{k}+\mathbf{q}}(i\omega_{j'})}\lambda_{n\mathbf{k},m\mathbf{k}+\mathbf{q}}(\omega_j - \omega_{j'}) \tag{20}$$

$$\phi_{n\mathbf{k}}(i\omega_j) = \frac{T}{N_\mathrm{F}}\frac{1}{N_\mathrm{q}}\sum_{\mathbf{q}m}^{\mathrm{fsthick}}\sum_{j'}\frac{\phi_{m\mathbf{k}+\mathbf{q}}(i\omega_{j'})}{\theta_{m\mathbf{k}+\mathbf{q}}(i\omega_{j'})}\left[\lambda_{n\mathbf{k},m\mathbf{k}+\mathbf{q}}(\omega_j - \omega_{j'}) - \mu_\mathrm{c}^*\right] \tag{21}$$

where $\sum_{\mathbf{q}m}^{\mathrm{fsthick}}$ represents the summation over the final states $(\boldsymbol{k}+\boldsymbol{q}, m)$ lying within the `fsthick` window. 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.

$$N_\mathrm{e} = \frac{1}{N_\mathrm{k}}\sum_{\mathbf{k}n}^{\mathrm{fsthick}}\left(1 - 2T\sum_{j}\frac{\varepsilon_{n\mathbf{k}} - \mu_\mathrm{F} + \chi_{n\mathbf{k}}(i\omega_j)}{\theta_{n\mathbf{k}}(i\omega_j)}\right) \tag{22}$$

where $\sum_{\mathbf{k}n}^{\mathrm{fsthick}}$ represents the summation over the states $(\boldsymbol{k}, n)$ lying within the `fsthick` window.

```
================================================================
Solve full-bandwidth anisotropic Eliashberg equations
================================================================
......

Start reading ir object file

Finish reading ir object file

Actual number of frequency points (     1) =     48 for sparse-ir     sampling

temp(  1) =      0.20000 K

Solve full-bandwidth anisotropic Eliashberg equations on imaginary-axis

Total number of frequency points nsiw(     1) =     48
Parameters for IR basis: Lambda =     1.00E+06, eps_IR =     1.00E-08
The noise reduction will be performed using the threshold of     1.00E-05
Maximum frequency =       131.8778 eV
linear mixing factor =      0.70000
mixing factor = 0.2 is used for the first three iterations.

   iter      ethr          znormi      deltai [meV]   shifti [meV]       mu [eV]
     1   1.129460E+00   2.444918E+00   1.657449E+00   1.255267E+01   1.786864E+01
     2   1.654835E-01   2.347088E+00   1.573371E+00   1.360830E+01   1.787025E+01
     ......
    23   8.826285E-04   2.239696E+00   2.463422E+00   1.447918E+01   1.787829E+01
Convergence was reached in nsiter =     23
```

▶ 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 `nb.imag_aniso_gap0_XX` files, where XX stands for the supplied temperatures. Use the `gnuplot` script `fig15.plt`.

```
$ gnuplot fig15.plt
$ evince fig15.pdf
```

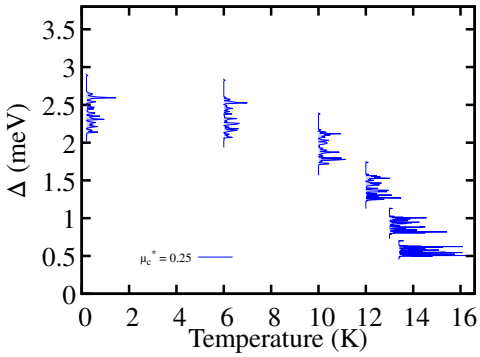

Fig. 15 Calculated anisotropic superconducting gap of Nb on the Fermi surface as a function of temperature.

▶ 7th step: Run a self-consistent calculation on a homogeneous 12x12x12 **k**-point grid and do a non self-consistent calculation on a 20×20×20 $\mathbf{k_c}$-point grid.

```
$ cd ../outerbands
$ sbatch job.bands
```

job.bands

```
#!/bin/bash
#SBATCH -J job.bands          # Job name
#SBATCH -N 1                  # Total # of nodes
#SBATCH --ntasks-per-node 24
#SBATCH -t 00:30:00           # Run time (hh:mm:ss)
#SBATCH -A DMR23030
#SBATCH -p skx
#SBATCH --reservation=NSF_Summer_School_Wed

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

ibrun $PATHQE/bin/pw.x -nk 6 -in scf.in > scf.out
#alternatively to re-run a scf calculation copy files from ../phonon/nb.save
#mkdir nb.save
#cp ../phonon/nb.save/charge-density.dat nb.save/
#cp ../phonon/nb.save/data-file-schema.xml nb.save/

ibrun $PATHQE/bin/pw.x -nk 6 -in nscf_outerbands.in > nscf_outerbands.out
```

nscf_outerbands.in

```
&control
   calculation     = 'bands'
   prefix          = 'nb'
   restart_mode    = 'from_scratch'
   pseudo_dir   = '../../pseudo/',
   outdir          = './'
   verbosity       = 'high'
/
&system
   ibrav           = 3
   celldm(1)       = 6.252854867061436
   nat             = 1
   ntyp            = 1
   nbnd            = 24
   ecutwfc         = 40
   occupations     = 'smearing'
   smearing        = 'm-p'
   degauss         = 0.01
/
&electrons
   diagonalization = 'cg'
```

```
    mixing_mode = 'plain'
    mixing_beta     = 0.7
    conv_thr        = 1.0d-10
 /
ATOMIC_SPECIES
  Nb   92.91  Nb_ONCV_PBE-1.2.upf
ATOMIC_POSITIONS {crystal}
Nb            0.0000000000        0.0000000000        0.0000000000    0   0   0
K_POINTS {automatic}
 20 20 20 0 0 0
```

▶ 8th step: Extract the eigenenergies into a new file (`nb.bands.20x20x20.dat`) using `nscf2supercond.x`.

```
$ /work2/05193/sabyadk/stampede3/EPWSchool2024/q-e/EPW/bin/nscf2supercond.x < nscf2supercond.in
```

nscf2supercond.in

```
&bands
   outdir = './',
   prefix='nb',
   filband='nb.bands.20x20x20.dat',
/
```

▶ 9th step: Solve the anisotropic full-bandwidth (FBW) Migdal-Eliashberg equations using IR basis functions with high-energy bands. Similar to step 5, create symbolic links for the necessary files, `nb.ephmat` and all the `fmt` files and set `ephwrite = .false.`. After this, do an EPW calculation using the following jobscript (`job.epw2`) and input file (`epw3.in`; only differences with respect to `../epw2-mustar/epw2.in` file are shown below):

```
$ cd ../epw3-outerbands
$ ln -s ../epw1-lambda/*fmt .
$ ln -s ../epw1-lambda/nb.ephmat .
$ ln -s ../epw1-lambda/nb.a2f .
$ sbatch job.epw3
```

job.epw3

```
#!/bin/bash
#SBATCH -J job.epw3           # Job name
#SBATCH -N 2                  # Total # of nodes
#SBATCH --ntasks-per-node 48
#SBATCH -t 01:00:00           # Run time (hh:mm:ss)
#SBATCH -A DMR23030
#SBATCH -p skx
#SBATCH --reservation=NSF_Summer_School_Wed

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

ibrun $PATHQE/bin/epw.x -nk 96 -in epw3.in > epw3.out
```

epw3.in

```
--
broyden_beta = -0.5         ! If this factor is negative, use the linear mixing instead Broyden mixing.

icoulomb = 1                ! To account for the outside fsthick window, set icoulomb = 1
filnscf_coul = '../outerbands/nb.bands.20x20x20.dat'
                            ! The file containing the data of eigenenergies.
emax_coulomb =  15.0d0      ! Upper limit of outer window in [eV]
emin_coulomb = -15.0d0      ! Lower limit of outer window in [eV]

muc    = 0.429              ! ab-initio Coulomb parameter used in ME eqs.
```

Note 1: By taking into account the high energy bands, we can use the ab-initio Coulomb parameter μ_c instead of the semi-empirical Coulomb potential μ_c^*. The `muc` value is taken from Phys. Rev. B **101**, 134511 (2020).

Note 2: The mixing factor (broyden_beta) is set to a smaller value because it is difficult to achieve convergence in self-consistent calculations with icoulomb = 1.

The anisotropic FBW Migdal-Eliashberg equations with considering the high-energy states are solved self-consistently on the imaginary frequency axis by setting the keywords fbw = .true., eliashberg = .true., laniso = .true., and limag = .true., gridsamp = 2, and icoulomb = 1 in the EPW input file.

The revised version of anisotropic FBW Migdal-Eliashberg equations take the following form:

$$Z_{n\mathbf{k}}(i\omega_j) = 1 + \frac{T}{\omega_j N_\mathrm{F}} \frac{1}{N_\mathrm{q}} \sum_{\mathbf{q}m}^{\mathrm{fsthick}} \sum_{j'} \frac{\omega_{j'} Z_{m\mathbf{k}+\mathbf{q}}(i\omega_{j'})}{\theta_{m\mathbf{k}+\mathbf{q}}(i\omega_{j'})} \lambda_{n\mathbf{k},m\mathbf{k}+\mathbf{q}}(\omega_j - \omega_{j'}) \tag{23}$$

$$\chi_{n\mathbf{k}}(i\omega_j) = \frac{-T}{N_\mathrm{F}} \frac{1}{N_\mathrm{q}} \sum_{\mathbf{q}m}^{\mathrm{fsthick}} \sum_{j'} \frac{\varepsilon_{m\mathbf{k}+\mathbf{q}} - \mu + \chi_{m\mathbf{k}+\mathbf{q}}(i\omega_{j'})}{\theta_{m\mathbf{k}+\mathbf{q}}(i\omega_{j'})} \lambda_{n\mathbf{k},m\mathbf{k}+\mathbf{q}}(\omega_j - \omega_{j'}) \tag{24}$$

$$\begin{aligned} \phi_{n\mathbf{k}}(i\omega_j) \;=\; & \frac{T}{N_\mathrm{F}} \frac{1}{N_\mathrm{q}} \sum_{\mathbf{q}m}^{\mathrm{fsthick}} \sum_{j'} \frac{\phi_{m\mathbf{k}+\mathbf{q}}(i\omega_{j'})}{\theta_{m\mathbf{k}+\mathbf{q}}(i\omega_{j'})} \lambda_{n\mathbf{k},m\mathbf{k}+\mathbf{q}}(\omega_j - \omega_{j'}) \\ & - \frac{T}{N_\mathrm{F}} \frac{1}{N_{\mathbf{k}_\mathrm{c}}} \left[\sum_{\mathbf{k}_\mathrm{c}m}^{\mathrm{fsthick}} \sum_{j'} \frac{\phi_{m\mathbf{k}_\mathrm{c}}(i\omega_{j'})}{\theta_{m\mathbf{k}_\mathrm{c}}(i\omega_{j'})} \mu_\mathrm{c} + \sum_{\mathbf{k}_\mathrm{c}m}^{\mathrm{Outer.}} \sum_{j'} \frac{\phi^{(\mathrm{out})}_{m\mathbf{k}_\mathrm{c}}(i\omega_{j'})}{\theta^{(\mathrm{out})}_{m\mathbf{k}_\mathrm{c}}(i\omega_{j'})} \mu_\mathrm{c} \right] \end{aligned} \tag{25}$$

$$\phi^{(\mathrm{out})}_{n\mathbf{k}}(i\omega_j) = - \frac{T}{N_\mathrm{F}} \frac{1}{N_{\mathbf{k}_\mathrm{c}}} \left[\sum_{\mathbf{k}_\mathrm{c}m}^{\mathrm{fsthick}} \sum_{j'} \frac{\phi_{m\mathbf{k}_\mathrm{c}}(i\omega_{j'})}{\theta_{m\mathbf{k}_\mathrm{c}}(i\omega_{j'})} \mu_\mathrm{c} + \sum_{\mathbf{k}_\mathrm{c}m}^{\mathrm{Outer.}} \sum_{j'} \frac{\phi^{(\mathrm{out})}_{m\mathbf{k}_\mathrm{c}}(i\omega_{j'})}{\theta^{(\mathrm{out})}_{m\mathbf{k}_\mathrm{c}}(i\omega_{j'})} \mu_\mathrm{c} \right] \tag{26}$$

where $\sum_{\mathbf{k}_\mathrm{c}m}^{\mathrm{fsthick}}$ and $\sum_{\mathbf{k}_\mathrm{c}m}^{\mathrm{Outer.}}$ denote the summations over the final states $(\mathbf{k}_\mathrm{c}, n')$ lying within and outside the fsthick window, respectively.

The data of eigenenergies for 20×20×20 $\mathbf{k}_\mathrm{c}$-point grids is read from the file (filnscf_coul).

```
Start reading nscf file for Coulomb

Finish reading nscf file for Coulomb

k-grid read from ../outerbands/nb.bands.20x20x20.dat :   20  20  20
Nr irreducible k-points read from ../outerbands/nb.bands.20x20x20.dat :       256
Minimum eigenvalue of bands taken from the file (eV) =    -3.6519037290E+01
Maximum eigenvalue of bands taken from the file (eV) =     6.5821232648E+01
emin_coulomb + "Fermi level" (eV)                    =     2.8752569141E+00
emax_coulomb + "Fermi level" (eV)                    =     3.2875256914E+01
Only states taken from nscf file between       2.875257 eV and      32.875257 eV
will be included in the Eliashberg calculations.
      9 bands in the interval [emin_coulomb + "Fermi level", emax_coulomb + "Fermi level"]
```

Note 1: EPW calculations with icoulomb = 1 require that the fine $\mathbf{k}$ and $\mathbf{k}_\mathrm{c}$ grids are commensurate. In this tutorial, since nkf1, nkf2, and nkf3 are set to 20, the grid in nscf_outerbands.in must be on a 20x20x20 grid. You can use grids like 20x20x20, 10x10x10, 5x5x5, 4x4x4, etc., in nscf_outerbands.in.

Note 2: You must set nbnd in pw.x so that the eigenenergies within the energy range [emin_coulomb + "Fermi level", emax_coulomb + "Fermi level"] are correctly computed. If you change the window size, you need to check the eigenvalues output in nscf_outerbands.out.

▶ 10th step: Plot the leading edge of the superconducting gap as a function of temperature and compare it with the previous one.

You should get the following graph by plotting the data from all `nb.imag_aniso_gap0_XX` files. Use the `gnuplot` script `fig16.plt`.

```
$ gnuplot fig16.plt
$ evince fig16.pdf
```

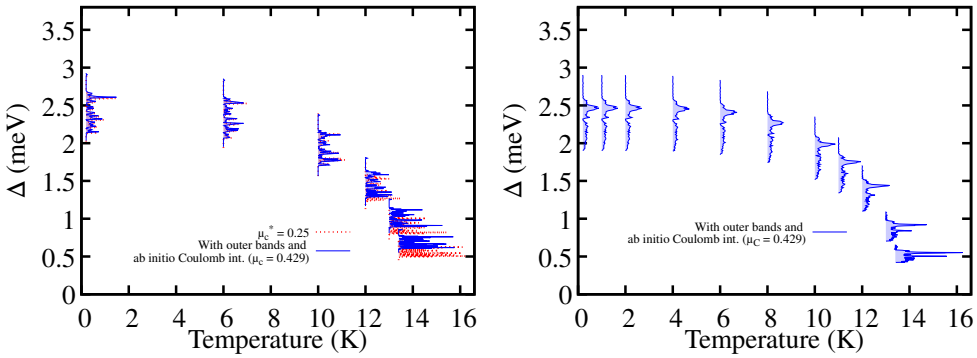

Fig. 16 Calculated anisotropic superconducting gap of Nb on the Fermi surface as a function of temperature. At convergence you should get something closer to the right hand-side figure (see arXiv: 2404.11528 for fully converged calculation parameters).

Restart options:

1. Restart by reading `prefix.epmatwp` and `epwdata.fmt`.

   This is useful when changing the following input values: `nkf1`, `nkf2`, `nkf3`, `nkq1`, `nkq2`, `nkq3`, `fsthick`, and `degaussw`.

   If the message "`Writing Hamiltonian, Dynamical matrix and EP vertex in Wann rep to file`" has already been output in the previous calculation, the electron-phonon interaction in the Wannier representation output to `prefix.epmatwp` can be read in subsequent calculations. **The `prefix.ephmat`, `restart.fmt`, `selecq.fmt`, and `prefix.a2f` output in the previous calculation need to be removed from the working directory.**

   Required files: `prefix.epmatwp`, `prefix.ukk`, `crystal.fmt`, `epwdata.fmt`, `vmedata.fmt` (or `dmedata.fmt`).

   Input setup:

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

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

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

   The number of pools (`npool`) can be changed from that used for writing `prefix.epmatwp`.

2. 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.         ! If .false., read *.ukk file
ephwrite    = .true.
```

   This requires to use the same number of pools (`npool`) as in the original run.

3. Restart by reading `ephmatXX` files in `prefix.ephmat`.

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

   Input setup:

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

epwwrite    = .false.
epwread     = .true.

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

   From version 5.8, `ephmatXX` files can be read with a different `npool` than used for writing.