ICTP/Psi-k/CECAM School on Electron-Phonon Physics from First Principles

Trieste, 19-23 March 2018

Lecture Fri.3

Superconducting gap and critical temperature using EPW

Roxana Margine

Department of Physics, Applied Physics, and Astronomy

Binghamton University - State University of New York

Lecture Summary

- Structure of the code
- Technicalities and convergences parameters

Migdal-Eliashberg Equations on the Imaginary Axis

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

mass renormalization function

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

superconducting gap function

anisotropic e-ph coupling strength

Migdal-Eliashberg Equations on the Imaginary Axis

Input variables:

```
ephwrite        = .true.
```

CALL write_ephmat(...)

$$\boxed{\lambda_{n\mathbf{k},m\mathbf{k}+\mathbf{q}}(\omega_j)} = N_{\mathrm{F}} \sum_{\nu} \int_0^{\infty} d\omega \, \frac{2\omega}{\omega_j^2 + \omega^2} \underset{\uparrow}{\boxed{|g_{mn\nu}(\mathbf{k},\mathbf{q})|^2}} \delta(\omega - \omega_{\mathbf{q}\nu})$$

e-ph matrix elements
on fine $\mathbf{k}$ and $\mathbf{q}$ grids
(prefix.ephmatX files)

Structure of the Code

The epw.f90 file:

```
CALL elphon_shuffle_wrap()
--> CALL ephwann_shuffle ( nqc , xqc )
```

The ephwann_shuffle.f90 file:

```
IF (ephwrite) THEN
   IF ( iq .eq. 1 ) THEN
      ! Find (irreducible) k-points on the fine grid within the Fermi window and write prefix.ikmap file
      CALL kmesh_fine
      ! Compute the index of k+q on the fine k-grid
      CALL kqmap_fine
   ENDIF
   ! Write prefix.ephmatX, prefix.freq, and prefix.egnv files
   CALL write_ephmat( iq )
ENDIF
```

Structure of the Code

Input variables:

```
ephwrite  = .true.       fsthick  = 0.4 ! eV
                         degaussw = 0.1 ! eV
mp_mesh_k = .true. ! use irreducible k-points
nkf1 = 20                nqf1 = 20
nkf2 = 20                nqf2 = 20
nkf3 = 20                nqf3 = 20
```

The fine $\mathbf{k}$ and $\mathbf{q}$ grids are required to be uniform and commensurate such that the $\mathbf{k}' = \mathbf{k} + \mathbf{q}$ grid maps into the $\mathbf{k}$ grid.

Files created (used for solving the Migdal-Eliashberg equations):

```
prefix.ephmatX    ! e-ph matrix elements within the Fermi window for the fine k and q grids (X=#files=#processors)
prefix.freq       ! Phonon frequencies on the q fine grid
prefix.egnv       ! Eigenvalues within the Fermi window on the fine k grid
prefix.ikmap      ! Index of each k-point on the uniform grid on the corresponding irreducible grid
```

Electron-phonon Coupling Strength

Input variables:

```
eliashberg     = .true.
```

`CALL lambdar_aniso_ver1(...)`

$$\boxed{\lambda_{n\mathbf{k},m\mathbf{k}+\mathbf{q}}(\omega_j)} = N_{\mathrm{F}} \sum_{\nu} \int_0^{\infty} d\omega \frac{2\omega}{\omega_j^2 + \omega^2} \boxed{|g_{mn\nu}(\mathbf{k},\mathbf{q})|^2} \delta(\omega - \omega_{\mathbf{q}\nu})$$

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

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

Structure of the Code

The epw.f90 file:

```fortran
IF ( eliashberg ) THEN
   CALL eliashberg_eqs()
ENDIF
```

The eliashberg.f90 file:

```fortran
  IF ( .not. liso .AND. .not. laniso ) THEN
     CALL read_frequencies  ! Read prefix.freq
     CALL read_eigenvalues  ! Read prefix.egnv
     CALL read_kqmap        ! Read prefix.ikmap and compute the index of k+q on the fine k-grid
     CALL read_ephmat       ! Read prefix.ephmatX
     !
     CALL eliashberg_init
     CALL evaluate_a2f_lambda
     --> CALL lambdar_aniso_ver1(...)
     CALL estimate_tc_gap
  ENDIF
```

Structure of the Code

Files created:

```
prefix.a2f ! Eliashberg spectral function as a function of
    frequency (meV) for various smearings
prefix.a2f_iso ! 2nd column is the Eliashberg spectral
    function corresponding to the first smearing in .a2f.
    Remaining columns are the mode-resolved Eliashberg
    spectral function (there is no specific information on
    which modes correspond to which atomic species).
prefix.lambda_k_pairs ! \lambda_nk distribution on FS
prefix.lambda_FS ! k-point Cartesian coords, n, E_nk-E_F[eV],
     \lambda_nk
prefix.phdos ! Phonon DOS (same as .a2f)
prefix.phdos_proj ! Phonon DOS (same as .a2f_iso)
```

Structure of the Code

Files created:

```
prefix.a2f ! Eliashberg spectral function as a function of frequency (meV) for various smearings
prefix.a2f_iso ! 2nd column is the Eliashberg spectral function corresponding to the first smearing in .a2f. Remaining columns are the mode-resolved Eliashberg spectral function (there is no specific information on which modes correspond to which atomic species).
prefix.lambda_k_pairs ! \lambda_nk distribution on FS
prefix.lambda_FS ! k-point Cartesian coords, n, E_nk-E_F[eV], \lambda_nk
prefix.phdos ! Phonon DOS (same as .a2f)
prefix.phdos_proj ! Phonon DOS (same as .a2f_iso)
```

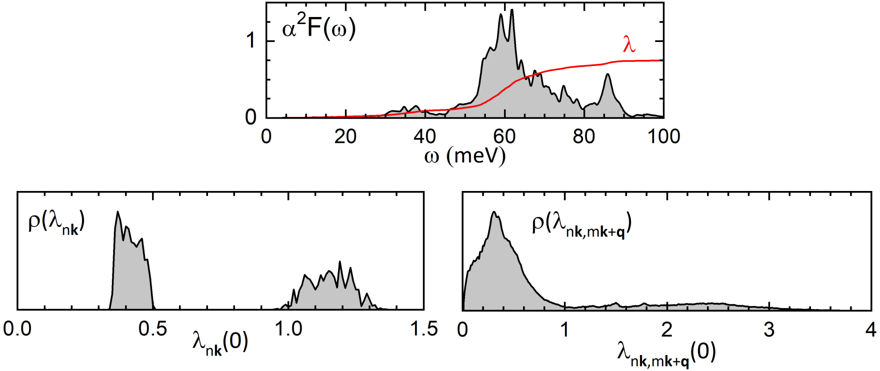
Files created with iverbosity = 2:

```
prefix.lambda_aniso ! E_nk-E_F[eV], \lambda_nk, k, n
prefix.lambda_pairs ! \lambda_nk,mk+q distribution on FS
prefix.lambda_YY.cube ! Same as prefix.lambda_FS for VESTA visualization. YY is the band index within the energy window
```

Electron-Phonon Coupling Strength

Files created (eliashberg = .true.):

```
prefix.a2f
prefix.lambda_k_pairs
prefix.lambda_pairs ! (iverbosity = 2)
```



Figures adapted from Margine and F. Giustino, Phys. Rev. B 87, 024505 (2013)

Migdal-Eliashberg Equations on the Imaginary Axis

Input variables:

```
eliashberg        = .true.
laniso            = .true.
limag             = .true.
```

CALL eliashberg_aniso_iaxis

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

mass renormalization function

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

superconducting gap function

CALL lambdar_aniso_ver1(...)

Structure of the Code

The epw.f90 file:

```fortran
IF ( eliashberg ) THEN
   CALL eliashberg_eqs()
ENDIF
```

The eliashberg.f90 file:

```fortran
  IF ( laniso ) THEN
     CALL read_frequencies   ! Read prefix.freq
     CALL read_eigenvalues   ! Read prefix.egnv
     CALL read_kqmap         ! Read prefix.ikmap and compute the index of k+q on the fine k-grid
     CALL read_ephmat        ! Read prefix.ephmatX

     CALL eliashberg_init
     CALL evaluate_a2f_lambda
     CALL estimate_tc_gap
     IF ( limag ) CALL eliashberg_aniso_iaxis
  ENDIF
```

Structure of the Code

The `eliashberg_aniso_iaxis.f90` file:

```
DO itemp = 1, nstemp ! loop over temperature
   ! Generate the frequency grid on the imaginary axis
   CALL gen_freqgrid_iaxis( itemp )
   IF ( ( limag .AND. .not. imag_read ) .OR. ( limag .AND.
       imag_read .AND. itemp .ne. 1 ) ) THEN
      iter = 1
      conv = .false.
      DO WHILE ( .not. conv .AND. iter .le. nsiter )
         ! Solve Migdal-Eliashberg eqs on the imaginary axis
         CALL sum_eliashberg_aniso_iaxis( itemp, iter, conv )
         --> CALL kernel_aniso_iaxis( itemp )
             --> CALL lambdar_aniso_ver1 (...)
         iter = iter + 1
      ENDDO ! iter
      IF ( conv ) CALL free_energy( itemp )
   ELSEIF ( limag .AND. imag_read .AND. itemp .eq. 1 ) THEN
      ! Read from file Delta and Znorm on the imaginary axis
      CALL eliashberg_read_aniso_iaxis( itemp )
   ENDIF
ENDDO
```

Structure of the Code

The `eliashberg_aniso_iaxis.f90` file:

```fortran
DO itemp = 1, nstemp ! loop over temperature
   ! Generate the frequency grid on the imaginary axis
   CALL gen_freqgrid_iaxis( itemp )
   IF ( ( limag .AND. .not. imag_read ) .OR. ( limag .AND.
       imag_read .AND. itemp .ne. 1 ) ) THEN
      ! Solve Migdal-Eliashberg eqs on the imaginary axis
      ....
      ....
   ELSEIF ( limag .AND. imag_read .AND. itemp .eq. 1 ) THEN
      ! Read from file Delta and Znorm on the imaginary axis
      CALL eliashberg_read_aniso_iaxis( itemp ) <--
   ENDIF
ENDDO
```

Restart option:

- `imag_read = .true.`

Structure of the Code

Input variables:

```
eliashberg   = .true.       conv_thr_iaxis = 1.0d-4
limag        = .true.       nsiter  = 500
laniso       = .true.       wscut   = 1.0 ! eV cutoff freq.

nstemp       = 4            muc     = 0.16 ! Coulomb parameter
tempsmin     = 15.0
tempsmax     = 60.0
```

Structure of the Code

Input variables:

```
eliashberg   = .true.          conv_thr_iaxis = 1.0d-4
limag        = .true.          nsiter = 500
laniso       = .true.          wscut  = 1.0 ! eV cutoff freq.

nstemp       = 4               muc    = 0.16 ! Coulomb parameter
tempsmin     = 15.0
tempsmax     = 60.0
```

Input variables:

```
eliashberg   = .true.          conv_thr_iaxis = 1.0d-4
limag        = .true.          nsiter = 500
laniso       = .true.          wscut  = 1.0 ! eV cutoff freq.

temps(1)     = 15.0            muc    = 0.16 ! Coulomb parameter
temps(2)     = 30.0
temps(3)     = 45.0
temps(4)     = 60.0
```

Structure of the Code

Files created (XX indicates the temperature):

```
prefix.imag_aniso_XX           ! w_j[eV], E_nk-E_F[eV], Z_nk,
    \Delta_nk[eV], Z^N_nk
prefix.imag_aniso_gap0_XX      ! \Delta_nk(0)[meV]
    distribution on FS
prefix.imag_aniso_gap_FS_XX    ! k-point Cartesian coords,
    band index within energy window, E_nk-E_F[eV],
    \Delta_nk(0)[eV]
```

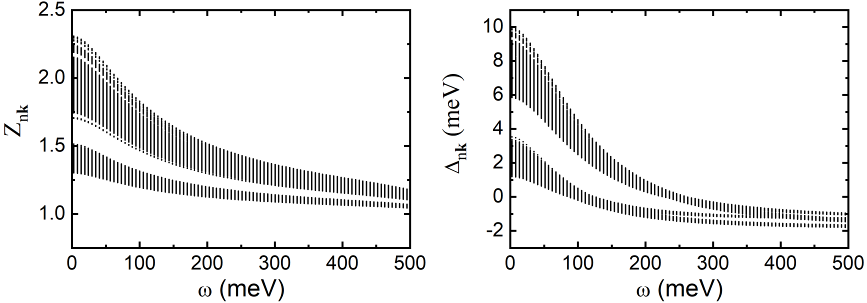
Files created with iverbosity = 2 (YY is the band index within the energy window):

```
prefix.imag_aniso_gap0_XX_YY.cube ! Same as prefix.
    imag_aniso_gap_FS_XX for VESTA visualization
```

Migdal-Eliashberg Equations on the Imaginary Axis

Files created:

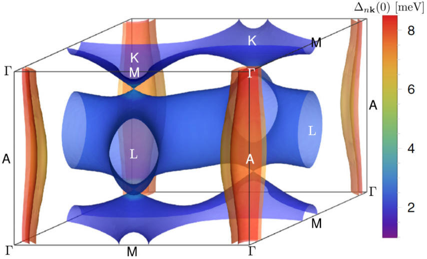
```
prefix.imag_aniso_XX
```

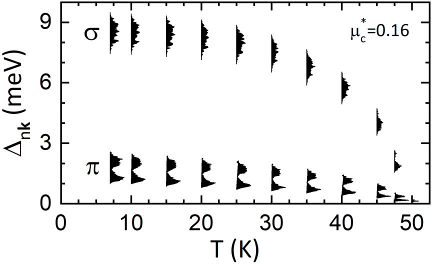


Superconducting Gap and Critical Temperature

Files created:

```
prefix.imag_aniso_gap0_XX_YY.cube   ! uniform k-grid, \Delta_nk(0)[eV]
prefix.imag_aniso_gap0_XX           ! \Delta_nk(0)[eV] distribution on FS
```





Left and right figures from Poncé et al, Comp. Phys. Commun. 209, 116 (2016) and Margine and Giustino, Phys. Rev. B 87, 024505 (2013)

Superconducting Gap Convergence

Convergence must be made for:

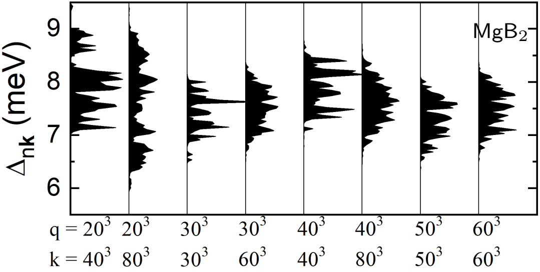
```
nkf1, nkf2, nkf3         wscut
nqf1, nqf2, nqf3         fsthick
```

Left figure from Margine and Giustino, Phys. Rev. B 87, 024505 (2013)

Superconducting Gap Convergence

Convergence must be made for:

```
nkf1, nkf2, nkf3          wscut
nqf1, nqf2, nqf3          fsthick
```

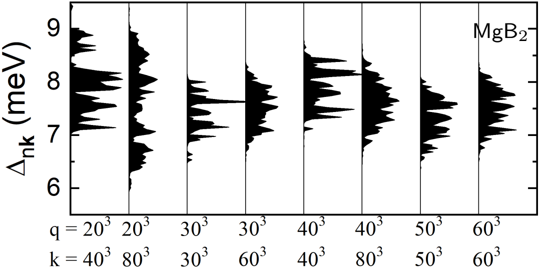


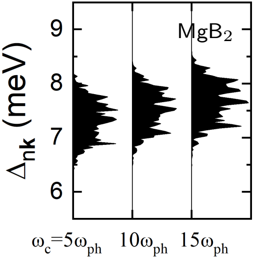
Left figure from Margine and Giustino, Phys. Rev. B 87, 024505 (2013)

Superconducting Gap Convergence

Convergence must be made for:

```
nkf1, nkf2, nkf3          wscut
nqf1, nqf2, nqf3          fsthick
```



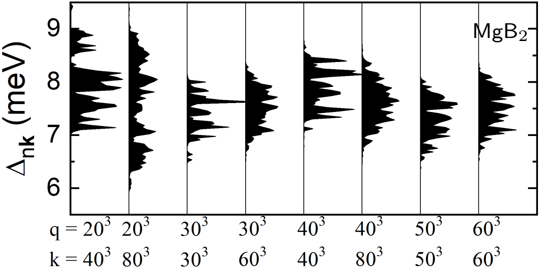


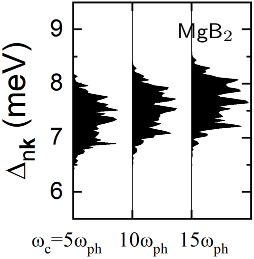
Left figure from Margine and Giustino, Phys. Rev. B 87, 024505 (2013)

Superconducting Gap Convergence

Convergence must be made for:

```
nkf1, nkf2, nkf3          wscut
nqf1, nqf2, nqf3          fsthick
```

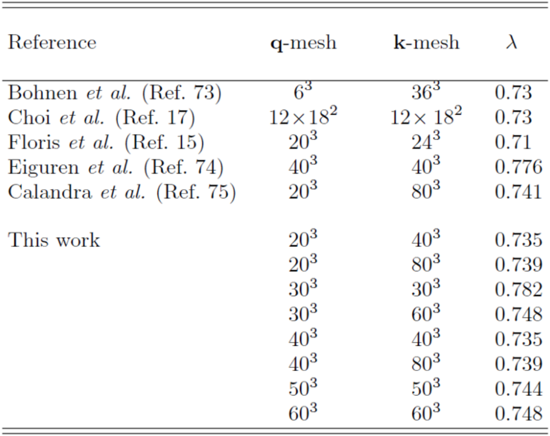




Description of anisotropic quantities requires very dense $\mathbf{k}$ and $\mathbf{q}$ grids

Left figure from Margine and Giustino, Phys. Rev. B 87, 024505 (2013)

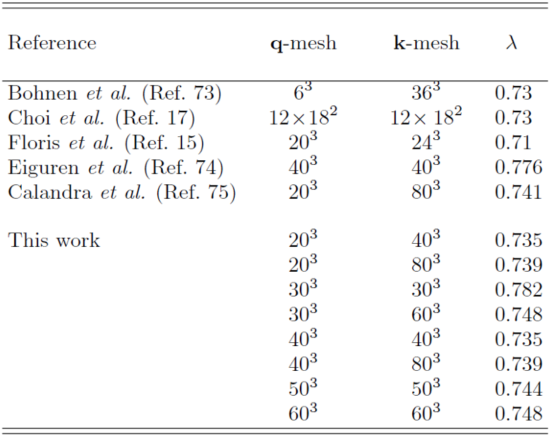
Electron-Phonon Coupling Strength Convergence



Reference	**q**-mesh	**k**-mesh	λ
Bohnen *et al.* (Ref. 73)	6^3	36^3	0.73
Choi *et al.* (Ref. 17)	12×18^2	12×18^2	0.73
Floris *et al.* (Ref. 15)	20^3	24^3	0.71
Eiguren *et al.* (Ref. 74)	40^3	40^3	0.776
Calandra *et al.* (Ref. 75)	20^3	80^3	0.741
This work	20^3	40^3	0.735
	20^3	80^3	0.739
	30^3	30^3	0.782
	30^3	60^3	0.748
	40^3	40^3	0.735
	40^3	80^3	0.739
	50^3	50^3	0.744
	60^3	60^3	0.748

Table from Margine and Giustino, Phys. Rev. B 87, 024505 (2013)

Electron-Phonon Coupling Strength Convergence



Reference	**q**-mesh	**k**-mesh	λ
Bohnen *et al.* (Ref. 73)	6^3	36^3	0.73
Choi *et al.* (Ref. 17)	12×18^2	12×18^2	0.73
Floris *et al.* (Ref. 15)	20^3	24^3	0.71
Eiguren *et al.* (Ref. 74)	40^3	40^3	0.776
Calandra *et al.* (Ref. 75)	20^3	80^3	0.741
This work	20^3	40^3	0.735
	20^3	80^3	0.739
	30^3	30^3	0.782
	30^3	60^3	0.748
	40^3	40^3	0.735
	40^3	80^3	0.739
	50^3	50^3	0.744
	60^3	60^3	0.748

Isotropic quantities are less sensitive to the size of $\mathbf{k}$ and $\mathbf{q}$ grids

Table from Margine and Giustino, Phys. Rev. B 87, 024505 (2013)

Superconducting Specific Heat

- Free energy difference between superconducting and normal states

$$\Delta F = -\pi T \sum_{nj} \int \frac{d\mathbf{k}}{\Omega_{\mathrm{BZ}}} \left[\sqrt{\omega_j^2 + \Delta_{n\mathbf{k}}^2(i\omega_j)} - |\omega_j| \right]$$
$$\times \left[Z_{n\mathbf{k}}(i\omega_j) - Z_{n\mathbf{k}}^N(i\omega_j)|\omega_j| / \sqrt{\omega_j^2 + \Delta_{n\mathbf{k}}^2(i\omega_j)} \right] \delta(\epsilon_{n\mathbf{k}} - \epsilon_{\mathrm{F}})$$

Figure from Poncé et al, Comp. Phys. Commun. 209, 116 (2016)

Superconducting Specific Heat

- Free energy difference between superconducting and normal states

$$\Delta F \;=\; -\pi T \sum_{nj} \int \frac{d\mathbf{k}}{\Omega_{\rm BZ}} \left[\sqrt{\omega_j^2 + \Delta_{n\mathbf{k}}^2(i\omega_j)} - |\omega_j|\right]$$
$$\times \left[Z_{n\mathbf{k}}(i\omega_j) - Z_{n\mathbf{k}}^N(i\omega_j)|\omega_j|/\sqrt{\omega_j^2 + \Delta_{n\mathbf{k}}^2(i\omega_j)}\right] \delta(\epsilon_{n\mathbf{k}} - \epsilon_{\rm F})$$

$$\Delta C = -T\frac{d^2\Delta F}{dT^2}$$

↑

superconducting
specific heat

Figure from Poncé et al, Comp. Phys. Commun. 209, 116 (2016)

Superconducting Specific Heat

- Free energy difference between superconducting and normal states

$$\Delta F = -\pi T \sum_{nj} \int \frac{d\mathbf{k}}{\Omega_{\rm BZ}} \left[\sqrt{\omega_j^2 + \Delta_{n\mathbf{k}}^2(i\omega_j)} - |\omega_j| \right] \times \left[Z_{n\mathbf{k}}(i\omega_j) - Z^N_{n\mathbf{k}}(i\omega_j)|\omega_j| / \sqrt{\omega_j^2 + \Delta_{n\mathbf{k}}^2(i\omega_j)} \right] \delta(\epsilon_{n\mathbf{k}} - \epsilon_{\rm F})$$

$$\Delta C = -T \frac{d^2 \Delta F}{dT^2}$$

↑

superconducting specific heat

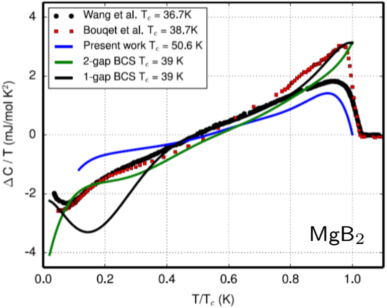


Figure from Poncé et al, Comp. Phys. Commun. 209, 116 (2016)

Structure of the Code

The `eliashberg_aniso_iaxis.f90` file:

```fortran
DO itemp = 1, nstemp ! loop over temperature
   CALL gen_freqgrid_iaxis( itemp )
   IF ( ( limag .AND. .not. imag_read ) .OR. ( limag .AND. imag_read .AND. itemp .ne. 1 ) ) THEN
      iter = 1
      conv = .false.
      DO WHILE ( .not. conv .AND. iter .le. nsiter )
         CALL sum_eliashberg_aniso_iaxis( itemp, iter, conv )
         iter = iter + 1
      ENDDO ! iter
      IF ( conv ) CALL free_energy( itemp ) <--
   ELSEIF ( limag .AND. imag_read .AND. itemp .eq. 1 ) THEN
      CALL eliashberg_read_aniso_iaxis( itemp )
   ENDIF
ENDDO
```

Files created :

```
prefix.fe_XX ! temperature, free energy difference between superconducting and normal state
```

Migdal-Eliashberg Equations on Real Axis

- The Migdal-Eliasberg equations on the imaginary frequency axis are computationally efficient (only involve sums over a finite number of Matsubara frequencies) and they are adequate for calculating the critical temperature and the superconducting gap.

Migdal-Eliashberg Equations on Real Axis

- The Migdal-Eliasberg equations on the imaginary frequency axis are computationally efficient (only involve sums over a finite number of Matsubara frequencies) and they are adequate for calculating the critical temperature and the superconducting gap.
- To extract information about the spectral properties (e.g., the quasi particle density of states or the single particle excitation spectrum), we need to solve the Migdal-Eliashberg equations on the real energy axis.

Migdal-Eliashberg Equations on Real Axis

- The Migdal-Eliasberg equations on the imaginary frequency axis are computationally efficient (only involve sums over a finite number of Matsubara frequencies) and they are adequate for calculating the critical temperature and the superconducting gap.
- To extract information about the spectral properties (e.g., the quasi particle density of states or the single particle excitation spectrum), we need to solve the Migdal-Eliashberg equations on the real energy axis.
- Direct evaluation of the Migdal-Eliashberg equations on the real energy axis is in principle possible but very demanding computationally since it involves the evaluation of many principal value integrals.

Migdal-Eliashberg Equations on Real Axis

- The Migdal-Eliasberg equations on the imaginary frequency axis are computationally efficient (only involve sums over a finite number of Matsubara frequencies) and they are adequate for calculating the critical temperature and the superconducting gap.
- To extract information about the spectral properties (e.g., the quasi particle density of states or the single particle excitation spectrum), we need to solve the Migdal-Eliashberg equations on the real energy axis.
- Direct evaluation of the Migdal-Eliashberg equations on the real energy axis is in principle possible but very demanding computationally since it involves the evaluation of many principal value integrals.
- As an altenative, the solutions on the real energy axis can be obtained by analytic continuation of the solutions along the imaginary frequency axis. The analytic continuation can be performed by using Padé approximants (very light computationally) or by means of an iterative procedure (very heavy computationally).

Structure of the Code

The `eliashberg_aniso_iaxis.f90` file:

```fortran
DO itemp = 1, nstemp ! loop over temperature
   ...
   ...
   IF ( lpade ) THEN
      conv = .false.
      CALL pade_cont_aniso_iaxis_to_raxis( itemp, N, conv )
      IF ( conv ) CALL dos_quasiparticle( itemp )
   ENDIF
   IF ( lacon ) THEN
      iter = 1
      conv = .false.
      DO WHILE ( .not. conv .AND. iter .le. nsiter )
         CALL analytic_cont_aniso_iaxis_to_raxis( itemp, iter, conv )
           iter = iter + 1
        ENDDO ! iter
      IF ( conv ) CALL dos_quasiparticle( itemp )
   ENDIF
ENDDO
```

Structure of the Code

Input variables:

```
eliashberg   = .true.        conv_thr_iaxis = 1.0d-4
limag        = .true.        nsiter = 500
laniso       = .true.        wscut   = 1.0 ! eV cutoff freq.
lpade        = .true.
lacon        = .true.        conv_thr_racon = 1.0d-4

nstemp       = 4             muc     = 0.16 ! Coulomb parameter
tempsmin     = 15.0
tempsmax     = 60.0
```

Structure of the Code

Input variables:

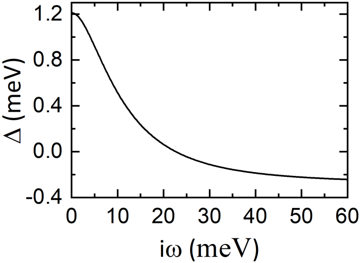
```
eliashberg   = .true.        conv_thr_iaxis = 1.0d-4
limag        = .true.        nsiter = 500
laniso       = .true.        wscut  = 1.0 ! eV cutoff freq.
lpade        = .true.
lacon        = .true.        conv_thr_racon = 1.0d-4

nstemp       = 4             muc    = 0.16 ! Coulomb parameter
tempsmin     = 15.0
tempsmax     = 60.0
```

Files created (XX indicates the temperature):

```
prefix.pade_aniso_XX ! w[eV], E_nk-E_F[eV], RE[Z_nk], Im[Z_nk
    ], Re[\Delta_nk][eV], Im[\Delta_nk][eV] (iverbosity=2)
prefix.pade_aniso_gap0_XX ! Re[\Delta_nk(0)][eV] distribution
     on FS
prefix.acon_aniso_XX ! w[eV], E_nk-E_F[eV], RE[Z_nk], Im[Z_nk
    ], Re[\Delta_nk][eV], Im[\Delta_nk][eV] (iverbosity=2)
prefix.acon_aniso_gap0_XX ! Re[\Delta_nk(0)][eV] distribution
     on FS
```

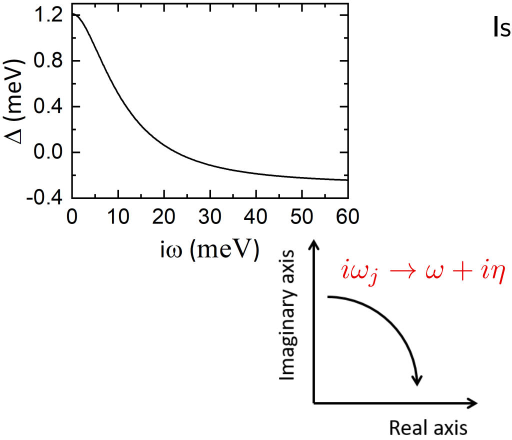
Analytic Continuation to Real Axis



Isotropic case in Pb

Figures from Margine and Giustino, Phys. Rev. B 87, 024505 (2013)

Analytic Continuation to Real Axis



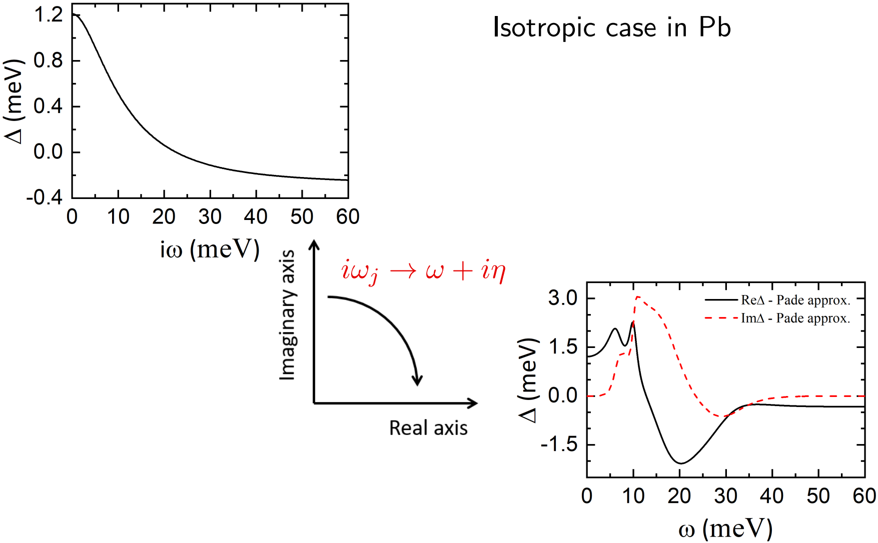
$i\omega_j \rightarrow \omega + i\eta$

Isotropic case in Pb

Figures from Margine and Giustino, Phys. Rev. B 87, 024505 (2013)

Analytic Continuation to Real Axis

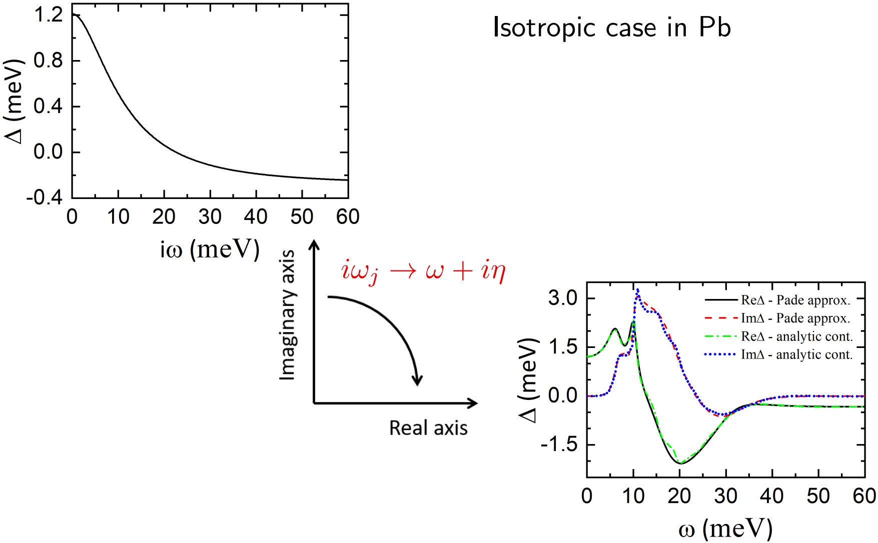
Isotropic case in Pb



Figures from Margine and Giustino, Phys. Rev. B 87, 024505 (2013)

Analytic Continuation to Real Axis

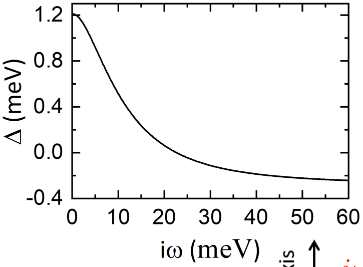
Isotropic case in Pb



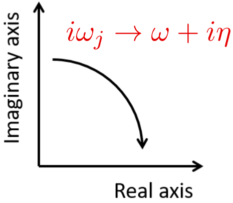
$$i\omega_j \to \omega + i\eta$$

Figures from Margine and Giustino, Phys. Rev. B 87, 024505 (2013)

Analytic Continuation to Real Axis

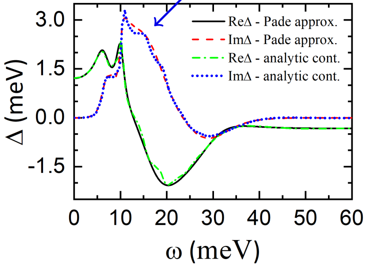


Isotropic case in Pb



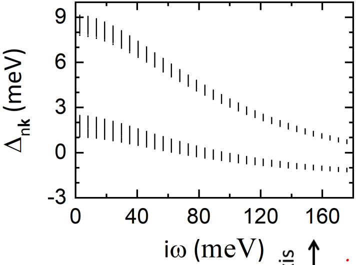
$i\omega_j \rightarrow \omega + i\eta$

structure in the real axis solutions on the scale of the phonon energy

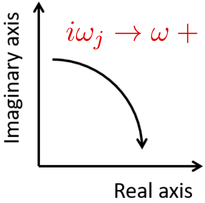


Figures from Margine and Giustino, Phys. Rev. B 87, 024505 (2013)

Analytic Continuation to Real Axis in MgB_2

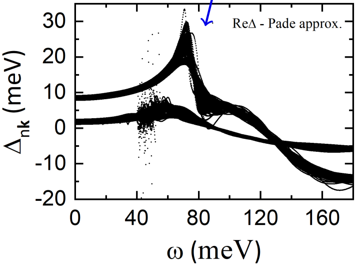


Anisotropic case in MgB_2



$i\omega_j \to \omega + i\eta$

structure in the real axis solutions on the scale of the phonon energy



Figures from Margine and Giustino, Phys. Rev. B 87, 024505 (2013)

Excitation Spectrum of a Superconductor

- The single-particle Green's function on real axis is given by:

$$\hat{G}_{n\mathbf{k}}(\omega) = \frac{\omega Z_{n\mathbf{k}}(\omega)\hat{\tau}_0 + (\epsilon_{n\mathbf{k}} - \epsilon_{\mathrm{F}})\hat{\tau}_3 + \Delta_{n\mathbf{k}}(\omega) Z_{n\mathbf{k}}(\omega)\hat{\tau}_1}{\left[\omega Z_{n\mathbf{k}}(\omega)\right]^2 - (\epsilon_{n\mathbf{k}} - \epsilon_{\mathrm{F}})^2 - \left[Z_{n\mathbf{k}}(\omega)\Delta_{n\mathbf{k}}(\omega)\right]^2}$$

$$\hat{\tau}_0 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \qquad \hat{\tau}_1 = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \qquad \hat{\tau}_3 = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

Excitation Spectrum of a Superconductor

- The single-particle Green's function on real axis is given by:

$$\hat{G}_{n\mathbf{k}}(\omega) = \frac{\omega Z_{n\mathbf{k}}(\omega)\hat{\tau}_0 + (\epsilon_{n\mathbf{k}} - \epsilon_{\mathrm{F}})\hat{\tau}_3 + \Delta_{n\mathbf{k}}(\omega) Z_{n\mathbf{k}}(\omega)\hat{\tau}_1}{\left[\omega Z_{n\mathbf{k}}(\omega)\right]^2 - (\epsilon_{n\mathbf{k}} - \epsilon_{\mathrm{F}})^2 - \left[Z_{n\mathbf{k}}(\omega)\Delta_{n\mathbf{k}}(\omega)\right]^2}$$

$$\hat{\tau}_0 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \qquad \hat{\tau}_1 = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \qquad \hat{\tau}_3 = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

- The poles of the diagonal components of $\hat{G}_{n\mathbf{k}}(\omega)$ give the elemental excitations of the superconductor

Excitation Spectrum of a Superconductor

- The single-particle Green's function on real axis is given by:

$$\hat{G}_{n\mathbf{k}}(\omega) = \frac{\omega Z_{n\mathbf{k}}(\omega)\hat{\tau}_0 + (\epsilon_{n\mathbf{k}} - \epsilon_{\mathrm{F}})\hat{\tau}_3 + \Delta_{n\mathbf{k}}(\omega) Z_{n\mathbf{k}}(\omega)\hat{\tau}_1}{\left[\omega Z_{n\mathbf{k}}(\omega)\right]^2 - (\epsilon_{n\mathbf{k}} - \epsilon_{\mathrm{F}})^2 - \left[Z_{n\mathbf{k}}(\omega)\Delta_{n\mathbf{k}}(\omega)\right]^2}$$

$$\hat{\tau}_0 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \qquad \hat{\tau}_1 = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \qquad \hat{\tau}_3 = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

- The poles of the diagonal components of $\hat{G}_{n\mathbf{k}}(\omega)$ give the elemental excitations of the superconductor

$$G^{11}_{n\mathbf{k}}(\omega) = \frac{\omega Z_{n\mathbf{k}}(\omega) + (\epsilon_{n\mathbf{k}} - \epsilon_{\mathrm{F}})}{\left[\omega Z_{n\mathbf{k}}(\omega)\right]^2 - (\epsilon_{n\mathbf{k}} - \epsilon_{\mathrm{F}})^2 - \left[Z_{n\mathbf{k}}(\omega)\Delta_{n\mathbf{k}}(\omega)\right]^2}$$

Excitation Spectrum of a Superconductor

- The single-particle Green's function on real axis is given by:

$$\hat{G}_{n\mathbf{k}}(\omega) = \frac{\omega Z_{n\mathbf{k}}(\omega)\hat{\tau}_0 + (\epsilon_{n\mathbf{k}} - \epsilon_{\mathrm{F}})\hat{\tau}_3 + \Delta_{n\mathbf{k}}(\omega) Z_{n\mathbf{k}}(\omega)\hat{\tau}_1}{\left[\omega Z_{n\mathbf{k}}(\omega)\right]^2 - (\epsilon_{n\mathbf{k}} - \epsilon_{\mathrm{F}})^2 - \left[Z_{n\mathbf{k}}(\omega)\Delta_{n\mathbf{k}}(\omega)\right]^2}$$

$$\hat{\tau}_0 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \qquad \hat{\tau}_1 = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \qquad \hat{\tau}_3 = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

- The poles of the diagonal components of $\hat{G}_{n\mathbf{k}}(\omega)$ give the elemental excitations of the superconductor

$$G^{11}_{n\mathbf{k}}(\omega) = \frac{\omega Z_{n\mathbf{k}}(\omega) + (\epsilon_{n\mathbf{k}} - \epsilon_{\mathrm{F}})}{\left[\omega Z_{n\mathbf{k}}(\omega)\right]^2 - (\epsilon_{n\mathbf{k}} - \epsilon_{\mathrm{F}})^2 - \left[Z_{n\mathbf{k}}(\omega)\Delta_{n\mathbf{k}}(\omega)\right]^2}$$

- The pole positions are: $E^2_{n\mathbf{k}} = \dfrac{(\epsilon_{n\mathbf{k}} - \epsilon_{\mathrm{F}})^2}{Z^2_{n\mathbf{k}}(E_{n\mathbf{k}})} + \Delta^2_{n\mathbf{k}}(E_{n\mathbf{k}})$

Superconducting Quasiparticle Energy and Lifetime

$$E_{n\mathbf{k}}^2 = \frac{(\epsilon_{n\mathbf{k}} - \epsilon_{\mathrm{F}})^2}{Z_{n\mathbf{k}}^2(E_{n\mathbf{k}})} + \Delta_{n\mathbf{k}}^2(E_{n\mathbf{k}})$$

Superconducting Quasiparticle Energy and Lifetime

$$E_{n\mathbf{k}}^2 = \frac{(\epsilon_{n\mathbf{k}} - \epsilon_{\mathrm{F}})^2}{Z_{n\mathbf{k}}^2(E_{n\mathbf{k}})} + \Delta_{n\mathbf{k}}^2(E_{n\mathbf{k}})$$

$\mathrm{Re}E_{n\mathbf{k}}$ quasiparticle energy renormalized by the superconducting pairing

Superconducting Quasiparticle Energy and Lifetime

$$E_{n\mathbf{k}}^2 = \frac{(\epsilon_{n\mathbf{k}} - \epsilon_{\mathrm{F}})^2}{Z_{n\mathbf{k}}^2(E_{n\mathbf{k}})} + \Delta_{n\mathbf{k}}^2(E_{n\mathbf{k}})$$

$\mathrm{Re}E_{n\mathbf{k}}$ quasiparticle energy renormalized by the superconducting pairing

$\mathrm{Im}E_{n\mathbf{k}}$ quasiparticle inverse lifetime (scattering time) due to the superconducting pairing

Superconducting Quasiparticle Energy and Lifetime

$$E_{n\mathbf{k}}^2 = \frac{(\epsilon_{n\mathbf{k}} - \epsilon_{\mathrm{F}})^2}{Z_{n\mathbf{k}}^2(E_{n\mathbf{k}})} + \Delta_{n\mathbf{k}}^2(E_{n\mathbf{k}})$$

$\mathrm{Re}E_{n\mathbf{k}}$ quasiparticle energy renormalized by the superconducting pairing

$\mathrm{Im}E_{n\mathbf{k}}$ quasiparticle inverse lifetime (scattering time) due to the superconducting pairing

At Fermi level $\epsilon_{n\mathbf{k}} = \epsilon_{\mathrm{F}}$ and the quasiparticle shift is $E_{n\mathbf{k}} = \mathrm{Re}\Delta_{n\mathbf{k}}(E_{n\mathbf{k}})$

This identity defines the leading edge $\Delta_{n\mathbf{k}}$ of the superconducting gap

$$\Delta_{n\mathbf{k}} = \mathrm{Re}\Delta_{n\mathbf{k}}(\Delta_{n\mathbf{k}})$$

Superconducting Quasiparticle Energy and Lifetime

$$E_{n\mathbf{k}}^2 = \frac{(\epsilon_{n\mathbf{k}} - \epsilon_{\mathrm{F}})^2}{Z_{n\mathbf{k}}^2(E_{n\mathbf{k}})} + \Delta_{n\mathbf{k}}^2(E_{n\mathbf{k}})$$

$\mathrm{Re}E_{n\mathbf{k}}$	quasiparticle energy renormalized by the superconducting pairing
$\mathrm{Im}E_{n\mathbf{k}}$	quasiparticle inverse lifetime (scattering time) due to the superconducting pairing

At Fermi level $\epsilon_{n\mathbf{k}} = \epsilon_{\mathrm{F}}$ and the quasiparticle shift is $E_{n\mathbf{k}} = \mathrm{Re}\Delta_{n\mathbf{k}}(E_{n\mathbf{k}})$

This identity defines the leading edge $\Delta_{n\mathbf{k}}$ of the superconducting gap

$$\Delta_{n\mathbf{k}} = \mathrm{Re}\Delta_{n\mathbf{k}}(\Delta_{n\mathbf{k}})$$

$\uparrow$

binding energy for electrons
in a Cooper pair

Superconducting Quasiparticle Density of States

- The superconducting quasiparticle density of states can be deduced from:

$$\frac{N_{n\mathbf{k},S}(\omega)}{N_{\mathrm{F}}} = -\frac{1}{\pi}\int_{-\infty}^{\infty} d\epsilon_{n\mathbf{k}} \mathrm{Im} G^{11}_{n\mathbf{k}}(\omega)$$

Superconducting Quasiparticle Density of States

- The superconducting quasiparticle density of states can be deduced from:

$$\frac{N_{n\mathbf{k},S}(\omega)}{N_{\mathrm{F}}} = -\frac{1}{\pi}\int_{-\infty}^{\infty} d\epsilon_{n\mathbf{k}} \mathrm{Im} G^{11}_{n\mathbf{k}}(\omega)$$

- In the BCS limit, $Z_{n\mathbf{k}} = 1$ and performing the integral leads to:

$$\frac{N_{n\mathbf{k},S}(\omega)}{N_{\mathrm{F}}} = \mathrm{Re}\left[\omega/\sqrt{\omega^2 - \Delta^2_{n\mathbf{k}}(\omega)}\right]$$

Superconducting Quasiparticle Density of States

- The superconducting quasiparticle density of states can be deduced from:

$$\frac{N_{n\mathbf{k},S}(\omega)}{N_{\mathrm{F}}} = -\frac{1}{\pi}\int_{-\infty}^{\infty} d\epsilon_{n\mathbf{k}} \mathrm{Im} G^{11}_{n\mathbf{k}}(\omega)$$

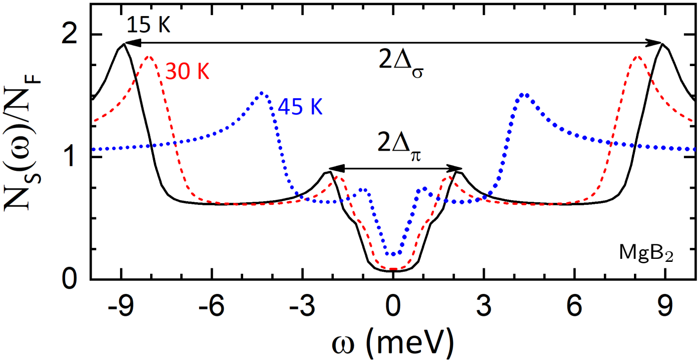
- In the BCS limit, $Z_{n\mathbf{k}} = 1$ and performing the integral leads to:

$$\frac{N_{n\mathbf{k},S}(\omega)}{N_{\mathrm{F}}} = \mathrm{Re}\left[\omega/\sqrt{\omega^2 - \Delta^2_{n\mathbf{k}}(\omega)}\right]$$

- Averaging over the Fermi surface leads to:

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

Superconducting Quasiparticle Density of States



Figures from Margine and Giustino, Phys. Rev. B 87, 024505 (2013)

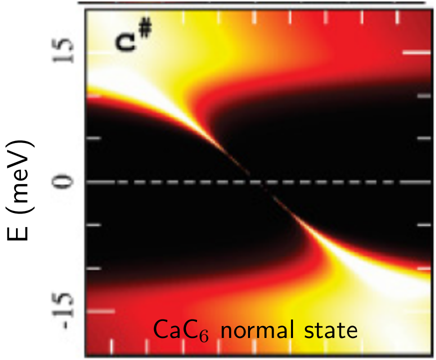
Spectral Function

$$A_{n\mathbf{k}}(\omega) = -\frac{1}{\pi}\mathrm{Im}G_{n\mathbf{k}}^{11}(\omega)$$

Figures from Sanna et al, Phys. Rev. B 85, 184514 (2012)

Spectral Function

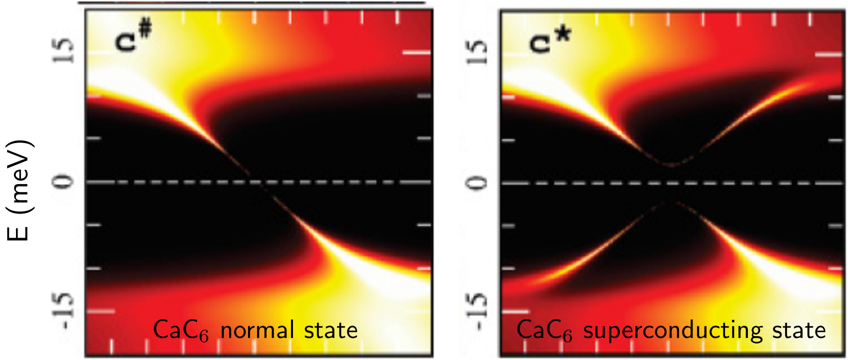
$$A_{n\mathbf{k}}(\omega) = -\frac{1}{\pi}\mathrm{Im}G^{11}_{n\mathbf{k}}(\omega)$$



Figures from Sanna et al, Phys. Rev. B 85, 184514 (2012)

Spectral Function

$$A_{n\mathbf{k}}(\omega) = -\frac{1}{\pi}\mathrm{Im}G^{11}_{n\mathbf{k}}(\omega)$$



CaC$_6$ normal state

CaC$_6$ superconducting state

Figures from Sanna et al, Phys. Rev. B 85, 184514 (2012)

Miscellaneous

- `wscut` is normally set to 4 to 10 times largest phonon energy
- `wscut` is ignored if the frequencies on the imaginary axis are given with `nswi`
- `laniso/liso` requires `eliashberg`
- `lpade` requires `limag`
- `lacon` requires `limag` and `lpade`
- T_c evaluated with Allen-Dynes formula can be used as a guide for defining the temperatures at which to evaluate the Migdal-Eliashberg eqs.
- `ephwrite` requires uniform fine $\mathbf{k}$ or $\mathbf{q}$ grids and `nkf1,nkf2,nkf3` to be multiple of `nqf1,nqf2,nqf3`
- `.ephmatXX`, `.egnv`, `.freq`, and `.ikmap` files need to be generated whenever $\mathbf{k}$ or $\mathbf{q}$ fine grid is changed

Miscellaneous

- `imag_read` requires `limag` and `laniso`
- `imag_read` allows the code to read from file the superconducting gap and renormalization function on the imaginary axis at specific temperature `XX` from file `.imag_aniso_XX`. The temperature is specified as `tempsmin = XX` or `temps(1) = XX`.
- `imag_read` can be used to:

  (1) solve the anisotropic Migdal-Eliashberg equations on the imaginary axis at temperatures greater than `XX` using as a starting point the superconducting gap estimated at temperature `XX`.

  (2) obtain the solutions of the Migdal-Eliashberg equations on the real axis with lpade or lacon starting from the imaginary axis solutions at temperature `XX`;

  (3) write to file the superconducting gap on the Fermi surface in cube format at temperature `XX` for `iverbosity = 2`.

References

- E. R. Margine and F. Giustino, Phys. Rev. B 87, 024505 (2013) [link]
- S. Poncé, E. R. Margine, C. Verdi, and F. Giustino, Comput. Phys. Commun. 209, 116 (2016) [link]
- D. J. Scalapino, J. R. Schrieffer, and J. W. Wilkins, Phys. Rev. 148, 263 (1966) [link]
- P. B. Allen, and B. Mitrović, Solid State Phys. 37, 1 (1982) [link]
- C. R. Leavens and D. S. Ritchie, Solid State Commun. 53, 137 (1985) [link]
- F. Marsiglio, M. Schossmann, and J. P. Carbotte, Phys. Rev. B 37, 4965 (1988) [link]

More info

EPW®

http://epw.org.uk

http://epwforum.uk

https://gitlab.com/QEF/q-e