ecalj details

https://github.com/tkotani/ecalj

November 17, 2024

Abstract

This explains details of ecalj on the based on Refs.[1] and [2] (These are contained at ecalj/-Document/Manual/ as Kotani2114QSGWinPMT.pdf and KotaniKinoAkai2015FormulationPMT.pdf) on top of developments [3].

Contents

1 Overview of PMT-QSGW algorithm

The `ecalj` package is based on the PMT method (=Linearized APW+MTO method) [1, 4, 5], which is a unique mixed basis method using two kinds of augmented waves simultaneously. With `ecalj`, we can do

- Total energy calculation and atomic-position relaxations within LDA, GGA and LDA/GGA+U. In addition, we can add spin-orbit coupling and so on (some limitations).
- QSGW calculations (quasiparticle self-consistent GW). Since it is on top of the PMT method, we call the QSGW method in ecalj as the PMT-QSGW method [2]. The calculation of self-energy can give impact ionization energy.
- The maximally localized Wannier functions. cRPA is implemented.
- We can calculate linear responses (dielectric and magetic).
- We use module-based coding in latest fortran (still on the way half). In principle, all the data are stored in modules, where data are genereted/read by subroutines in modules. Virtually, all the data are protected. When we read data in modules, we declear variables by use,only. Thus it is not so difficult to figure out where a data is modified or generated.

Original QSGW had implemented in the LMTO[6], while the LMTO-QSGW is very difficult to use. In contrast, PMT-QSGW is rather easier to use.

In addition, we added another developments to the original LMTO-QSGW; some ideas are from papers Ref.[7] by Friedrich, Blügel, and Schindlmayr, and Ref.[8] by Freysoldt et al. In this Sec. 1, we try to explain some details along the line of Ref.[2].

1.1 Crystal structure, notations, and common data in code

We use unit `alat` (measured by a.u.=0.528177Å) as the unit in the code. Thus, to convert quantities in the unit of a.u., we multiply `alat`Ḣere is some common notations in **ecalj** .

- Primitive cell vectors $\mathbf{p}_i$ (in a.u.) are `alat*plat(1:3,i)`, where i=1,2,3. For exampke, see `LATTC`, which appears in `work/si_gwsc` after install test (plat=PLAT given in ctrl file). $\mathbf{q}_i$=`qlat(1:3,i)` is reciprocal unit vectors such that `sum(plat(1:3,i),qlat(1:3,j))`=δ_{ij}.
- The centers of MT sites $\{\mathbf{R}\}$ in the primitive cell is given by $\{\mathbf{R}\}$ =`alat*bas(1:3,ibas),ibas=1,nbas` (we use `pos,natom` in cases instead of `bas,nbas`). $\{\mathbf{R}\}$ is the position vector measured from a center of primitive cell. `nbas` is number of atoms.
- Thus the MT sites are specified by $\mathbf{R}+\mathbf{T}$, where $\mathbf{T}$ specify centers of primitive cells. $\mathbf{T}(n1,n2,n3) = n_1\mathbf{q}_1 + n_2\mathbf{q}_2 + n_3\mathbf{q}_3$.
- In the followings, we use $\mathbf{k}$ and $\mathbf{q}$ (in cases, mixed up... sorry), both of which means vectors in the BZ. In procedures such as $\mathbf{k}+\mathbf{k}'$, we sometimes need to pull it back to the BZ (then $\mathbf{k}+\mathbf{k}'$ may be written as $\mathbf{k}+\mathbf{k}' = \mathbf{k}''+\mathbf{G}$, where $\mathbf{G}$ is a recprocal vector).
- We specify MTs (atoms) in the primitive cell in ctrl file, where the MTs with the same SPEC (species) can be further divided into some classes (CLASS). Thus `NBAS >= NSPEC >= NCLASS`. The numbering of MTs (`ibas=1,nbas`) are given by the order of SITE in ctrl file. We can use `lmchk` to check how they are divided into CLASS.
- `iclass(ibas)` is the id for class. The crystallographcally equivalent MT sites should have the same class id as `iclass(ibas1)=iclass(ibas2)`. However, for the convenience of program developments (historical reason), I assume `ibas=iclass(ibas)` for GW part of programs.

1.2 structure of ecalj

Make procedure generates fortran programs.

lmfa,lmf-MPIK,lmfgw-MPIK, lmf2gw, and lmchk are main programs for lmf part. Look into `main/lmv7.F`, which is the main programs of lmf-MPIK and lmfgw-MPIK. `main/lmv7.F` have a line `call M_lmfinit_init(prgnam)`. Then initial data are set by reading ctrl file. Except `v_ssite,v_sspec`, all data are protected.

Others are the GW-related and the Wannier-related programs (Wannier source codes are in wannier/ directory).

For GW, we run programs successively as given by scripts `gwsc` and so on. In `gwsc`, the stage up to `lmf2gw` is the preparation stage. After we run `lmf2gw`, we have files

```
BZDATA      EPSwklm   QGcou    QPLIST.lmfgw    efermi.lmf
CLASS       GWinput   QGpsi    lmfgw_kdivider  evec.*
CphiGeig    HAMindex  QOP      QIBZ            vxc.*
DATA4GW_V2  ZBAK      QBZ      QPLIST.IBZ
```

These are files required for performing GW calculations. We start GW part from `rdata4gw`, which just reformat these files. We do not read rst as well as ctrl in the GW part.

Duplicated informations are kept in `CLASS, DATA4GW_V2`, and `HAMindex`. (and something else, a little confusing). I think we need to clean up a little more.

Here is a list of variables for GW part (not everyting...).

```
alat: unit in a.u. This is in LATTC
plat: primitive vector. this is in LATTC (or call genallc_v3)
qlat: reciprocal primitive vector
      \delta_ij= sum(plat(:,i)*qlat(:,j))

QpGcut_psi: cutoff to determine G vector for eigenfunction
         |q+G|< QpGcut_cou (in a.u.)
         CAUTION: in code, we usually represent q and G in the unit of
         2pi/alat, thus the cutoff is (in the program)
         2*pi/alat*sum((q+G)**2))< QpGcut_cou**2

QpGcut_cou: cutoff to to determine G vector for Coulmb matrix
         |q+G|< QpGcut_psi

===
genalloc_v3: this allocate variables in m_genallc_v3. some variables are
 natom: number of MTs in the primitive cell.
    corresponding to nbas, we usually use ibas for do loop
    as "do ibas=1,nbas". Instead of nbas, we sometimes use natom.
 pos(1:3,nbas): MT centers for R within the cell
         Cartesian cordinates in the unit of alat.
 ngrp: number of space group operations.

 tiat miat: private in m_zmel
           Space group operations. See document in subrouitne mptauof in suham.F

----------
Memo for hx0fp0.sc.m.F and so on.

ixc: control of job, read by fortran read

call getkeyvalue("GWinput","ecut_p" ,ecut, default=1d10 )
    This read "ecut_p" given in GWinput.
    We can read  arrays x (real, integer, logical) by the same
    getkeyvalue (interface judge type of arguments). Instead of read the
```

```
    getkeyvalue.F, check how it is used.

call read_BZDATA():
    This allocate and give data related to the BZ.
    After it is called, we have data given in the m_read_bzdata
    as shown in "use m_read_bzdata,only:"
    ngrp: number of space group operation
    nqbz:  = n1 x n2 x n3, # of BZ
    nqibz: # of irreducible k points in the BZ.
    qbas: probably the same as qlat
    ginv: inverse of qlat (essentially the same as plat, but transposed).
    dq_: shift vector for qbzreg mode
        This is for qbzreg(). (When qbzreg=F, BZ mesh do not
contain Gamma point). This mechanism should be reconsidered.
    qbz: q point in the BZ
    qibz: qpoints in the irrecucible BZ.
    wbz:  weight. 1/(n1*n2*n3)
    wibz: weight for qibz.

qbzreg():
    If F, we use off-Gamma mesh for qbz.

Radial mesh: hbasfp0
  a,b, rofi,nr (or aa,bb, nrad).
  MT site radial data. Radial integrals are only in
  subroutine basnfp_v2 in hbasfp0.m.F.
  We assume r(ir)=b*(exp(aa*(ir-1)-1.), ir=1,nr
  rhoMT is read.
```

1.3 Representation of eigenfunctions

In the PMT method [9], the valence eigenfunctions for a given H^0 are represented in the linear combinations of the Bloch-summed MTOs $\chi^{\mathbf{k}}_{\mathbf{R}Lj}(\mathbf{r})$ and the APWs $\chi^{\mathbf{k}}_{\mathbf{G}}(\mathbf{r})$;

$$\Psi_{\mathbf{k}n}(\mathbf{r}) = \sum_{\mathbf{R}Lj} z^{\mathbf{k}n}_{\mathbf{R}Lj}\chi^{\mathbf{k}}_{\mathbf{R}Lj}(\mathbf{r}) + \sum_{\mathbf{G}} z^{\mathbf{k}n}_{\mathbf{G}}\chi^{\mathbf{k}}_{\mathbf{G}}(\mathbf{r}), \tag{1}$$

where we use indexes of the wave vector $\mathbf{k}$, band index n, and reciprocal lattice vector $\mathbf{G}$. The MTOs in the primitive cell are specified by the index of MT site $\mathbf{R}$, angular momentum $L = (l, m)$, and j for radial functions. As for core eigenfunctions, we calculate them under the condition that they are restricted within MTs. Then we take into accounts the contributions of the cores to the exchange part defined in Eq. (18) in the following. But not to the correlation part. (caution: we now usally apply "core1 treatment" give in Ref.[3] for all cores. Rarely use core2).

1.3.1 MTO part

Within MTs, the Bloch sum of the MTO, $\chi^{\mathbf{k}}_{\mathbf{R}Lj}(\mathbf{r})$, is expressed by a linear combination of atomic like orbitals $A_{\mathbf{R}Lj}(\mathbf{r}) \equiv \{\phi_{\mathbf{R}Lj}(r), \dot{\phi}_{\mathbf{R}Lj}(r), \phi^z_{\mathbf{R}Lj}(r)\} \times Y_L$. ($\phi^z$ means local orbital). These radial functions are solutions of the radial Schrödinger equations(or their energy derivatives) within $\mathbf{R}$. The MTO basis is specified by smHankel functions which contains two parameters ($E = -|\kappa|^2, R_{\rm sm}$).

$A_{\mathbf{R}Lj}(\mathbf{r})$ makes orthonormalized basis for each MT $\mathbf{R}$. Then the MTO including tail part can be written as

$$\begin{aligned} \chi^{\mathbf{k}}_{\mathbf{R}Lj}(\mathbf{r}) &= \sum_{\mathbf{R}Lj} C^{\mathbf{k}}_{\mathbf{R}Lj} A^{\mathbf{k}}_{\mathbf{R}Lj}(\mathbf{r}) \quad \text{if } \mathbf{r} \in \text{any MT} \\ &= H^{\kappa,R_s,\mathbf{k}}_{\mathbf{R}L}(\mathbf{r}) \quad \text{otherwise,} \end{aligned} \tag{2}$$

where we use the Bloch sums,

$$A^{\mathbf{k}}_{\mathbf{R}Lj}(\mathbf{r}) \equiv \sum_{\mathbf{T}} A_{\mathbf{R}Lj}(\mathbf{r}-\mathbf{R}-\mathbf{T})\exp(i\mathbf{k}\mathbf{T}), \tag{3}$$

$$H^{\mathbf{k}s}(\mathbf{r}) \equiv \sum_{\mathbf{T}} H_s(\mathbf{r}-\mathbf{R}-\mathbf{T})\exp(i\mathbf{k}\mathbf{T}). \tag{4}$$

Here the smoothe Hankel functions $H^{\mathbf{k}s}(\mathbf{r})$ are the envelope functions of MTOs.

The MTOs have the symmetry of the angular momentum L. This is also the casee for any atomic-like localized functions. All such kinds of functions follow the same mapping formulas by space-group operations. See Sec.H.

1.3.2 APW part

The APW $\chi^{\mathbf{k}}_{\mathbf{G}}(\mathbf{r})$ are given as a linear combination of atomic like orbitals $A_{\mathbf{R}Lu}(\mathbf{r}) \equiv \{\phi_{\mathbf{R}lu}(r)Y_L(\hat{\mathbf{r}}), \dot{\phi}_{\mathbf{R}lu}(r)Y_L(\hat{\mathbf{r}})\}$ within MTs, and just the usual plane waves within the interstitial region. Here $\phi_{\mathbf{R}lu}(r)$ and $\dot{\phi}_{\mathbf{R}lu}(r)$ denote two solutions of the radial Schrödinger equations at an energy `enu` for each l (an usual choice of `enu` is the center of gravity of occupied PDOS). $\dot{\phi}$ means energy derivatives (or something similar). u is the composite index to diffrenciate ϕ and $\dot{\phi}$. $\mathbf{R}$ is the index to specify MTs in the primitive cell. The APW basis is specified by $s \equiv \mathbf{R}jL$, where $L \equiv (l,m)$ is the angular momentum index, and j is the additional index (principle quantum number or so). $A_{\mathbf{R}Lu}(\mathbf{r})$ makes normalized-orthogonal basis in each MT $\mathbf{R}$. The APW can be written as

$$\begin{aligned}\chi^{\mathbf{k}+\mathbf{G}}(\mathbf{r}) &= \sum_{au} C^{\mathbf{k}+\mathbf{G}}_{au} A^{\mathbf{k}}_{au}(\mathbf{r}) \quad \text{if } \mathbf{r} \in \text{any MT} \\ &= \exp(i(\mathbf{k}+\mathbf{G})\mathbf{r}) \quad \text{otherwise},\end{aligned} \tag{5}$$

where we use the Bloch sums,

$$A^{\mathbf{k}}_{\mathbf{R}u}(\mathbf{r}) \equiv \sum_{\mathbf{T}} A_{\mathbf{R}u}(\mathbf{r}-\mathbf{R}-\mathbf{T})\exp(i\mathbf{k}\mathbf{T}), \tag{6}$$

The number of $\mathbf{G}$ is limited by the condition $|\mathbf{k}+\mathbf{G}| <$ `QpGcut_psi` (IPWpsi). The coefficients $\alpha^{\mathbf{k}n}_{au}$ can be calculated as

$$\alpha^{\mathbf{k}n}_{au} = \sum_{\mathbf{G}} C^{\mathbf{k}+\mathbf{G}}_{au} z^{\mathbf{k}+\mathbf{G}}_{n}. \tag{7}$$

1.4 Re-expansion of eigenfunctions: CPHI and GEIG

To perform the *GW* calculation, we first have to prepare all eigenfunctions (and eigenvalues) for given setting of BZ mesh. Then the eigenfunctions are represented as follows; we re-expand $\Psi_{\mathbf{k}n}(\mathbf{r})$ in Eq. (1) as the sum of the augmentation parts in MTs and the PW parts in the interstitial region.

$$\Psi_{\mathbf{k}n}(\mathbf{r}) = \sum_{\mathbf{R}u} \alpha^{\mathbf{k}n}_{\mathbf{R}u}\varphi^{\mathbf{k}}_{\mathbf{R}u}(\mathbf{r}) + \sum_{\mathbf{G}} \beta^{\mathbf{k}n}_{\mathbf{G}} P^{\mathbf{k}}_{\mathbf{G}}(\mathbf{r}), \tag{8}$$

where the interstitial plane wave (IPW) is defined as

$$P^{\mathbf{k}}_{\mathbf{G}}(\mathbf{r}) = \begin{cases} 0 & \text{if } \mathbf{r} \in \text{any MT} \\ \exp(i(\mathbf{k}+\mathbf{G})\cdot\mathbf{r}) & \text{otherwise} \end{cases} \tag{9}$$

and $\varphi^{\mathbf{k}}_{Ru}(\mathbf{r})$ are Bloch sums of the atomic functions $\varphi_{Ru}(\mathbf{r})$ defined within the MT at R,

$$\varphi^{\mathbf{k}}_{Ru}(\mathbf{r}) \equiv \sum_{\mathbf{T}} \varphi_{Ru}(\mathbf{r}-\mathbf{R}-\mathbf{T})\exp(i\mathbf{k}\cdot\mathbf{T}). \tag{10}$$

$\mathbf{T}$ and $\mathbf{G}$ are lattice translation vectors in real and reciprocal spaces, respectively. We explain how they can be represented in codes in Sec. **??**.

We expand the eigenfunctions as the sum of the augmentation parts in MTs and the PW parts in the interstitial region

$$\Psi_{\mathbf{k}n} = \sum_{\mathbf{R}u} \alpha^{\mathbf{k}n}_{\mathbf{R}u}\phi^{\mathbf{k}}_{\mathbf{R}u}(\mathbf{r}) + \sum_{\mathbf{G}} \beta^{\mathbf{k}n}_{\mathbf{G}} P^{\mathbf{k}}_{\mathbf{G}}(\mathbf{r}) \tag{11}$$

This is Eq.(17) in Ref.[2]. Here, Files CPHI contains the information of $\alpha^{\mathbf{k}n}_{\mathbf{R}u}$ and GEIG contains $\beta^{\mathbf{k}n}_{\mathbf{G}}$. We use subroutines `readcphi` and `readeig` to read them; see `m_zeml.F` for example.

(In future, we may start from better representation based on the 3 component formalism in Ref.[1].)

We need $\Psi_{\mathbf{k}n}$ for given $\mathbf{q}$ points (in this text, we mix up $\mathbf{q}$ and $\mathbf{k}$... Sorry.). Then $P^{\mathbf{k}}_{\mathbf{G}}(\mathbf{r})$ is just specified by $\mathbf{G}$, which is generated by **qg4gw**. $\phi^{\mathbf{k}}_{\mathbf{R}u}(\mathbf{r})$ is specified by radial functions. It is contained in PHIVC read in hbasfp0.m.F. Number of radial functions are `ncore(ic)+nrad(ic)` (dependes on l, n, σ, but not on m). For simplicity, maximum of l is fixed by LMXA in ctrl file. It must be the same for all MTs.

In the GW calculation of ecalj, important matrix elements related to the eigenfunction is only the matrix element as

$$\langle E^{\mathbf{q}}_{\mu}\Psi_{\mathbf{k}n}|\Psi_{\mathbf{q}+\mathbf{k}n'}\rangle, \tag{12}$$

, where $E^{\mathbf{q}}_{\mu}$ is the MPB (an unitary transformation of MPB). The information of eigenfunctions are used to calculate this matrix elements, which is read by `get_zmel`.

Coefficients of Eq. (8) (here is MTO part only) are calculated as

$$\alpha^{\mathbf{k}n}_{au} = \sum_{s} C^{\mathbf{k}s}_{au} z^{\mathbf{k}n}_{s} \tag{13}$$

$$\beta^{\mathbf{k}n}_{\mathbf{G}} = \sum_{\mathbf{G}'s} \langle P^{\mathbf{k}}_{\mathbf{G}}|P^{\mathbf{k}}_{\mathbf{G}'}\rangle^{-1}\langle P^{\mathbf{k}}_{\mathbf{G}'}|H^{\mathbf{k}s}\rangle z^{\mathbf{k}n}_{s}, \tag{14}$$

where the number of $\mathbf{G}$ is limited by the condition $|\mathbf{k}+\mathbf{G}| <$ `QpGcut_psi`; $\mathbf{G}'$ is by $|\mathbf{k}+\mathbf{G}'| <$ `QpGcutHakel`.

1.5 Overview of GW calculation

In the *GW* calculation, we need not only the basis set for eigenfunctions, but also the basis set for expanding the product of eigenfunctions. The basis is called the mixed product basis (MPB) $\{M^{\mathbf{k}}_{I}(\mathbf{r})\}$ first introduced in Ref.[10] by Kotani. The MPB consists of the product basis (PB) within MTs [11] and the IPW in the interstitial region. Since $\{M^{\mathbf{k}}_{I}(\mathbf{r})\}$ contains IPWs which are not orthogonal, we define dual for $\{M^{\mathbf{k}}_{I}(\mathbf{r})\}$ as

$$|\tilde{M}^{\mathbf{k}}_{I}\rangle \equiv \sum_{I'} |M^{\mathbf{k}}_{I'}\rangle (O^{\mathbf{k}})^{-1}_{I'I}\,, \tag{15}$$

$$O^{\mathbf{k}}_{I'I} = \langle M^{\mathbf{k}}_{I'}|M^{\mathbf{k}}_{I}\rangle. \tag{16}$$

From $v^{\mathbf{k}}_{IJ} = \langle M^{\mathbf{k}}_{I}|v|M^{\mathbf{k}}_{J}\rangle$, we calculate the eigenfunction for the generalized eigenvalue problem defined by $\sum_J (v^{\mathbf{k}}_{IJ} - v^{\mathbf{k}}_{\mu} O^{\mathbf{k}}_{IJ}) w^{\mathbf{k}}_{\mu J} = 0$, where $v_\mu(\mathbf{k})$ are the eigenvalues of the Coulomb interaction matrix. Then we have the Coulomb interaction represented by matrix elements as

$$v(\mathbf{k}) = \sum_{\mu} |E^{\mathbf{k}}_{\mu}\rangle v_{\mu}(\mathbf{k}) \langle E^{\mathbf{k}}_{\mu}|, \tag{17}$$

where we define a new MPB $|E^{\mathbf{k}}_{\mu}(\mathbf{r})\rangle = \sum_J |M^{\mathbf{k}}_{J}\rangle w^{\mathbf{k}}_{\mu J}$, which is orthonormal and is diagonal to the Coulomb interaction $v(\mathbf{k})$. For the all-electron full-potential *GW* approximation, Eq. (17) is introduced in Ref.[7]. This corresponds to the representation in the plane wave expansion $v(\mathbf{k}+\mathbf{G}, \mathbf{k}+\mathbf{G}') = \frac{4\pi\delta_{\mathbf{G}\mathbf{G}'}}{|\mathbf{k}+\mathbf{G}|^2}$. $\mu = 1$ corresponds to the largest eigenvalue of v_μ, and $v_{\mu=1}$ is $\sim \frac{4\pi e^2}{|\mathbf{k}|^2}$, which is related to the divergent term discussed in Sec.5.

With the definition of $\langle A|B\rangle = \int d^3r A^*(\mathbf{r})B(\mathbf{r})$, the exchange part of $\Sigma(\omega)$ is written as

$$\Sigma^{\mathrm{x}}_{nm}(\mathbf{q}) = \langle \Psi_{\mathbf{q}n}|\Sigma_{\mathrm{x}}|\Psi_{\mathbf{q}m}\rangle = -\sum_{\mathbf{k}}^{\mathrm{BZ}}\sum_{n'}^{\mathrm{occ}} \langle \Psi_{\mathbf{q}n}|\Psi_{\mathbf{q}-\mathbf{k}n'}E^{\mathbf{k}}_{\mu}\rangle v_{\mu}(\mathbf{k}) \langle E^{\mathbf{k}}_{\mu}\Psi_{\mathbf{q}-\mathbf{k}n'}|\Psi_{\mathbf{q}m}\rangle. \tag{18}$$

The screened Coulomb interaction $W(\omega)$ is calculated from

$$W = \epsilon^{-1}v = (1 - v\Pi)^{-1}\, v, \tag{19}$$

where the Lindhard polarization function $\Pi(\omega)$ is written as

$$\begin{aligned}\Pi_{\mu\nu}(\mathbf{q},\omega) \quad &= \sum_{\mathbf{k}}^{\mathrm{BZ}}\sum_{n}^{\mathrm{occ}}\sum_{n'}^{\mathrm{unocc}} \frac{\langle E^{\mathbf{q}}_{\mu}\Psi_{\mathbf{k}n}|\Psi_{\mathbf{q}+\mathbf{k}n'}\rangle\langle\Psi_{\mathbf{q}+\mathbf{k}n'}|\Psi_{\mathbf{k}n}E^{\mathbf{q}}_{\nu}\rangle}{\omega - (\varepsilon_{\mathbf{q}+\mathbf{k}n'} - \varepsilon_{\mathbf{k}n}) + i\delta} \\ &+ \sum_{\mathbf{k}}^{\mathrm{BZ}}\sum_{n}^{\mathrm{unocc}}\sum_{n'}^{\mathrm{occ}} \frac{\langle E^{\mathbf{q}}_{\mu}\Psi_{\mathbf{k}n}|\Psi_{\mathbf{q}+\mathbf{k}n'}\rangle\langle\Psi_{\mathbf{q}+\mathbf{k}n'}|\Psi_{\mathbf{k}n}E^{\mathbf{q}}_{\nu}\rangle}{-\omega - (\varepsilon_{\mathbf{k}n} - \varepsilon_{\mathbf{q}+\mathbf{k}n'}) + i\delta}.\end{aligned} \tag{20}$$

When time-reversal symmetry is assumed (if $\Psi_{\mathbf{k}n}$ is the eigenfunction, $\Psi^*_{\mathbf{k}n}$ is also the eigenfuntion with the same eigenvalue $\varepsilon_{\mathbf{k}n}$), $\Pi(\omega)$ is simplified to be

$$\Pi_{\mu\nu}(\mathbf{q},\omega) = \sum_{\mathbf{k}}^{\mathrm{BZ}} \sum_{n}^{\mathrm{occ}} \sum_{n'}^{\mathrm{unocc}} \langle E^{\mathbf{q}}_{\mu}\Psi_{\mathbf{k}n}|\Psi_{\mathbf{q}+\mathbf{k}n'}\rangle\langle\Psi_{\mathbf{q}+\mathbf{k}n'}|\Psi_{\mathbf{k}n}E^{\mathbf{q}}_{\nu}\rangle \times\left(\frac{1}{\omega-\varepsilon_{\mathbf{q}+\mathbf{k}n'}+\varepsilon_{\mathbf{k}n}+i\delta}-\frac{1}{\omega+\varepsilon_{\mathbf{q}+\mathbf{k}n'}-\varepsilon_{\mathbf{k}n}-i\delta}\right). \quad (21)$$

This definition is the same as that in the Fetter-Walecka text book. At $\omega=0$, this is negative definite matrix. Im part at any ω is negative definite matrix. This is even function of ω.

To evaluate Eq. (20) or Eq. (21), we first accumulate the imaginary parts (anti-Hermitian part) of $\Pi_{\mu\nu}(\mathbf{q},\omega)$ along bins of histograms on the real axis ω by the tetrahedron technique [12], and then determine the real part via the Hilbert transformation. The bins are dense near the Fermi energy and coarser at higher energy as described in Ref.[6]. This procedure is not only more efficient but also safer than the methods of calculating the real part directly. We also use the extended irreducible zone (EIBZ) symmetrization procedure described in Ref.[7].

The correlation part of the screened Coulomb interaction $W^c(\omega)=W(\omega)-v$, which is calculated from v and $\Pi(\omega)$, is given as

$$W^{\mathrm{c}}(\mathbf{k},\omega)=\sum_{\mu\nu}|E^{\mathbf{k}}_{\mu}\rangle W^{\mathrm{c}}_{\mu\nu}(\mathbf{k},\omega)\langle E^{\mathbf{k}}_{\mu}|. \quad (22)$$

With this $W^{\mathrm{c}}(\mathbf{k},\omega)$, we have the correlation part of the self-energy as

$$\Sigma^{\mathrm{c}}_{n,n'}(\mathbf{q},\omega)=\sum_{\mathbf{k},m}\int_{-\infty}^{\infty}d\omega'\sum_{\mu,\nu}\frac{\langle\Psi_{\mathbf{q}n}|\Psi_{\mathbf{q}-\mathbf{k}m}E^{\mathbf{k}}_{\mu}\rangle W^{\mathrm{c}}_{\mu\nu}(\mathbf{k},\omega')\langle E^{\mathbf{k}}_{\nu}\Psi_{\mathbf{q}-\mathbf{k}m}|\Psi_{\mathbf{q}n'}\rangle e^{-i\delta\omega'}}{\omega-\omega'-\epsilon_{\mathbf{q}-\mathbf{k}m}\pm i\delta}. \quad (23)$$

Here, we use $-i\delta$ for occupied states of $\mathbf{q}-\mathbf{k}m$, and $+i\delta$ for unoccupied states. In QSGW, we have to calculate the Hermitian part of $\Sigma_{nn'}(\mathbf{q},\epsilon_{\mathbf{q}n})$, to obtain $V^{\mathrm{xc}}_{\mathbf{q}}$ using Eq. (**??**).

There are two key points to handle the *GW* procedure given above. The first key point, given in Sec.5, is the improved offset-Γ method, which treats the divergence of $W^{\mathrm{c}}(\mathbf{k}\to 0,\omega)$ in Eq. (23). For this purpose, we define the non-divergent effective interaction $\overline{W^{\mathrm{c}}}(\mathbf{k}=0,\omega)$ instead of $W^{\mathrm{c}}(\mathbf{k}=0,\omega)$. Then we can take a simple discrete sum for both expressions of Eqs.(18) and (23).

The second point in Sec.10 is how to perform an interpolation to give $V^{\mathrm{xc}}_{\mathbf{q}}$ at any $\mathbf{q}$ in the whole BZ, from $V^{\mathrm{xc}}_{\mathbf{q}}$ calculated only at limited numbers of $\mathbf{q}$ points. This is required in the offset-Γ method shown in Sec.5, that is, we have to calculate eigenfunctions at some $\mathbf{q}$ points near $\mathbf{q}=0$. For the interpolation, we expand the static nonlocal potential V^{xc} in Eq. (**??**) in highly localized MTOs in real space. Thus such MTOs are used for two purposes: one as the basis of the eigenfunctions; and two as the basis of expanding V^{xc}. The interpolation procedure of $V^{\mathrm{xc}}_{\mathbf{k}}(\mathbf{r},\mathbf{r}')$ becomes stabler and simpler than the complicated interpolation procedure in Ref.[6]. This is because we now use highly localized MTOs. In the planewave-based QSGW method by Hamann and Vanderbilt [13], they expand V^{xc} in the maximally localized Wannier functions instead of MTOs.

In practical implementation, the LDA or GGA exchange-correlation potential $V^{\mathrm{xc}}_{\mathrm{LDA}}$ is used to perform efficient numerical calculations. That is, it is used in order to generate core eigenfunctions as well as radial functions within MTs (in this paper, we use the subscript LDA even when we use GGA. "LDA/GGA" means LDA or GGA). The difference $V^{\mathrm{xc}}-V^{\mathrm{xc}}_{\mathrm{LDA}}$ is used for the interpolation in the BZ (explained in Sec.10), because this difference is numerically small as long as $V^{\mathrm{xc}}_{\mathrm{LDA}}$ roughly gives an approximation to V^{xc}. This procedure utilizing $V^{\mathrm{xc}}_{\mathrm{LDA}}$ to perform efficient numerical calculations give a very weak dependence to the final numerical results in practice as seen in Sec.**??**, although the results formally does not depend on the LDA/GGA exchange-correlation functions.

2 q and G vector generation. qg4gw

... xxxxx under construction xxxxx...

To see the theory, read Section 3.2 in in [2] $\mathbf{k}$ points are the regular mesh points used for the

integration in the BZ. Regular mesh points are given as

$$\mathbf{k}_{i_1,i2,i3} = (i1/N1) * \mathbf{P}_1 + (i2/N2) * \mathbf{P}_2 + (i3/N3) * \mathbf{P}_3 \tag{24}$$

, where $\mathbf{P}_i$ is the primitive vector $\frac{2\pi}{\text{alat}}\times$ Quantities $f(\mathbf{k})$ (periodic in BZ) can be integrated just as a sum on the regular mesh points. However, when $f(\mathbf{k}=0)$ is divergent, we have careful treatments.

`qlat`(:,i1), where qlat=$\mathbf{Q}_i$ is the reciprocal vectors. See Sec.3.2 .

mkqg.F is the main part of **qg4gw** (fpgw/main/qg4gw.F). iq0pin is the job control of mkqg. iq0pin=101 is only for backward compatibility. The purpose is generates required q and G vectors. Not only regular mesh points, or q along symmetry lines, but also offset Gamma points.

iq0pin=1: normal mode. Generate q and G for regular mesh point and Q0P points (offset Gamma points).

iq0pin: input to qg4gw

`ncindx, lcindx`

`getkeyvalue`
`phi`

`radial mesh`
`CLASS`
`symgg`
`core, radial functions`

2.0.1 Make G vectors: getgv2

To get $\mathbf{G}$ vectors, we use an algorithm in `fpgw/getgv2.F`, whose head is

```
      subroutine getgv2(alat,plat,qlat,q, QpGcut,job,
     o                  ng, ngvec)
!! == Set up a list of recip vectors within cutoff |Q+G| < QpGcut a.u. ==
!! job==1 -> return ng (number of G ) and imx(as ngvec(1,1));mar2012takao add imx.
!! job==2 -> return ng and ngvec
!! True G is given as
!!    G(1:3,1:ng) = 2*pi/alat * matmul(qlat * ngvec(1:3,1:ng))
!! NOTE: we need some geometorial consideration for this routine.
!!   Consiser ellipsoid. Takao need to give more detailed explanation...
!! ---------------------------------------------------------------
```

We can use this to get $\{\mathbf{G}\}$ for given q. The algorism of `getgv2` is a little complicated. We first gives the upper and lower limits `n1max` $\leq n_1 \leq$ `n1min`, where $\mathbf{G} = n_1 Q_1 + n_2 Q_2 + n_3 Q_3$, n_2 and n_3 as well.

... xxxxx under construction xxxxx...

Algorithm of `getgv2`.
Let us consider the three dimensional space of $\mathbf{x} = \mathbf{q} + \mathbf{G}$. For given $\mathbf{q}$, allowed $\mathbf{G}$ make a set of regular mesh points $\{\mathbf{q} + \mathbf{G}\}$. The purpose of `getgv2` is picking up only mesh points satisfying $|\mathbf{q} + \mathbf{G}|$ <`QpGcut` among these mesh points.

At first, we can calculate allowed range of n_1 for given maximum of $|\mathbf{q} + \mathbf{G}|$ (=`QpGcut`). Note $|\mathbf{x}| = |\mathbf{q} + \mathbf{G}|$=`QpGcut` gives a sphere; we have to pick up mesh points within the sphere. When we spefcify n_1, we have a plane (allowing n_2, n_3 can take any values). The range is determined by the condition that the sphere $|\mathbf{q} + \mathbf{G}|$=`QpGcut` cross the plane specified by n_1 (exactly speaking, such n_1 is real number). The vector normal to the plane is the external product $Q_2 \times Q_3$.

After we get the range of n_1, as well as n_2, n_3 , we simply test whether $\mathbf{q}+\mathbf{G}$ for (n_1, n_2, n_3) is allowed or not.

2.0.2 Make G vectors: shortn3

Find shortest vector in modulo of $\{Q_i\}$. That is, pull back $\mathbf{q}$ in the 1st BZ. Caution; it can be not unique when $\mathbf{q}$ is on the BZ boundary; then we need to know all $\mathbf{q}$ and degeneracy.

3 Mixed Product basis

The mixed product basis consists of two types of basis sets, that is the product basis and the IPW: $\{M_I^{\mathbf{k}}(\mathbf{r})\} \equiv \{P_{\mathbf{G}}^{\mathbf{k}}(\mathbf{r}), B_{\mathbf{R}\mu}^{\mathbf{k}}(\mathbf{r})\}$, where the index $I \equiv \{\mathbf{G}, \mathbf{R}\mu\}$ classifies the members of the basis. The PB $B_{\mathbf{R}\mu}^{\mathbf{k}}(\mathbf{r})$ is defined as

$$B_{\mathbf{R}\mu}^{\mathbf{k}}(\mathbf{r}) \quad = \quad \sum_{\mathbf{T}} B_{\mathbf{R}\mu}(\mathbf{r}-\mathbf{R}-\mathbf{T})e^{i\mathbf{k}\cdot\mathbf{T}}, \tag{25}$$

where $B_{\mathbf{R}\mu}(\mathbf{r})$ is made from the products of radial functions. $B_{\mathbf{R}\mu}(\mathbf{r})$ is real and zero for outside of MT, $|\mathbf{r}| > R$ (See Sec.3.1). We set up $\{B_{\mathbf{R}\mu}(\mathbf{r})\}$ so that they are orthonormalized;

$$\int_{|\mathbf{r}|<R} B_{\mathbf{R}\mu}(\mathbf{r})B_{\mathbf{R}\mu'}(\mathbf{r})d^3r = \delta_{\mu\mu'}. \tag{26}$$

In addition, it is trivial that $\{B_{\mathbf{R}\mu}(\mathbf{r})\}$ and $\{P_{\mathbf{G}}^{\mathbf{k}}(\mathbf{r})\}$ are orthogonal. Thus only the elements of overlap matrix is

$$O_{IJ}^{\mathbf{k}} = \int_{\Omega} \{M_I^{\mathbf{k}}(\mathbf{r})\}^* M_J^{\mathbf{k}}(\mathbf{r})d^3r, , \tag{27}$$

because $\{P_{\mathbf{G}}^{\mathbf{k}}(\mathbf{r})\}$ are not orthogonal. Thus it is convenient to define the dual of $M_I^{\mathbf{k}}(\mathbf{r})$ as $\tilde{M}_I^{\mathbf{k}}(\mathbf{r})$ in the manner of Eq. (16).

Functions made from the product of eigenfunctions can be virtually completely expanded in the basis of $M_I^{\mathbf{k}}(\mathbf{r})$ in this manner;

$$\begin{cases} F^{\mathbf{k}}(\mathbf{r}) = \sum_I^I M_I^{\mathbf{k}}(\mathbf{r})F_I(\mathbf{k}) \\ F_I(\mathbf{k}) = \int_{\Omega} \{\tilde{M}_I^{\mathbf{k}}(\mathbf{r})\}^* F^{\mathbf{k}}(\mathbf{r})d^3r. \end{cases} \tag{28}$$

—- NOTE: —- Now we use Coulomb matrix orthogonalized basis $\{E_\mu^{\mathbf{k}}\}$ as discussed in Eq. (17).

3.1 Product basis (hbasfp0)

We denote the radial function of atom a as

$$u_{apl\sigma}(r) = r\phi_{apl\sigma}(r), \tag{29}$$

where the index p takes 1 for ϕ, 2 for $\dot{\phi}$, 3 for local orbital as well. We do not allow m dependence (m is m of $L = (l, m)$.) for the radial functions. In addition, p can taks indexes to specify core functions: we combine core and valence functions (these are stored in `PHIVC`, which read in `hbasfp0.m.F`). Here is a part of copy to read `PHIVC` in hvccfp0.m.F

```
      ifphi  = iopen('PHIVC', 0,-1,0)! augmentation wave and core
      read(ifphi) nbas, nradmx, ncoremx
      allocate(  ncindx(ncoremx,nbas),
     &           lcindx(ncoremx,nbas),
     &           nrad(nbas), nindx_r(1:nradmx,1:nbas),
     &           lindx_r(1:nradmx,1:nbas),
     &        aa(nbas),bb(nbas),zz(nbas), rr(nrx,nbas), nrofi(nbas) ,
     &        phitoto(nrx,0:nl-1,nn,nbas,nsp),
     &        phitotr(nrx,0:nl-1,nn,nbas,nsp),
     &        nc_max(0:nl-1,nbas),ncore(nbas) )
      read(ifphi) nrad(1:nbas)
      read(ifphi) nindx_r(1:nradmx,1:nbas),lindx_r(1:nradmx,1:nbas)
      nc_max=0
      do ibas=1,nbas
        write(6,*)' --- read PHIVC of ibas=',ibas
        ic = ibas
        read(ifphi) ncore(ic), ncoremx                              !core
        read(ifphi) ncindx(1:ncoremx,ibas),lcindx(1:ncoremx,ibas) !core
        read(ifphi) icx,zz(ic),nrofi(ic),aa(ic),bb(ic)
        if(ic/=icx) then
          write(6,*) 'ic icx=',ic,icx
```

```
        call rx( 'hbasfp0: ic/=icx')
      endif
      read(ifphi) rr(1:nrofi(ic),ic)
      do isp = 1, nsp
        write(6,*)'---  isp nrad ncore(ic)=',isp, nrad(ic),ncore(ic)
        do icore = 1, ncore(ic)
          l =  lcindx(icore,ic)
          n =  ncindx(icore,ic)
          read(ifphi) phitoto(1:nrofi(ic),l,n, ic,isp)!core orthogonal
          phitotr(1:nrofi(ic),l,n, ic,isp)=   !we set core raw= core orthgonal
&         phitoto(1:nrofi(ic),l,n, ic,isp)
          if(n>nc_max(l,ic)) nc_max(l,ic)=n
        enddo
        do irad = 1, nrad(ic)
          l = lindx_r (irad,ic)
          n = nindx_r (irad,ic) + nc_max(l,ic)
          read(ifphi) phitoto(1:nrofi(ic),l,n, ic,isp) !valence orthogonal
          read(ifphi) phitotr(1:nrofi(ic),l,n, ic,isp) !valence raw
        enddo
      enddo
    enddo
```

[note: The orthonomalized radial functions $u_{apl\sigma}(r)$ are stored in `phitoto`; we also have the un-orthonormalized ones in `phitotr`.]

Note that the *true* radial function is $\phi_{apl\sigma}(r) = u_{apl\sigma}(r)/r$. Normalization is $1 = \int_0^{R_a} \{u_{apl\sigma}(r)\}^2 dr = \int_0^{R_a} \{\phi_{apl\sigma}(r)\}^2 r^2 dr$. The function $u_{apl\sigma}(r)$ is stored in `phitot`.

When producing the product functions, we use spin-averaged function `phiav` given as

$$u_{apl}(r) = \frac{1}{N_{\rm spin}} \sum_\sigma u_{apl\sigma}(r). \tag{30}$$

(See `subroutine basnfp_v2`). From them, we make the product functions `rprod`

$$\tilde{b}_{al\nu}(r) = \frac{1}{r} u_{apl}(r) u_{ap'l'}(r) = r\phi_{apl}(r)\phi_{ap'l'}(r), \tag{31}$$

where the index l runs $|l-l'| \le l \le |l+l'|$; ν is the index of the combination (p, p'). Note the *true* product functions are given as

$$\tilde{B}_{al\nu}(r) = \frac{1}{r} \tilde{b}_{al\nu}(r). \tag{32}$$

This relation is as same as $\phi_{apl}(r) = u_{apl}(r)/r$.

Then we calculate the overlap matrix `ovmt`,

$$O_{\nu_1\nu_2} = \int_0^{R_a} \tilde{B}_{al\nu_1}(r)\tilde{B}_{al\nu_2}(r) r^2 dr = \int_0^{R_a} \phi_{ap_1l_1}(r)\phi_{ap'_1l'_1}(r)\phi_{ap_2l_2}(r)\phi_{ap'_2l'_2}(r) r^2 dr \tag{33}$$

and solve the eigenvalue problem of the overlap matrix, $Oz_\nu = \epsilon_\nu z_\nu$, by `call rs(..)`. (See `basnfp_v2`.)

After neglecting eigenvectors z_ν with eigenvalues $\epsilon_\nu <$ tolerance $\sim 10^{-4}$ (given in GWinput, we finally have the optimal product functions as the linear combinations of the product functions as

$$b_{al\nu}(r) = \frac{1}{\sqrt{\epsilon_\nu}} \sum_{\nu'} \tilde{b}_{al\nu'}(r) z_{\nu'\nu}, \tag{34}$$

which are stored in `rprodx` and written into BASFP* and used in the successive Coulomb matrix routine hvccfp0.m.f. Of course, *true* product function is $B_{al\nu}(r) = b_{al\nu}(r)/r$.

We check the normalization of the optimal product function in standard output (See `lbasC` and `lbas` when you did **gw_lmf**):

```
...
Use rs diagonalization for real symmetric
Diag ibx ovv=  1 0.9999999999999930D+00 eb=   0.2716113799D-01 nod=   2
Diag ibx ovv=  2 0.9999999999999980D+00 eb=   0.4993303381D-01 nod=   3
Diag ibx ovv=  3 0.1000000000000001D+01 eb=   0.1467546915D+00 nod=   3
Diag ibx ovv=  4 0.9999999999999996D+00 eb=   0.4415639258D+01 nod=   0

...
```

In `basnfp`, we calculate all the required radial integrations $< \phi\phi B >$=`ppbrd`;

$$< \phi\phi B >= \int_0^{R_a} \phi_{ap_1 l_1}(r)\phi_{ap_2 l_2}(r)B_{al\nu}(r)r^2 dr = \int_0^{R_a} \frac{1}{r}u_{ap_1 l_1}(r)u_{ap_2 l_2}(r)b_{al\nu}(r)dr, \quad (35)$$

which are stored into PPBRD*. At `call rdpp(...` in hx0fp0.m.f hxfp0.m.f, we allocate and read `ppbrd`.

In addition, we read the "rotated Clebsh-Gordon coefficient" $C(L, L_1, L_2, g)$ `cgr(lm,lm1,lm2,ng)` , where g is the index for space group coefficient (rotated by point group symmetries).

4 The Coulomb matrix (hvccfp0.m.F)

We have to calculate $v_{IJ}^{\mathbf{k}} = \langle M_I^{\mathbf{k}}|v|M_J^{\mathbf{k}}\rangle$, which appears right after Eq. (16). Our `hvccfp0.m.F` can handle the case $\frac{\exp(-|\kappa||\mathbf{r}_1-\mathbf{r}_2|)}{|\mathbf{r}_1-\mathbf{r}_2|}$. The default is the bare Coulomb interaction, that is, $\kappa = 0$. The energy variable E is given by $E = \kappa^2$, where imaginary part of κ is defined to be positive for negative E. This E =`eee=screenfac()` in hvccfp0.m.F. For example, we can set $|\kappa| = 0.1$ (a.u.) by a line "`TFscreen 0.1`" in GWinput if necessary. Look for `TFscreen` in `swithes.F`.

The MPB is made of IPWcou (|**q**+**G**|<`QpGcut_cou`) and PB, that is, $\{M_J^{\mathbf{k}}\} = \{P_{\mathbf{G}}^{\mathbf{k}}(\mathbf{r}), B_{\mathbf{R}li L\mu}^{\mathbf{k}}(\mathbf{r})\}$. For given $\mathbf{k}$ (not explicitly shown in cases for simplicity), IPWcou is specified just by the $\mathbf{G}$ vector. PB can be specified just by the radial functions for each l PB is generated in the manner of Sec.3 . (because we have neither m nor spin dependence). For GW calculation, we need the Coulomb matrix elements $\langle B|v|B\rangle$, $\langle B|v|P_{\mathbf{G}}\rangle$, and $\langle B|v|P\rangle$. `hvccfp0.m.F-->vcoulq_4` handles these calculations.

4.1 Spherical Bessel and related functions

We use a notation such that $X_L(\mathbf{r}) = X_l(r)Y_L(\hat{r})$; note that their radial part is dependent only on l. The ordinary definition of L-dependent spherical For given energy E, Bessel functions $J_L(E,\mathbf{r}), H_L(E,\mathbf{r})$ are (here we introduce $\sqrt{E} = \kappa$, where $\sqrt{E} = \kappa = i|\kappa|$ for $E < 0$),

$$J_L(E,\mathbf{r}) = j_l(\sqrt{E}r)Y_L(\hat{r}) = \begin{cases} j_l(i|\kappa|r)Y_L(\hat{r}) \text{ for } E < 0, \\ j_l(\kappa r)Y_L(\hat{r}) \text{ for E} > 0, \end{cases}$$
$$H_L(E,\mathbf{r}) = h_l(i|\kappa|r)Y_L(\hat{r}) \text{ for E} < 0, \quad (36)$$

where $j_l(z)$ and $h_l(z)$ are usual spherical Bessel and Hankel functions which behaves

$$j_l(z) \sim \frac{z^l}{(2l+1)!!}$$
$$h_l(z) \sim \frac{-i(2l-1)!!}{z^{l+1}} \quad (37)$$

at $z \sim 0$. For convenience, we define the **Methfessel's Bessel functions (convension)** $\bar{J}_L = \bar{J}_l Y_L$ and $\bar{H}_L = \bar{H}_l Y_L$, where

$$\bar{J}_l(E,\mathbf{r}) = j_l(i|\kappa|r)/(i|\kappa|)^l$$
$$\bar{H}_l(E,\mathbf{r}) = h_l(i|\kappa|r)i(i|\kappa|)^{l+1}. \quad (38)$$

(memo: for example, $7!! = 7\cdot5\cdot3\cdot1$ `;function fac2m(i)`). These are real functions for $E \leq 0$. See note for `genjh` in `mkjp.F`. At $E = 0$, this is reduced to be

$$\bar{J}_l(E=0,\mathbf{r}) = r^l/(2l+1)!!,$$
$$\bar{H}_l(E=0,\mathbf{r}) = r^{-l-1}(2l-1)!!. \quad (39)$$

(We use very small negetive $E = -10^{-5}$ as default instead of $E = 0$ for Coulomb matrix generator `hvccfp0.m.F` to avoid numerical troubles (see default `screenfac` in switch.F)).

The source codes to define bessl is a little confusing because of some convensions are mixed up...(note at the beginning of besslr.F).

Main one is `call bessel(ex2,lx,phi(0:lx),psi(0:lx))` in mkjp.F. This is defined in `bessl(ex2,lmax,phi,psi)` in `besslr.F` (it calls `besslr` with `loka=F`). `lx` is the upper limit of l. For given `ex2`= $E \times x^2$, this return spherical Bessel functions $j_l(\kappa x)$ in the following

manner.

$$\left.\begin{array}{l}\bar{J}_l(E,\mathbf{r}) = j_l(i|\kappa|r)/(i|\kappa|)^l = \texttt{phi(l)}r^l \\ \bar{H}_l(E,\mathbf{r}) = h_l(i|\kappa|r)i(i|\kappa|)^{l+1} = \texttt{psi(l)}/r^{l+1}\end{array}\right\} \quad \text{for } E<0,\ \kappa = i|\kappa|, |\kappa| = \sqrt{|E|}$$

$$\left.\begin{array}{l}\bar{J}_l(E,\mathbf{r}) = j_l(\kappa r)/(\kappa)^l = \texttt{phi(l)}r^l \\ \bar{N}_l(\kappa r) = \texttt{psi(l)}\frac{1}{(\kappa)^{l+1}} = \texttt{psi(l)}/r^{l+1}\end{array}\right\} \quad \text{for } E>0,\ \kappa = \sqrt{E} \tag{40}$$

for $0 \le l \le$ `lmax`. These definition gives `phi`$= 1/(2l+1)!!$ and `psi`$= (2l-1)!!$ at $E \to 0$. Here $n_l(r)$ is the spherical Neumann functions. That is, `psi` is for Hankel function for negative E, and for Neumann function for positive E. See `hansmr.F`, for example. `psi` is not used so often. We have `lm7K/subs/besslr.F`, it is similar.

For convenience, we sometimes use

$$\begin{aligned} R^{+l} &= \bar{J}_l(E,r)(2l+1)!! = \texttt{psi(l)}r^l(2l+1)!!, \\ R^{-l} &= \bar{H}_l(E,r)/(2l-1)!! = \texttt{psi(l)}r^{-l-1}(2l-1)!!. \end{aligned} \tag{41}$$

Here note $R^{+k} \to r^k$ for $E \to 0$,and $R^{-k} \to r^{-k-1}$ for $E \to 0$. These rR^{+k}, rR^{-k} are `rkpr,rkmr` generated in subroutine `genjh`.

xxxx We may need to simplify our treatment of bessel functions in future... xxxx

Electron-phonon coupling in MPB To calculate derivative for electron-phonon (EP) coupling, we need

$$\frac{\partial}{\partial \mathbf{r}_i}\left(\bar{J}_l(E,\mathbf{r})Y_L(\hat{r})\right)\big|_{\mathbf{r}=0} = \frac{1}{3}\sqrt{\frac{3}{4\pi}}, \quad \text{only for } L=1, \text{ zerootherwize.} \tag{42}$$

For real spherical harmonics, cases are: $i=y$ and $L=(1,-1)$; $i=z$ and $L=(1,0)$; $i=x$ and $L=(1,1)$. Because we use MPB, it is necessary to evaluate the the bare matrix element of the Coulomb interaction between nucleus and $\langle B\rangle$ as

$$\langle B|\frac{\partial v(\mathbf{r}-\mathbf{R})}{\partial \mathbf{R}_i}\rangle \tag{43}$$

4.2 Green function

(Readers can skip this subsection). We explain the free-space Green's function $G(\mathbf{r}-\mathbf{r}',E)$ here. Let us start from $G(\mathbf{r}-\mathbf{r}',E)$, which satisfies

$$(E+i\delta+\nabla^2)G(\mathbf{r}-\mathbf{r}',E) = \delta(\mathbf{r}-\mathbf{r}'). \tag{44}$$

($+i\delta$ is to specify the boundary condition along time axis. This pick up the retarded Green's function). Roughly speaking, this is $(\omega - H)G = 1$. Its Fourier transform is easily written as $G(\mathbf{k},E) = 1/(E-|\mathbf{k}|^2+i\delta)$. We apply back Fourier transformation to this, and get $G(\mathbf{r}-\mathbf{r}',E)$. It is

$$G(\mathbf{r}-\mathbf{r}',E) = -\frac{1}{4\pi}\frac{e^{i\kappa|\mathbf{r}-\mathbf{r}'|}}{|\mathbf{r}-\mathbf{r}'|}, \tag{45}$$

where $\kappa \equiv \sqrt{E}$; Imaginary part of κ is positive for $E<0$, that is, $\kappa = i|\kappa| = i\sqrt{-E}$ for negative E. Eq. (45) is nothing but the solution of Helmhorz differential equation; it reduces to the usual Poisson equation at $E=0$. For $E<0$, we have Thomas-Fermi type function; the numerator of Eq. (45) is $\exp(i\kappa|\mathbf{r}-\mathbf{r}'|) = \exp(-|\kappa||\mathbf{r}-\mathbf{r}'|)$.

4.3 Used formulas

$$\frac{1}{|\mathbf{r}-\mathbf{r}'|} = 4\pi\sum_K \frac{r_<^k}{r_>^{k+1}}\frac{1}{2k+1}Y_K^*(\hat{\mathbf{r}})Y_K(\hat{\mathbf{r}}) \tag{46}$$

See Appendix A in Ref.[14]. This is generalized to be

$$\frac{e^{-|\kappa||\mathbf{r}-\mathbf{r}'|}}{|\mathbf{r}-\mathbf{r}'|} = 4\pi\sum_K R^{+k}(r_<)R^{-k}(r_>)\frac{1}{2k+1}Y_K^*(\hat{\mathbf{r}})Y_K(\hat{\mathbf{r}}) = 4\pi\sum_L \bar{J}_L(E,r_<)\bar{H}_L(E,r_>), \tag{47}$$

The definition of $\bar{J}, R^{+l}$ are in Eq. (38) and Eq. (41). Here $E<0$ and $\kappa = i|\kappa| = \sqrt{-E}$.

$$\exp(i\mathbf{kr}) = 4\pi \sum_L i^l j_l(|\mathbf{k}|r) Y_L^*(\widehat{\mathbf{k}}) Y_L(\widehat{\mathbf{r}}) \tag{48}$$

$$\frac{2l+1}{4\pi} P_l(\cos\Theta) = \sum_m Y_L^*(\widehat{\mathbf{r}}_1) Y_L(\widehat{\mathbf{r}}_2) \qquad [\cos\Theta = \widehat{\mathbf{r}}_1 \cdot \widehat{\mathbf{r}}_2]. \tag{49}$$

$$\langle P_{\mathbf{G}}^{\mathbf{k}} | P_{\mathbf{G}'}^{\mathbf{k}} \rangle = \Omega \delta_{\mathbf{G},\mathbf{G}'} - \sum_{a,L} \exp(i(\mathbf{G}-\mathbf{G}')\mathbf{R}_{\mathbf{a}}) \times Y_L(\widehat{\mathbf{k}+\mathbf{G}'}) Y_L(\widehat{\mathbf{k}+\mathbf{G}}) \times \int_0^{R_a} j_l(|\mathbf{k}+\mathbf{G}|r) j_l(|\mathbf{k}+\mathbf{G}'|r) 4\pi^2 r^2 dr, \tag{50}$$

4.4 Hankel function and Structure constant

We can expand $v(\mathbf{r},\mathbf{r}') = \frac{e^{-|\kappa||\mathbf{r}-\mathbf{r}'|}}{|\mathbf{r}-\mathbf{r}'|}$ in the one-center expansion Eq. (47).

For the Hankel function in Eq. (47), we use the off-center expansion theorem of the Hankel function Eq. (51), that is, a Hankel whose center is at $\mathbf{X} \equiv \mathbf{R}+\mathbf{T}$, $H_L(\mathbf{r}-\mathbf{X})$, can be expanded in the Bessel functions whose center is at $\mathbf{X}'$;

$$\bar{H}_L(E, \mathbf{r}-\mathbf{X}) = \frac{1}{4\pi} \sum_{L'} \bar{J}_{L'}(E, \mathbf{r}-\mathbf{X}') S_{\mathbf{X}'L',\mathbf{X}L} \ , \tag{51}$$

where the Hankel function for negative energy E. Here E-dependence of $S_{\mathbf{X}'L,\mathbf{X}L'}$ is not explicitly shown. Note the difference between $\bar{J}_L$ and J_L ($\bar{H}_L$, as well).

Thus, for $(\mathbf{R}',\mathbf{T}') \neq (\mathbf{R},\mathbf{T})$, we have two-center expansion;

$$\frac{e^{-|\kappa||\mathbf{r}+\mathbf{R}+\mathbf{T}-(\mathbf{r}'+\mathbf{R}'+\mathbf{T}')|}}{|\mathbf{r}+\mathbf{R}+\mathbf{T}-(\mathbf{r}'+\mathbf{R}'+\mathbf{T}')|} = \sum_L \sum_{L'} \bar{J}_L(E,\mathbf{r}) S_{\mathbf{R}+\mathbf{T}L,\mathbf{R}'+\mathbf{T}'L'} \bar{J}_{L'}(E,\mathbf{r}') \tag{52}$$

for $(\mathbf{R}',\mathbf{T}') \neq (\mathbf{R},\mathbf{T})$.

The Bloch sum of $S_{L,\mathbf{X}L'}$ gives the structure constant of $\mathbf{k}$ as

$$S^{\mathbf{k}}_{\mathbf{R}L,\mathbf{R}'L'} = \sum_{\mathbf{T}'} S_{\mathbf{R}L,\mathbf{R}'+\mathbf{T}'L'} \exp(i\mathbf{kT}'). \tag{53}$$

Usually we use bare Coulomb at $E = 0$.

No 4π factor in the definition of $S^{\mathbf{k}}_{\mathbf{R}L,\mathbf{R}'L'}$

See the top of `strxq` defined in strxq.F (called in hvccfp0.m.F). This routine is for the one-center expansion of usual Bloch summed Hankels. (not for smooth Hankels). This result is finally converted to be the Bloch sum of the structure constant $S^{\mathbf{k}}_{\mathbf{R}L,\mathbf{R}'L'}$ used in Eq. (54).

==========

NOTE in strxq.F; it says

```
-----------------------------------
Cr Expansion Theorem: H_{RL}(r) = H_L(r-R)
Cr H_{RL}(E,r) = J_{R'L'}(E,r) * S_{R'L',RL}
Cr S_R'L',RL = 4 pi Sum_l" C_{LL'L"} (-1)^l (-E)^(l+l'-l")/2 H_L"(E,R-R')
-----
CAUTION!: We use R to denote MT position in the primitive cell;
thus this R is R+T in our notation.
--------------------------------
```

Thus it seeminly 4π factor is missing in 52. However, righ after the subroutine `strxq` is called, we have added 4π factor as `strx(1:nlx1,ibas1,1:nlx2,ibas2) = fpi*s` in `hvcc0.m.F`. Thus Eq. 52 don't include 4π factors.

4.5 $\langle B|v|B\rangle$ part

Let us start from $\langle B|v|B\rangle$ part. For this calculation, we need structure constant, and a few types of radial integrals. With the Bloch-summed structure constant `strx` $= S^{\mathbf{k}}_{RL,R'L'}$, we

have

$$v^{\mathbf{k}}(\mathbf{r},\mathbf{r}') = \sum_{\mathbf{T}'} v(\mathbf{r},\mathbf{r}'+\mathbf{T}')e^{i\mathbf{k}\mathbf{T}'} = e^2 \sum_{L,L'} \bar{J}_L(E,\mathbf{r}-\mathbf{R}) S^{\mathbf{k}}_{\mathbf{R}L,\mathbf{R}'L'} \bar{J}_{L'}(E,\mathbf{r}'-\mathbf{R}'), \quad (54)$$

for $\mathbf{R} \neq \mathbf{R}'$. Note that we use $e^2 = 1$ (a.u.) in hvccfp0.m.F. (See note at the beginning of mkjp.F. And note the normalization check at the end of hvccfp0.m.F; $\langle \exp(i\mathbf{qr})|v|\exp(i\mathbf{qr})\rangle = 4\pi\Omega/|\mathbf{q}|^2$, where Ω is the cell volume.)

`strx calculated by "call strxq@L806:hvccfp0.m.F"` means $S^{\mathbf{k}}_{\mathbf{R}L,\mathbf{R}'L'}$. Note that 4π is mulipled right after `call strxq` to obtain `strx`. This `strx` is used in `call vcoulq_4`. `nlx1` means `(l+1)**2` for R `(=ibas1)`, `nlx2` as well.

Except the contribution for $(\mathbf{R},\mathbf{T}) = (\mathbf{R}',\mathbf{T}')$, we can evaluate $\langle B^{\mathbf{k}}_{\mathbf{R}L\mu}(\mathbf{r})|v^{\mathbf{k}}(\mathbf{r},\mathbf{r}')|B^{\mathbf{k}}_{\mathbf{R}'L'\mu'}(\mathbf{r}')\rangle$, from the
`rojb` integrals $\rho^l(B_{\mathbf{R}l\mu})$ as

$$\rho^l(B_{\mathbf{R}l\mu}) = \int_0^R r\bar{J}_l(E,r)\ rB_{\mathbf{R}l\mu}(r)dr = \frac{1}{(2l+1)!!}\int_0^R \texttt{rkpr(r)}\ \texttt{rprodx(r)}\ dr \quad (55)$$

Here $B_{\mathbf{R}l\mu}(r)$ is the radial part of $B_{\mathbf{R}L\mu}(\mathbf{r})$. This `rojb` integrals are calculated in the subrouitne `mkjb_4` in `mkjp.F`. We use radial functions `rprodox` $= rB_{\mathbf{R}l\mu}(r)$, `rkpr`$= r\bar{J}_l(E,r)(2l+1)!!$, and `rkmr`$= r\bar{H}_l(E,r)/(2l-1)!!$. (Here `rkpr` and `rkmr` are propotional to r^l and r^{-l-1} for $E=0$.)

The contribution from $(\mathbf{R},\mathbf{T}) = (\mathbf{R}',\mathbf{T}')$ should be added. This is $\mathbf{k}$-independent, and given by the
`sgbb` integral, which is also calculated in `mkjb_4`.

$$\sigma^l(B_{\mathbf{R}l\mu},B_{\mathbf{R}l\nu}) = 4\pi\int_0^R\int_0^R (r_<)\bar{J}_l(E,r_<)(r_>)\bar{H}_l(E,r_>)rB_{\mathbf{R}l\mu}(r)r'B_{\mathbf{R}l\nu}(r')drdr'$$

$$= \frac{4\pi}{2l+1}\int_0^R\int_0^R \texttt{rkpr(r}_<\texttt{)}\ \texttt{rkmr(r}_>\texttt{)}\ \texttt{rprodx(n1,r)}\ \texttt{rprodx(n2,r}'\texttt{)}\ drdr'. \quad (56)$$

With the integrals `rojb` and `sgbb`, we can calculate $\langle B|v|B\rangle$ in `vcoulq_4` as follows (`nbloch` means the total number of PB);

```
 do ibl1= 1, nbloch
   ibas1= ibasbl(ibl1)
   n1   = nbl (ibl1)
   l1   = lbl (ibl1)
   m1   = mbl (ibl1)
   lm1  = lmbl(ibl1)
   do ibl2= 1, ibl1
     ibas2= ibasbl(ibl2)
     n2   = nbl (ibl2)
     l2   = lbl (ibl2)
     m2   = mbl (ibl2)
     lm2  = lmbl(ibl2)
     vcoul(ibl1,ibl2) =
&     rojb(n1, l1, ibas1) *strx(lm1,ibas1,lm2,ibas2)
&    *rojb(n2, l2, ibas2)
     if(ibas1==ibas2 .and. lm1==lm2) then
       vcoul(ibl1,ibl2) = vcoul(ibl1,ibl2) + sgbb(n1,n2,l1, ibas1)
       ! sigma-type contribution. onsite coulomb
     endif
   enddo
 enddo
```

4.6 RL expansion of $|P^{\mathbf{k}}_{\mathbf{G}'}\rangle$

To evaluate $\langle P^{\mathbf{k}}_{\mathbf{G}}|v|P^{\mathbf{k}}_{\mathbf{G}'}\rangle$, we can use

$$\bar{P}^{\mathbf{k}}_{\mathbf{G}} \equiv \left(1 - \sum_{\mathbf{R}L}^{l\le l_{\rm Pmax}} P_{\mathbf{R}L}\right) e^{i(\mathbf{k}+\mathbf{G})\mathbf{r}}, \quad (57)$$

in the place of $P^{\mathbf{k}}_{\mathbf{G}}$ as long as we use large enough $l_{\rm Pmax}$. Here $P_{\mathbf{R}L}$ denotes the projection operator to extract the component of $\mathbf{R}L$ contribution. In fact, we use large enough $l_{\rm Pmax}$; $l_{\rm Pmax} = 2\times l_{\rm max}$`=2*LMXA`, where $l_{\rm max}$ denotes the maximum angular momentum for the expansion of eigenfunctions within MT (maximum l cutoff for $\alpha^{\mathbf{k}n}_{\mathbf{R}u}$ in Eq. (1)). In the defalut setting, we use $l_{\rm Pmax} = 8$ since we use $l_{\rm max} = 4$.

The matrix elements $\langle P_1|v|P_2\rangle$ can be calculated as

$$\langle P_1(phiphi)|v|P_2(phiphi)\rangle = \sum_{\mathbf{G_{1'}}\mathbf{G_{1''}}\mathbf{G_{2'}}\mathbf{G_{2''}}} \langle P_1(phiphi)|P_{1'}\rangle\langle P_{1'}|P_{1''}\rangle^{-1}\langle P_{1''}|v|P_{2''}\rangle\langle P_{2''}|P_{2'}\rangle^{-1}\langle P_{2'}|P_2(phiphi)\rangle, \quad (58)$$

where $1 \equiv (\mathbf{k}, \mathbf{G_1})$ and so on. Here $P_1(phiphi)$ indidates that an IPW made from a product of IPWs.The matrix elements $\langle P_{2''}|P_{2'}\rangle^{-1}$ (stored into PPOVLG,PPOVLI) and $\langle P_{2'}|P_2(phiphi)\rangle$ (stored into PPOVLGG) are given at `rdata4gw`.

4.7 Overlap matrix of PPOVL* files

PPOVL* files contains the overlap matrix of IPWs $\langle P^{\mathbf{k}}_{\mathbf{G}}|P^{\mathbf{k}}_{\mathbf{G'}}\rangle$. We have two types of $\langle P^{\mathbf{k}}_{\mathbf{G}}|P^{\mathbf{k}}_{\mathbf{G'}}\rangle$.
One is for the Coulomb matrix (PPOVLG,PPOVLI). The other is for generating $\langle G(\text{eigenfun.})G(\text{eigenfun.})|G(\text{cou})\rangle$.
(a product of IPWs of eigenfunctions can be expanded by IPWs for Coulomb matrix).

The overlap matrix elements $\langle P^{\mathbf{k}}_{\mathbf{G}}|P^{\mathbf{k}}_{\mathbf{G'}}\rangle$ are generated in `rdata4gw`. These are read and allocated in the module `m_read_ppovl (rppovl.F)` when we call `getppx2`. We have "call getppx2" in the subroutine melpln2t in ppbafp.fal.F. The melpln2t is for generating the matrix element of `<IPW psi |psi>`.

```
(1)PPOVLG + PPOVLI:
   For q in qibze(1:3,1:nqnumt) (=IBZ + Q0P points),
   number of IPWcou =ngc can be dependent of q.
   We have <k+G|k+G'>= ppovl(ngc,ngc)
   PPOVLG: G vectors as ngvecc(1:ngc).
   PPOVLI: ppovl^-1(ngc,ngc). Inverse of PPOVL0
   (PPOVL0 is unused now. It is divided into PPOVLG and PPOVLI).

   In principle the matrix element itself is k-independent,
   (just the difference of G vectors due to periodicity).
   But, for convenience, we generate them separately for each k.

(2)PPOVLGG:
  This is used for <Gphi Gphi|Gc>.
  ppovl(nggg,ngcgp) for nvggg,nvgcgp
  Range of G for nvggg is |Gc+Gp+Gp|< |Gcou|+ |Gphi|+ |Gphi|
  (triangle inequality.)
  This is only for k=0 (Thus we remove k-dependece).
  ngcgp
    QpGcutggg = (2d0+1d-2)*QpGcut_psi+QpGcut_cou+ 2d0*pi/alat*dQpG
    QpGcutgcgp= (1d0+1d-2)*QpGcut_psi+QpGcut_cou+ 2d0* 2d0*pi/alat*dQQ
  dQpG, dQQ is to enlarge range related to Q0P points.
```

4.8 $\langle P^{\mathbf{k}}_{\mathbf{G}}|v|P^{\mathbf{k}}_{\mathbf{G'}}\rangle$

To evaluate Eq. (58), we need to know its main part $\langle P^{\mathbf{k}}_{\mathbf{G}}|v|P^{\mathbf{k}}_{\mathbf{G'}}\rangle$. It is written as

$$\begin{aligned}\langle P^{\mathbf{k}}_{\mathbf{G}}|v|P^{\mathbf{k}}_{\mathbf{G'}}\rangle \approx \langle \bar{P}^{\mathbf{k}}_{\mathbf{G}}|v|\bar{P}^{\mathbf{k}}_{\mathbf{G'}}\rangle &= \langle \exp(i(\mathbf{k}+\mathbf{G})\mathbf{r})|v|\exp(i(\mathbf{k}+\mathbf{G'})\mathbf{r})\rangle - \sum_{\mathbf{R}L}\langle P^{\mathbf{k}+\mathbf{G}}_{\mathbf{R}L}|v|\exp(i(\mathbf{k}+\mathbf{G'})\mathbf{r})\rangle \\ &- \sum_{\mathbf{R'}L'}\langle \exp(i(\mathbf{k}+\mathbf{G'})\mathbf{r})|v|P^{\mathbf{k}+\mathbf{G'}}_{\mathbf{R'}L'}\rangle + \sum_{\mathbf{R}L}\sum_{\mathbf{R'}L'}\langle P^{\mathbf{k}+\mathbf{G}}_{\mathbf{R}L}|v|P^{\mathbf{k}+\mathbf{G'}}_{\mathbf{R'}L'}\rangle, \quad (59)\end{aligned}$$

where $P^{\mathbf{k}+\mathbf{G}}_{\mathbf{R}L}$ denotes the projection of PW to $\mathbf{R}L$, That is, $P^{\mathbf{k}+\mathbf{G}}_{\mathbf{R}L} \equiv P_{\mathbf{R}L}e^{i(\mathbf{k}+\mathbf{G})\mathbf{r}}$.

The first term The first term in the right-hand side of Eq. (59) is

$$\langle \exp(i(\mathbf{k}+\mathbf{G})\mathbf{r})|v|\exp(i(\mathbf{k}+\mathbf{G'})\mathbf{r})\rangle = \frac{4\pi\Omega}{|\mathbf{k}+\mathbf{G}|^2+|E|}\delta_{\mathbf{G}\mathbf{G'}}, \quad (60)$$

because we simply use $\exp(i(\mathbf{k}+\mathbf{G})\mathbf{r})$ (no prefactor for normalization) for IPW. Here E=`eee` is negative (or (almost) zero). This is coded by a line

```
  if(ig1==ig2) vcoul(ipl1,ipl2) = fpivol/(absqg2(ig1) -eee)
@subr:vroulq_4@L281:mkjp.F.
```

The second and third term In the second term, we can replace v with $\frac{4\pi\Omega}{|\mathbf{k}+\mathbf{G}'|^2+|E|}$ since v is diagonal for $|\exp(i(\mathbf{k}+\mathbf{G}')\mathbf{r})\rangle$, the third term as well. Without v, we have

$$\begin{aligned}
&\langle\exp(i(\mathbf{k}+\mathbf{G})\mathbf{r})|P^{\mathbf{k}+\mathbf{G}'}_{\mathbf{R}'L'}\rangle = \sum_{\mathbf{R}L}\langle P^{\mathbf{k}+\mathbf{G}}_{\mathbf{R}L}|P^{\mathbf{k}+\mathbf{G}'}_{\mathbf{R}'L'}\rangle \\
&= \sum_{\mathbf{R}L}(\texttt{pjyl_}(\mathbf{k}+\mathbf{G},L)\exp(i(\mathbf{k}+\mathbf{G})\mathbf{R}))^* \times R^{JJ}(|\mathbf{k}+\mathbf{G}|,|\mathbf{k}+\mathbf{G}'|,l) \\
&\times\texttt{pjyl_}(\mathbf{k}+\mathbf{G}',L)\exp(i(\mathbf{k}+\mathbf{G}')\mathbf{R}),
\end{aligned} \tag{61}$$

where we use

$$\begin{aligned}
&P^{\mathbf{k}+\mathbf{G}}_{\mathbf{R}L}(\mathbf{r}) = 4\pi i^l j_l(|\mathbf{k}+\mathbf{G}|r)Y_L(\widehat{\mathbf{k}+\mathbf{G}})Y_L(\hat{\mathbf{r}})\exp(i(\mathbf{k}+\mathbf{G})\mathbf{R}) \\
&= \texttt{pjyl_}(\mathbf{k}+\mathbf{G},L)\bar{J}_l(|\mathbf{k}+\mathbf{G}|r)Y_L(\hat{\mathbf{r}})\exp(i(\mathbf{k}+\mathbf{G})\mathbf{R}),
\end{aligned} \tag{62}$$

where $\mathbf{r}$ is measured from the center $\mathbf{R}$. Here we use `pjyl_` defined as

$$\texttt{pjyl\_}(\mathbf{k}+\mathbf{G},L) = 4\pi i^l|\mathbf{k}+\mathbf{G}|^l Y_L(\widehat{\mathbf{k}+\mathbf{G}}) \tag{63}$$

(recall the definition of $\bar{J}_l$. In codes, `cy(lm)*yl(lm)` $= Y_L(\widehat{\mathbf{k}+\mathbf{G}})$.) Search `pjyl_` in `mkjp.F`. The Bessel functions appear here in the expansion of PW; see Eq. (48). $R^{JJ}(|\mathbf{k}+\mathbf{G}|,|\mathbf{k}+\mathbf{G}'|,l)$ is given as

$$R^{JJ}(|\mathbf{k}+\mathbf{G}|,|\mathbf{k}+\mathbf{G}'|,l) = \int_0^R r^2\bar{J}_l(|\mathbf{k}+\mathbf{G}|r)\bar{J}_l(|\mathbf{k}+\mathbf{G}'|r)dr, \tag{64}$$

which can be calculated by the wronskian (`wronskj`) by the formula

$$R^{JJ}(\kappa_\mathrm{A},\kappa_\mathrm{B},l) = \int_0^R r^2\bar{J}_l(\kappa_\mathrm{A}r)\bar{J}_l(\kappa_\mathrm{B}r)dr = R^2\frac{\bar{J}_l(\kappa_\mathrm{A}r)\frac{d\bar{J}_l(\kappa_\mathrm{B}r)}{dr} - \frac{d\bar{J}_l(\kappa_\mathrm{A}r)}{dr}\bar{J}_l(\kappa_\mathrm{B}r)}{\kappa_\mathrm{A}^2-\kappa_\mathrm{B}^2}\Big|_{r=R} = -\texttt{fjj} \tag{65}$$

(`-fjj` is used in `mkjp.F`. In codes, the contributions to the second and third terms of Eq. (59) due to $\mathbf{R}L$ components are given as (simplified for illustration)

```
  fouvp_ig1_ig2 = fpi/(absqg2(ig1)-eee)                  &
        * dconjg(pjyl_(lm2,ig1)*phase(ig1,ibas2))       &
        * (-fjj(l)) * pjyl_(lm2,ig2)*phase(ig2,ibas2)
  fouvp_ig2_ig1 = fpi/(absqg2(ig2)-eee)                  &
        * dconjg(pjyl_(lm2,ig2)*phase(ig2,ibas2))       &
        * (-fjj(l)) * pjyl_(lm2,ig1)*phase(ig1,ibas2)
```

Look for the keyword `fourvp` in `mkjp.F`. Correspondences are

```
fpi --> 4 pi
absqg2(ig1) --> |q+G1|**2
-eee        --> |E|
lm2         --> L
ibas2       --> R
```

The forth term The last term of Eq. (59) can be calculated essentially the same manner with $\langle B|v|B\rangle$, where we use the Bessel function instead of $B_{\mathbf{R}l\mu}(r)$ appeared in Sec.4.5. Then we define integrals `rojp` and `sgpp` defined as (in `fpgw/gwsrc/mkjp.F`);

$$\texttt{rojp}(\mathbf{R}L) = \texttt{pjyl\_}\ \exp(i(\mathbf{q}+\mathbf{G})\mathbf{R})\rho^l(\bar{J}_l) \tag{66}$$

$$\begin{aligned}
\texttt{sgpp}(\mathbf{R}L,\mathbf{G},\mathbf{G}') = {}& \texttt{pjyl_}^*(\texttt{ig1})\exp(-i(\mathbf{q}+\mathbf{G})\mathbf{R})\texttt{pjyl_}(\texttt{ig2})\exp(i(\mathbf{q}+\mathbf{G}')\mathbf{R}) \\
&\times\texttt{radsig}, \\
&\text{where } \texttt{radsig} = \sigma^l(\bar{J}_l(|\mathbf{q}+\mathbf{G}|r),\bar{J}_l(|\mathbf{q}+\mathbf{G}'|r)).
\end{aligned} \tag{67}$$

Thus, `rojp` and `sgpp` made of coefficients for explansion and radial integral. Search `sgpp_ig1_ig2` in `mkjp.F`.

4.9 $\langle P|B\rangle$ part

$$\langle P^{\mathbf{k}}_{\mathbf{G}}|v|B^{\mathbf{k}}_{\mathbf{R}L\mu}(\mathbf{r})\rangle = \langle \exp(i(\mathbf{k}+\mathbf{G})\mathbf{r})|v|B^{\mathbf{k}}_{\mathbf{R}L\mu}(\mathbf{r})\rangle - \sum_{\mathbf{R}'L'} \langle P^{\mathbf{k}+\mathbf{G}}_{\mathbf{R}L}|v|B^{\mathbf{k}}_{\mathbf{R}L\mu}(\mathbf{r})\rangle$$

$$= \texttt{fouvb}(\mathbf{G}, \mathbf{R}L\mu) - \sum_{\mathbf{R}'L'} \texttt{rojp} * \texttt{strx} * \texttt{rojb} \tag{68}$$

The first term is stored in `fouvb(ngc,nxx,nlxx,nbas)` allocated at `L824:hvccfp0.m.F`. Search `fouvb` in `mkjp.F`. Since v is diagonal to PWs, we can evaluate this in the similar manner of Eq. (61). The second term can be evaluated from `rojb` and `sgpb` in the same manner of last section.

... xxxxx under construction xxxxx...
(we will detail a little more...)

5 Offset-Γ method; W(k=0) averaged in the Γ cell.

The offset-Γ method, originally invented for Ref.[10] by Kotani (it is described in Ref.[6]), was a key to perform accurate GW calculation in our papers. It is for the integration of $\mathbf{k}$ in Eqs.(18) and (23), where we have the integrands that diverge at $\mathbf{k} \to 0$. The original offset Γ method works well for highly symmetric systems; however, it may be problematic to apply to less symmetric systems, because the anisotropic divergence of the integrands may not be treated accurately.

Here we show an improved offset-Γ method, which treats the anisotropy of $W(\mathbf{k},\omega)$ accurately. In the followings, we use expression $W(\mathbf{k})$ for simplicity (omit subscripts and ω) instead of $W_{\mu\nu}(\mathbf{k},\omega)$, since we are concerned with the $\mathbf{k}$ integral here.

Let us give a formula for calculating $\int_{\rm BZ} f(\mathbf{k})d^3k$ using a discrete sum on $\mathbf{k}$-mesh, where $f(\mathbf{k}) = G(\mathbf{q}-\mathbf{k}) \times W(\mathbf{k})$. For the $\mathbf{k}$-mesh, we use

$$\mathbf{k}(i_1,i_2,i_3) \quad = \quad 2\pi(\frac{i_1}{N_1}\mathbf{b}_1 + \frac{i_2}{N_2}\mathbf{b}_2 + \frac{i_3}{N_3}\mathbf{b}_3),$$

where $\mathbf{b}_1, \mathbf{b}_2$, and $\mathbf{b}_3$ are the primitive reciprocal vectors (the same as the Eq.(47) in Ref.[6]). The 1st BZ is divided into $N = N_1 \times N_2 \times N_3$ microcells ($i_1 = 0, 1, ...N_1 - 1$, and also the same for i_2 and i_3.). The microcell including the Γ point is called the Γ cell [15]. The main problem is how to evaluate the contribution of the Γ cell. The divergent part of $f(\mathbf{k})$ behaves ≈ (analytic function of $\mathbf{k}$) $/(\mathbf{k}^{\rm T}\mathbf{L}\mathbf{k})$, where $\mathbf{k}^{\rm T}$ denotes the transpose of $\mathbf{k}$; $\mathbf{L}$ is a 3×3 Hermitian matrix [7]. We neglect an odd part of $\mathbf{k}$ in the above (analytic function of $\mathbf{k}$) because it has no contribution to the integral around $\mathbf{k} = 0$. Thus it is sufficient to consider the integral for $f(\mathbf{k})$ whose divergent parts behave as $f(\mathbf{k}) = \sum_L \frac{f_L Y_L(\hat{\mathbf{k}})}{|\mathbf{k}|^2}$ at $\mathbf{k} \to 0$, where l of $L \equiv (l,m)$ is restricted to be even numbers. We evaluate the integral using the formula

$$\int_{\rm BZ} f(\mathbf{k})d^3k \approx \frac{1}{N}\sum^{\mathbf{k}\neq 0} f(\mathbf{k}) + \sum_L f_L w_L + \frac{1}{N}\tilde{f}, \tag{69}$$

which is introduced in Ref.[15]. Here the weight w_L is determined in a manner as follows, so as to take into account the contributions of the divergent part of $f(\mathbf{k})$ at $\mathbf{k} \to 0$ in the Γ cell. $\tilde{f}$ is the constant part of $f(\mathbf{k})$ at $\mathbf{k} \to 0$.

To determine w_L, we can use the following procedure instead of that given in Ref.[15]. We first introduce the auxiliary function

$$F_L(\mathbf{k}) = \sum_{\mathbf{G}} \frac{\exp(-\alpha|\mathbf{k}-\mathbf{G}|^2)Y_L(\widehat{\mathbf{k}-\mathbf{G}})}{|\mathbf{k}-\mathbf{G}|^2}. \tag{70}$$

This is a generalization of an auxiliary function used in the offset-Γ method (then we only used F_{00} [6]). We usually take the $\alpha \to 0$ limit, or a sufficiently small α instead. Let us apply Eq. (69) to $F_L(\mathbf{k})$. Then we can evaluate the left-hand side of Eq. (69) exactly (the exact values are zero except for $L = (0,0)$). On the other hand, the first and third terms on the right-hand side of Eq. (69) can be evaluated numerically. In addition, we know that $f_{L'}$ for $F_L(\mathbf{k})$ is unity for $L' = L$, and zero otherwise. Thus we can determine w_L in Eq. (69) so that Eq. (69) is exactly satisfied for $F_L(\mathbf{k})$ for any L.

Let us apply Eq. (69) to $f(\mathbf{k}) = G(\mathbf{q}-\mathbf{k}) \times W(\mathbf{k})$. Then we perform an approximation taking only the most divergent term in $W(\mathbf{k})$ in addition to its analytic part. That is, we use

$$W_{\mu\nu}(\mathbf{k}) \sim \widetilde{W}_{\mu\nu}(\mathbf{0}) + \frac{4\pi}{\mathbf{k}^{\mathrm{T}}\mathbf{L}\mathbf{k}}\delta_{1\mu}\delta_{1\nu} \tag{71}$$

at $\mathbf{k} \to 0$. $\widetilde{W}_{\mu\nu}(\mathbf{0}) = 0$ for $\mu = 1$ or $\nu = 1$. See Eq.(36) in Ref.[7] to know what is neglected in the approximation of Eq. (71).

Then we finally obtain

$$\int_{\mathrm{BZ}} d^3k G(\mathbf{q}-\mathbf{k})W(\mathbf{k}) \approx \overline{\sum G(\mathbf{q}-\mathbf{k})W(\mathbf{k})}, \tag{72}$$

where its right-hand side is defined as

$$\begin{aligned}
&\overline{\sum G(\mathbf{q}-\mathbf{k})W(\mathbf{k})} \\
&\equiv \frac{1}{N}\sum_{\mathbf{k}\neq 0} G(\mathbf{q}-\mathbf{k})W(\mathbf{k}) + \frac{1}{N}G(\mathbf{q})\overline{W}(\mathbf{0}), &(73)\\
&\overline{W}(\mathbf{0}) \equiv N\sum w_L W_L + \widetilde{W}(\mathbf{0}). &(74)
\end{aligned}$$

Here $\overline{W}(\mathbf{0})$ is an average of W in the Γ cell. With this $\overline{W}(\mathbf{0})$, we can evaluate integrals just as the sum on the discrete $\mathbf{k}$-mesh. When the matrix $\mathbf{L}$ is given (a method of calculating $\mathbf{L}$ is given in the next paragraph), the non-analytic (but non-divergent) function $\mathbf{k}^{\mathrm{T}}\mathbf{L}\mathbf{k}/|\mathbf{k}^2|$ is expanded in the spherical harmonics. Then W_L is calculated for a given $\mathbf{L}$ in the manner shown in Ref.[7]. We can evaluate the accuracy of integrals with a discrete $\mathbf{k}$-mesh in combination with the approximation of Eq. (71) by calculations while changing the size of the $\mathbf{k}$-mesh.

> **[NOTE 2016-03-18:]** We now use only w_L for $L = (0,0)$. T.Kotani found strange behavior for $W(\omega)$ for the Wannier functions (for La2CuO4) when we use all w_L. Then I observed non-monotonic behavior of its real part; recall that we should see monotonic behavior as long as causality is satisfied. I found it is originated from non zero w_L for $L \neq (0,0)$. Thus we now use the formula "$w_L = 0$ for $L \neq (0,0)$" in Eq. (69).In the case of metal at $\omega = 0$, we have very large $\mathbf{L}$ (virtually infinite); thus no $1/|\mathbf{k}|^2$ behavior.

The remaining problem is how to calculate the matrix $\mathbf{L}$ in Eq. (71); there are two possible ways. One is the $\mathbf{k}\cdot\mathbf{p}$ method (perturbation) used in Ref.[7]; the other is the numerical method to calculate $\mathbf{L}$ at some $\mathbf{k}$ points near $\mathbf{k} = 0$. Here we use the latter method. Because of the point-group symmetry of the system, $\mathbf{L}$ can be expressed by the linear combination of invariant tensors μ^g_{ij} for the symmetry of the unit cell,

$$L_{ij}(\omega) = \sum_{g=1}^{N_g} a_g(\omega)\mu^g_{ij}, \tag{75}$$

where g is the index of the invariant tensor. The number of g's N_g, can be from one (cubic symmetry) through six (no symmetry). It is possible to determine the coefficient $a_g(\omega)$ from the dielectric functions $\hat{\mathbf{k}}^{\mathrm{T}}_{0i}\mathbf{L}\hat{\mathbf{k}}_{0i}$ calculated at $\{\mathbf{k}_{0i}\}$ points around $\mathbf{k} = 0$, where $\{\mathbf{k}_{0i}; i = 1, N_g\}$ is a set of the offset-Γ points. The offset-Γ points are chosen so that the conversion matrix from $\hat{\mathbf{k}}^{\mathrm{T}}_{0i}\mathbf{L}(\omega)\hat{\mathbf{k}^{0i}}$ to $a_g(\omega)$ is not numerically degenerated. The length $|\mathbf{k}^{0i}|$ can be chosen to be sufficiently enough, but avoiding numerical error as the average of $W(\mathbf{k})$ in the Γ cell. The improved offset-Γ method shown here can be applicable even to metal cases, as long as $\hat{\mathbf{k}}^{\mathrm{T}}_{0i}\mathbf{L}(\omega)\hat{\mathbf{k}_{0i}}$ contains the contribution of intraband transition.

> **[NOTE 2016-Nov:]** T.Kotani think evaluation of $L_ij(\omega)$ can be more effectively performed as for the convergence on number of $\mathbf{k}$ points. At least, without local field correction, we only need to calculate head part. For the head part, we may use large number of $\mathbf{k}$ points.

6 hx0fp0.sc.m.F. $W(\mathbf{k},\omega)$ calculation

. ... xxxxx under construction xxxxx...

7 self-energy

. ... xxxxx under construction xxxxx...

8 Fourier transformation of MTO-Hamiltonian

It is better for you to run `job_band` first. Then you have QPLIST (after 17jan2018), which contains q point list for syml.*. In addition, numbers for x-axis is shown (convenient for plotting).

When you use only MTOs, we can have the tight-binding type (real space) Hamiltonian. `job_ham` shows the procedure. Its main part is

```
mpirun -np 4 ~/ecalj/lm7K/lmf-MPIK fe --quit=band -vpwmode=0 > lmf_efermi
 ! we get efermi.lmf
mpirun -np 4 ~/ecalj/lm7K/lmf-MPIK fe --writeham --mkprocar --fullmesh -vpwmode=0
 ! we get HamiltonianMTO.(rankid) and HamitonianMTOInfo files.
cat HamiltonianMTO.* >HamiltonianMTO
 ! Merge files into a file
~/ecalj/lm7K/lmfham
```

Here, `efermi.lmf` contains the Fermi energy (determine it from the calculated energy bands for the given basis. Potential is rst.* and sigm.*). In addtion, number of electrons and magnetic moments are shown in it.

Main procedure is `~/ecalj/lm7K/lmf-MPIK fe --writeham --mkprocar --fullmesh -vpwmode=0`. Here, all the requied informations are HamiltonianMTO.* and HamiltonianMTOInfo. These files contains Hamiltonians $H_{ij}(\mathbf{k})$ and overlap matrix $O_{ij}(\mathbf{k})$. In addition, it contains info for the set of $\mathbf{k}$ points and the set of cell translational vectors $\mathbf{T}$. (And some additional info. See `lm7K/lmham.F`). Then we run `lmfham` (small program). This reads these file, and convert them to the real space representation as $H_{ij}(\mathbf{T})$ and $O_{ij}(\mathbf{T})$.

$$H_{ij}(\mathbf{T}) = \frac{1}{N}\sum_{\mathbf{k}} H_{ij}(\mathbf{k})\exp(i\mathbf{k}\mathbf{T}) \tag{76}$$

$$H_{ij}(\mathbf{k}) = \sum_{\mathbf{T}\in\mathbf{T}(i,j)} \frac{1}{n_{\mathrm{T}}} H_{ij}(\mathbf{k})\exp(-i\mathbf{k}\mathbf{T}) \tag{77}$$

Here, N is the number of $\mathbf{k}$ points. n_{T} is the degeneracy for $\mathbf{T}$. $\mathbf{T}(i,j)$ means a set of $\mathbf{T}$ for given ij (exactly speaking, a pair of atomic sites are specified from orbital index i and j. The pair of atomic sites determines the set $\mathbf{T}(i,j)$).

Sum check is

$$\sum_{\mathbf{T}\in\mathbf{T}(i,j)} \frac{1}{n_{\mathrm{T}}} = 1, \quad \frac{1}{N}\sum_{\mathbf{k}} 1 = 1. \tag{78}$$

Completeness relation for the Fourier transformation (combine Eqs.(76) and (77)) is

$$\delta_{\mathbf{k}\mathbf{k}'} = \frac{1}{N}\sum_{\mathbf{T}\in\mathbf{T}(i,j)} \frac{1}{n_{\mathrm{T}}}\exp(i(\mathbf{k}-\mathbf{k}')\mathbf{T}). \tag{79}$$

9 Fourier transformation of non-local quantity

We have "`call bloch`" in `lm7K/fp/bndfp.F`. This is for the three dimensional FFT. The usual FT is by

$$f(\mathbf{T}) = \sum_{\mathbf{k}} f(\mathbf{k}) \exp(i\mathbf{k}\mathbf{T}), \tag{80}$$

where $\{\mathbf{k}\}$ is are on the regular mesh points. The total number of its members is $N_1 \times N_2 \times N_3$. (the number is the same as that of $\{\mathbf{T}\}$). Note that we have periodicity both in $\mathbf{k}$ points and in $\mathbf{T}$ points. Because of the periodicity, the range of $\{\mathbf{k}\}$ is not unique, the range of $\{\mathbf{T}\}$ as well.

Let us think about non-local quantity which is dependent on $\mathbf{T} - \mathbf{T}'$. Then we have

$$f(\mathbf{R}\mathbf{T}, \mathbf{R}'\mathbf{T}') = \sum_{\mathbf{k}} f_{\mathbf{R}\mathbf{R}'}(\mathbf{k}) \exp(i\mathbf{k}(\mathbf{T} - \mathbf{T}')), \tag{81}$$

In practical calculations (static version of self-energy treated by `bloch` called in `fp/bndfp.F`), we first calculate $f_{\mathbf{R}\mathbf{R}'}(\mathbf{k})$ on $\mathbf{k}$ of regular mesh points. Then we need to obtain its real-space representation $f(\mathbf{R}\mathbf{T}, \mathbf{R}'\mathbf{T}')$. Because of the periodicity, we have ambiguity for the choice of possible $|\mathbf{T} - \mathbf{T}'|$. If we introduce $\bar{\mathbf{T}} = \mathbf{T} - \mathbf{T}'$, Eq. (81) is written as $f(\mathbf{R}\bar{\mathbf{T}}, \mathbf{R}'0) = \sum_{\mathbf{k}} f_{\mathbf{R}\mathbf{R}'}(\mathbf{k}) \exp(i\mathbf{k}\bar{\mathbf{T}})$ because of translational symmetry.

A reasonable choice is that we allow $\bar{\mathbf{T}}$ which satisfy $|\mathbf{R} - \mathbf{R}' + \bar{\mathbf{T}}| \leq \eta_{\mathrm{FTmax}}$. Here we should choose η_{FTmax} so that the number of alllowed $\{\bar{\mathbf{T}}\}$ is $N_1 \times N_2 \times N_3$. Howerever, it can be not possible, because of deneneracy; for the largest value of $|\mathbf{R} - \mathbf{R}' + \bar{\mathbf{T}}|$ in the alllowed $\{\bar{\mathbf{T}}\}$, we may have some of $\bar{\mathbf{T}}$ (we say degenerated). Then we need to give fractional weight for such $\bar{\mathbf{T}}$.

To get a list of $\bar{\mathbf{T}}$, we need to collect them satisfying $|\mathbf{R} - \mathbf{R}' + \bar{\mathbf{T}}| < \eta_{\mathrm{FTmax}}$. η_{FTmax} should be automatically chosen. However, in the "`bloch`" subroutine, this it too primitive yet(aug2015); we need to specify possible upper limit of "range of allowed pairs" $(\mathbf{R}\bar{\mathbf{T}}, \mathbf{R}'0)$ by hand. (`RSRNGE` in `ctrl` file). This should be fixed in future. In the current version `iaxs (=sham%iv_a_oiaxs)` contains such pair table. It is generated by `call hft2rs` in `call seneinterp` in `bndfp.F`, I think. We will have to replace "`bloch`" with better version. Pair table must be generated in a simple manner(with the technique of `getgv2` (`getgv2` is given in `fpgw/gwsrc/getgv2.F` and `lm7K/subs/pairs.F`.

10 Interpolation of the self-energy in the Brillouin zone

Here we show an interpolation procedure for giving $V^{\mathrm{xc}}_{\mathbf{k}}$ at any $\mathbf{k}$, from $V^{\mathrm{xc}}_{\mathbf{k}}$ calculated only at the regular mesh points $\mathbf{k}(i_1, i_2, i_3)$. This interpolation is used for the offset-Γ method that requires $W(\omega)$ at $\{\mathbf{k}_{0i}\}$; to calculate this $W(\omega)$, we need eigenfunctions and eigenvalues not only at the regular mesh points $\mathbf{k}(i_1, i_2, i_3)$ but also at $\mathbf{k}(i_1, i_2, i_3) + \mathbf{k}_{0i}$. This interpolation is also useful for plotting energy bands. A key point of the interpolation is that V^{xc} is expanded in real space in highly localized MTOs as follows.

At the end of step (IV) in Sec.**??**, we obtain the matrix elements $\langle \Psi_{\mathbf{k}n} | \Delta V^{\mathrm{xc}}_{\mathbf{k}} | \Psi_{\mathbf{k}m} \rangle$ on the regular mesh points of $\mathbf{k}$, where $\Delta V^{\mathrm{xc}}_{\mathbf{k}} = V^{\mathrm{xc}}_{\mathbf{k}} - V^{\mathrm{xc,LDA}}_{\mathbf{k}}$. Then it is converted to the representation in the APW and MTO bases as

$$\langle \chi^{\mathbf{k}}_a | \Delta V^{\mathrm{xc}}_{\mathbf{k}} | \chi^{\mathbf{k}}_b \rangle = \sum_{n,m} \left(z^{-1}\right)^*_{an} \langle \Psi_{\mathbf{k}n} | \Delta V^{\mathrm{xc}}_{\mathbf{k}} | \Psi_{\mathbf{k}m} \rangle z^{-1}_{bm}, \tag{82}$$

where we use the simplified basis index a, which is the index for specifying a basis ($\mathbf{R}Lj$ for MTO or $\mathbf{G}$ for APW). Thus $\chi^{\mathbf{k}}_a$ denotes the APWs or MTOs in Eq. (1); z_{na} ($\mathbf{k}$ is omitted for simplicity) denotes the coefficients of the eigenfunctions at $\mathbf{k}$, that is, $z^{\mathbf{k}n}_{\mathbf{R}Lj}$ and $z^{\mathbf{k}n}_{\mathbf{G}}$ in Eq. (1) together. This z_{an} is identified as a conversion matrix that connects eigenfunctions (band index n) and the APW and MTO bases (basis index a).

To obtain real-space representation of ΔV^{xc}, we need a representation expanded in the basis that consist of the Bloch-summed localized orbitals, which are periodic for $\mathbf{k}$ in the BZ.

However, this is not the case for the APWs in Eq. (82). To overcome this problem, we use an approximation in which we only take the matrix elements related to MTOs, that is, the elements $\langle\chi^{\mathbf{k}}_a|\Delta V^{\rm xc}_{\mathbf{k}}|\chi^{\mathbf{k}}_b\rangle$ where a and b specify MTOs. This means that the part of $\Delta V^{\rm xc}$ related to APWs is projected onto the basis of MTOs. This approximation can be reasonable as long as the main part of $\Delta V^{\rm xc}$ can be well expanded in MTOs, although we need numerical tests to confirm the accuracy as shown in Sec.**??**. Then we obtain a real-space representation of $\Delta V^{\rm xc}$ expanded in MTOs from the MTO part of $\langle\chi^{\mathbf{k}}_a|\Delta V^{\rm xc}_{\mathbf{k}}|\chi^{\mathbf{k}}_b\rangle$ by Fourier transformation. Then we can have interpolated $\Delta V^{\rm xc}$ at any $\mathbf{k}$ by inverse Fourier transformation. Since we use highly localized MTOs, this interpolation is more stable than the previous one in FP-LMTO-QSGW [6]. The complicated interpolation procedure given in Sec.II-G in Ref.[6] is no longer necessary.

To reduce the computational time, we calculate the matrix elements $\langle\Psi_{\mathbf{k}n}|\Delta V^{\rm xc}_{\mathbf{k}}|\Psi_{\mathbf{k}m}\rangle$ only up to the states whose eigenvalues are less than $E^{\Sigma}_{\rm MAX}$. Then the high energy parts of the matrix elements are assumed to be diagonal, where their values are given by a simple average of calculated diagonal elements.

11 Overview of gwsc and other scripts

The fpgw/exec/gwsc is the main script to run QSGW. After we finish one-body self-consistent calculation, we run **echo 0|lmfgw**, resulting small files. See Sec.**??**. Then we run qg4gw to generate q+G vectors stored in QGpsi,QGcou,Q0P files, in addition to EPSwklm, which is for offset-Gamma method **??**. Then we run lmfgw-MPI which is to calculate eigenfunctions and eigenvalues (and some quantities) required for successive main part of QSGW calculation. We recommend you to examine this first.

For the one-shot GW, we have another script **gw_lmfh**. For dielectric functions (and for χ^0_{+-}, we have **eps***. These are slightly different from gwsc, calling slightly different version of fortran programs. Wannier function calculations can be done by **genMLWF**, which not only generates Wannier functions (tight-binding parameters), but also W and U between Wannier functions (RPA and cRPA) together. It is in fpgw/Wannier directory.

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

In Sec. 5, we show a new improvement in the offset-Γ method, which is made in order to treat the $\mathbf{k}\to 0$ divergence of the integrand for the self-energy calculation. This improvement can correctly capture the anisotropy of the screened Coulomb interaction, although the previous offset-Γ method in FP-LMTO-QSGW [6] can be problematic for treating anisotropic systems.

In Sec. 10, we explain the interpolation procedure of $V^{\rm xc}_{\mathbf{k}}(\mathbf{r},\mathbf{r}')$. The procedure is simplified in comparison with that used in FP-LMTO-QSGW.

11.1 xxxxxxxxxxxxxxxxx, bz setting, q+G for phi and for vcoul

qg4gw-mkqg routines. QIBZ,QBZ

qibz nqbz,qibz wibz, nqibze Q0P

iq0pin mode: generate q0p: algorism

12 Used files

12.1 @MNLA_CPHI

```
m    n    l ibas
 0    1    0    1    1    1
 0    2    0    1    2    2
-1    1    1    1    3    3
 0    1    1    1    4    3
 1    1    1    1    5    3
-1    2    1    1    6    4
 0    2    1    1    7    4
 1    2    1    1    8    4
-2    1    2    1    9    5
-1    1    2    1   10    5
 0    1    2    1   11    5
```

m is a magnetic quantum number, n is the degree of freedom which means $1:\phi$, $2:\dot{\phi}$, and $3:$ local orbital. l is the orbital angular quantum number. The match of orbitals and m number is shown by **job_pdos** command.The following number is the number of atom. And the next is the numerating number. It corresponds to the number in GWinput which is the most left one in the initial conditions.

13 General cautions for developers

gwsc is the main script to perform QSGW. Sec.**??** gives an overview. Sec.**??** explain main output files. Sec. **??** explains all i/o files.

At first, note that one-body part lmv7 and fpgw are divided, mainly because of historical reasons. Make procedure is complicated, but automatic by ecalj/InstallAll.* (See. `ecalj/README.md`).

Main make system of ecalj is in ecalj/InstallAll.*. As you see in it, makefile for fpgw (GW part) is located at fpgw/gwsrc/exec/makefile. (memo: apr2015. A little too much complicated because of duplicated definition of subroutines... We need to simplify variables...).

Cautions are;

- Integrated Make system; ecalj/InstallAll.*
  For development, see ecalj/InstallAll.ifort (.gfortran) This let you know how to invoke make. The ecalj consists of three make procedure. lmv7, fpgw/exec/, fpgw/Wannier.
- Install test
  At the end of InstallAll.*, we have `make mpi_size=4 all` at ecalj/TestInstall. This is an unique way to run a series of installation tests.
- Machine dependence
  For fpgw/exec/, Machine-dependent part is given by a file such as make.inc.gfortran, which is included in the makefile by the variable PLATFORM. For lmv7, we have lmv7/MAKEINC/, where we have files which describes machine-dependences.
- CPU time and Memory measurements
  At the bottom of makefile, we have a mechanism to insert clock routines in source code. For example, hsfp0.sc.m.F is converted to `time_hsfp0.sc.m.F`, and then compiled. Time measurement is specified directive lines

  ```
  !TIME0\_number  memo
  ...
  !TIME1_number 'LABEL'
  ```

  Here memo is just comment line, LABEL is a label to identify the block. For example, see `sxcf_fal2.sc.F`. The computational time for codes sandwitched by these `!TIME0_number` and `TIME1_number` are measure, and shown at the bottom of console output file lx0 (see **gwsc** script).

See fpgw/exec/makefile to understand how to make binaries for gw part. We can make binaries by `make PLATFORM=ifort LIBMATH=-mkl` at the directory.

Other cautions for computer codes;

- We often use modules. A typical example is `use m_genallcf_v3,only: ...` . For example, see `hsfp0.sc.m.F`, which is for the calculation of $W - v$. For `call genallcf_v3` in it, all data for the `use m_genallcf_v3,only:` are allocated. Thus we can use these data after `call genallcf_v3` in the code `hsfp0.sc.m.F`.
- Methods(functions) in modules are keys to learn fpgw/ codes. For example, we have geteval(eigenvalues), readcphi(coefficient of eigenfunction for MTO part), readgeig(coefficient of APW part), get_zmel (`<phi|phi MPB>`). In cases, we have initialization routines such as `readqgcou()` defined in readeigen.F. After it is called, we can access to all date in module `m_readqgcou`. In principle, this kind of initialization routines must be called at the top of main programs... But not organized yet. In addition, it may be better to allocate even scalar in fortran2003. But such new features in fortran2003 is still buggy (at least in ifort15) as long as I tested.
- A possible mechanism to make things safer is given by a variabl `done_genallcf_v3` defined in `genallcf_mod.F`. Observe how it work in this routine. This ensures that `genallcf_v3` is called only once in a program. Thus variables in `m_genallcf_v3` has uniqueness (But we have no simple way to make write protections for them. You know a way?) In my opinion, fortran is not suitable to write long computer codes. It is better to use glue languages such as python or bash, as I did in gwsc...
- `nbas` is the number of MT sites in the primitive cell. We use `ibas` for a loop of `do ibas=1,nbas`. This is a general rule; another example is `iqbz=1,nqibz` where `nqibz` is the number of irreducible q points.
- `getkeyvalue` defined in `fpgw/gwsrc/keyvalue.F` is an universal i/o routine for GWinput. Its arguments can be one of types among "logical, int, real, int array, real array". Do grep 'call getkeyvalue' for fpgw/*/*.F to find out how to use it.
- Recently the definition of POSCAR of VASP changes. Now its CARTESIAN case is essentially the same as the standard input of ctrl file. We changed structuretools on Feb.12. 2016, following the new definition.

14 Coding rule and Developer's memo

Here is my current rule for coding. But ecalj codes alreay have long history, thus not unified in a manner. Here is my recommendations. We don't like dirty code, but simultaneously, not spend too much time for cleaning up computer code, but not too dirty logic.

- fpgw/ directory:

```
main routines are in main/*.m.F
subroutines are in gwsrc/
makefile, shell scripts are in exec/

Wannier routines (main and sub) are in Wannier/

We use fixed format f90 (or more in future).
```

- How to add new fortran file ? (dependency checker)

```
(for the case fo fpgw/ code. Essentially similar for lm7K/ part).
Because of modules of f90, we need moduledependes.inc
which describe dependency of source files given in makefile.
We have a system automatically making it by 'make init'.
Steps are:
1.Make a *.F file in gwsrc/ or main/ or Wannier/
```

```
2.Add *.o in fpgw/exec/makefile
3.Run 'make init' at fpgw/exec (or at fpgw/Wannier/ or )
  This check dependecny and moduledependes.inc,
  which is included when you run make.
  You may need to removed ../*/*.mod and/or ../*/*.o files if some erroroccurs.
4. make
```

moduledepends.inc is automatically generated by `TOOLS/checkmodule` (python code; I sometimes need to do make init). But `TOOLS/checkmodule` is not well written.

- TIME directive and makefile

```
We have
!TIME_00010 Q0P
!TIME_00010 'Q0P'
in some files such as ../main/hx0fp0.sc.m.F.
How many times and how many clock time used
is reported at the end of console output.
This reports computational time at the end of output from each node.
(see STDOUT/stdout.{rankID=0000}.* files) when you run gwsc.

At the bottom of makefile, we have conversion from
*.F to time*.F.
For exmple, hx0fp0.sc.F is converted
to time_hx0fp0.sc.F and compiled.
Here we replace directions "!TIME0" and "!TIME1..."
by a timing-measurement routine by awk.

Thus, be careful.
When you compile hx0fp0.sc.F with -g
option, it shows the error stop (such as segmentation error)
in the line number of time_hx0fp0.sc.F

If you have make error such as
> ERROR: inconsistent key, key= __x0kf_sym
, it means error when the conversion find syntax error.

You can see
>make
gawk -f script/addtime.awk  -vSTART=1 ../main/hx0fp0.sc.m.F | gawk -f script/then_separate.awk
...
This shows fpgw/exec/script/addtime.awk is used for the conversion.
```

- MPI is not so efficient yet.

```
We like to make simple MPI procedure, not nested.
For this purpose, it may be better to divide matrix elements generator
and core of GW part.

When you run gwsc or so,
 STDOUT/stdout.0000.hx0fp0_sc
contains output of hx0fp0_sc due to rank=0000.
```

- double path formalism in lmf (a problem to be cleaned up)

```
 For eigenvalues, we show twice a iteration.
 This is historical reason.
 We will improve it.
```

- emacs(vm) skills

```
multi window mode, compare files,
emacs git mode
emacs ediff mode
emacs etags
git rebase -i ==> See http://liginc.co.jp/web/tool/79390
gitk --all
python
```

- Doxygen: At ecalj/fpgw, run doxygen. Because we have Doxyfile there, we can have doxygen html and pdfs. Doxygen is not so good but not so bad for fortran. We will use doxygen for a while. But not believe doxygen too much.

```
module at the beginning of x0kf_v4h.F.
1. Not allow comment line in declearation of subrouitne.
2."double precision" is not allowed.
3. comments lines outside of subroutine.
4. To overliad doxygen bug, dummy declear needed.
   integer:: dummy4doxygen at the begining.
```

- callcaller tree generator

```
We can make a table callcaller.dat by
>make dep
at fpgw/exec/ and lm7K/ (may take one minute).
Not believe it so much...
Need to check it in other manner.
```

- Test system is at `ecalj/TestInstall/`. We can say test system is very important. We usually include new bug when you add new functionarity in a code. Test system is very critical to develop ocmputer programs quickly. Current test system is a little complicated; but we will use it for a while.

```
At ecalj/TestInstall/
We have
---------------------------------------
./Makefile
./Makefile.inc (this is called from test directories as si_gwsc).
./si_gwsc/Makefile and data for test
./crn/Makefile and data for test.
---------------------------------------
To add a test,
we keep input and output files in xxx/ directory,
and make Makefile as in the case of si_gwsc/.
In addition, you have to add the name of test in Makefile.

\item
\begin{verbatim}
Line length for fortran; Add .emacs the following three lines.
(add-hook 'fortran-mode-hook
  '(lambda ()
     (setq fortran-line-length 132)))
```

- Use fixed format of fortran. Use -132 line option. Give a line number for long do loop (not do end do)). And respect the do loop number (not delete line numbers without a reason.)
- We use

```
integer::
real(8)::
complex(8)::
```

- Supplemental documents embedded in codes should be very minimum. We have to prepare a document separately. The document(this document) explains the formulation and algorithm. Ideal code should have no comment lines; program itself should be a document corresponding to the document of formalism. (we may explain data structures in this document, but it should be very minimum).

14.1 module coding

(this section is by S.H.Ryee, a little modified). We will develop a big code with full use of modules. It is convenient to manage the code. Advantages are;

- We can easily understand, handle and modify the code without making complicated problems.
- We don't have to be careful about the order of data.
- We can easily divide a job for cooperation.

Therefore, developers are strongly recommended to make codes with modules. However, we have to be careful about how to use the modules.

A small example is contained in `ecalj/TOOLS/ModuleCodingSample/`. One can see a file named `m_test.F`. To execute the `m_test.F`, type in

```
gfortran m_test.F -I. -J. -g -ffixed-line-length-132
```

in your command line. This code is designed to read and print the file named `sample.dat` contained in the same directory. `sample.dat` is written as follows:

```
1 Ndup 3 4 8
2 Nddn 5 6 7 8
3 Cu 2 3 4 123 556 45
```

We would like to emphasize several points contained in the `m_test.F` as an example. Developers are urged to follow these points.

- Use `protected` and `private` option when delcaring variables in a module to avoid the same variable names being used outside of the module. :

```
module m_readline
   integer,protected:: nclass,nbasclassMax
   integer,protected,allocatable:: cbas(:,:),nbasclass(:)
   character(20),protected,allocatable:: classname(:)
   integer,parameter,private::maxdat=1024

   contains
   subroutine s_readclass()
   ...
```

- Use *labels* for loops to avoid confusion (for long loop). :

```
...
     do
       read(ifix,"(a)",end=999) aaa
       iline=iline+1
     end do
999  continue
     nclass=iline
     allocate(iclassin(nclass),cbastemp(maxdat,nclass),nbasclass(nclass),classname(nclass))

     rewind(ifix)
     cbastemp=-999

     do 1001, iclass=1,nclass
```

```
        read(ifix,"(a)") aaa
        read(aaa,*,end=1201) iclassin(iclass),a,(cbastemp(i,iclass),i=1,maxdat)
 1201 continue
        if(iclassin(iclass)/=iclass) call rx('iclass is not i')
        classname(iclass)=trim(a)
        do i=1,maxdat
          if(cbastemp(i,iclass)==-999) then
            nbasclass(iclass)=i-1
            exit
          endif
        enddo
 1001 continue
 ...
```

The module can be used in the main program by using *use* command:

```
...
program test
      use m_readline,only: s_readclass,  nbasclass, nclass, cbas, classname, nbasclassmax
      integer:: i,ix,iclass
      call s_readclass()

      write(*,*) '=== Read lines nclass=',nclass
      do iclass=1,nclass
        write(*,*)'output:',iclass,trim(classname(iclass)),cbas(1:nbasclass(iclass),iclass)
      enddo
end
```

- Memo for usagee of modules.
  In the PMT part (one-body part)) `ecalj/lm7K/`, we use strucrue. But no structures in the GW part `ecalj/fpgw` part. I recommned to use modules.
  Module names are `m_foobar`. Use 'only' option when you use a module. Use 'protected' for all variables in module; then these can be written only by the subroutine in the moudle.

  ```
  An example is
        module m_get_bzdata1 in getbzdata1.F
  This is related to reading BZDATA file.
  All public data defined at the head part of this module are output.
  These are set by a call as
    "call getbzdata(... arguments list ...)",
  where arguments list are all inputs.

  Thus we can have all the public data, suddenly appear
  right after "call getbzdata".

  To make the data flow clear as possible, we have to declear
  "only" option when we use a module.
  Here is an example of hx0fp0.sc.m.F, which uses a modle m_genallcf_v3.
        use m_genallcf_v3,only: genallcf_v3,
       &     nclass,natom,nspin,nl,nn,ngrp,
       &     nlmto,nlnmx, nctot,niw,nw_input=>nw,
       &     alat,ef, diw,dw,delta,deltaw,esmr,symgrp,clabl,iclass,
       &     invg, il, in, im, nlnm,
       &     plat, pos, ecore, symgg
  ---------
  In the main routine, we call genallcf_v3.
  Then all following variables are set.
  ```

- Module example.

```
To get the matrix element: zmel = <E_nu phi|phi>,
which is the parts of numerator of equations in the steps of GW calculaitons,
we use "readeigen mehanism". Let me explain this.

At first, we call
      call init_readeigen(ginv,nspin,nband,mrece)!EVU EVD are read in
      call init_readeigen2(mrecb,nlmto,mrecg)
. These are needed for initialization.
Then we do
      call get_zmelt2(exchange, ... (matrix elements generator)
in a subroutine x0kf_v4hz (which is called from main routine hx0fp0.m.F).
Then we have the matrix elements zmeltt after this call.
Because of historical reason exchange=T,F gives different names of
      zmel,  zmelt or zmeltt, which are suitable exchange calculation or
      correlaiton calculaiton.

NOTE:
get_zmelt2 internally call function readeigen (to get eigenfunctions).
```

15 Phonon project

Key papers of phonon theories are [16] [17] [18] [19]. Pratical implementations are in [7] [20].

1. The polarization function

$$\bar{\Pi} = \Pi\sqrt{v}\frac{1}{1-\sqrt{v}\Pi\sqrt{v}}\sqrt{v}\Pi \tag{83}$$

   are calculated on the regular $\mathbf{k}$ mesh points (expect $\mathbf{k}=0$), and on the offset-Gamma points (=Q0P points instead of $\mathbf{k}=0$). Expanded in the Coulomb-diagonalized MPB set $\{|E^{\mathbf{k}}_\nu\rangle\}$ as $\langle E^{\mathbf{k}}_\mu|\bar{\Pi}^{\mathbf{k}}|E^{\mathbf{k}}_\nu\rangle$. For phonon calculation, we only need to know quantities at $\omega=0$.

2. We reorganize the results of $\langle E^{\mathbf{k}}_\mu|\bar{\Pi}^{\mathbf{k}}|E^{\mathbf{k}}_\nu\rangle$ at Q0P. In is represented in the expansion of $\mathbf{k}$ near $\mathbf{k}=0$. In other words, Q0P-points method (=offset-Gamma method) is just in order to get the numerical derivative as the kdotp method.

   See Eq.36 in [7], in which we have $\mathbf{L}(\omega)$ matrix. It is given in Eq.34 in [20].

3. Bare Coulomb between ions. Calculate $Z_R Z_{R'}\frac{\partial^2 v^{\mathbf{k}}(R_\alpha - R'_\beta)}{\partial R_\alpha \partial R'_\beta}$.

4. Calculate $\langle\frac{\partial v^{\mathbf{k}}(\mathbf{r}-\mathbf{R})}{\partial R_\alpha}|E^{\mathbf{k}}_\nu(\mathbf{r})\rangle \equiv \int d^3r \frac{\partial v^{\mathbf{k}}(\mathbf{r}-\mathbf{R})}{\partial R_\alpha}E^{\mathbf{k}}_\nu(\mathbf{r})$. Note that $E^{\mathbf{k}}_{\nu=1}(\mathbf{r})$ corresponds to $\exp(i\mathbf{kr})$.

5. We have dynamical matrix

$$C^{\mathbf{k}}_{\alpha\beta} = \frac{\partial^2 W^{\mathbf{k}}(R_\alpha - R'_\beta)}{\partial R_\alpha \partial R'_\beta} \tag{84}$$

   on regular mesh points. At $\mathbf{k}=0$, we have an expansion. In stead of C, we treat $\bar{C}$ which satisfy translational symmetry.

6. We calculate dynamical matrix in the form $\bar{C} = \bar{C}^{\mathrm{N}} + \bar{C}^{\mathrm{NA}}$, where $\bar{C}^{\mathrm{N}}$ and $\bar{C}^{\mathrm{NA}}$ are analytic and non-analytic parts, respectively.

7. Non-analytic part $\bar{C}^{\mathrm{NA}}$ are specified by the Born-effective charge and the static dielectic tensor (it is in the denominator). $\bar{C}^{\mathrm{NA}}$ is given in the Bloch sum (the BZ periodicity).

8. We have analytic part $\bar{C}^{\mathrm{N}}$ Sum rule correction (sum of born effective charge, translational symmetry) may be needed.

9. Interpolarion in the whole BZ (non-analytic part and analytic part).

10. Then we can calculate phonos.

11. Calculate electron phonon coupling in the same manner.

12. Mobility calculation and so on.

16 Magnon project

Generalized Lindhard polarization function (or Kernel) is defined as (c.f.Eq. (20));

$$
\begin{aligned}
-K^{\alpha\beta} &= (\mathbf{r}_1,\mathbf{r}_2;\mathbf{r}_3,\mathbf{r}_4;\omega) = -iG_\alpha(1,3)G_\beta(4,2) = \\
&= \sum_{\mathbf{k}}^{\mathrm{BZ}}\sum_{n}^{\mathrm{occ}}\sum_{\mathbf{k}'}^{\mathrm{BZ}}\sum_{n'}^{\mathrm{unocc}} \frac{\Psi^*_{\mathbf{k}n\beta}(\mathbf{r}_2)\Psi_{\mathbf{k}n\beta}(\mathbf{r}_4)\Psi_{\mathbf{k}'n'\alpha}(\mathbf{r}_1)\Psi^*_{\mathbf{k}'n'\alpha}(\mathbf{r}_3)}{\omega-(\varepsilon_{\mathbf{k}'n'\alpha}-\varepsilon_{\mathbf{k}n\beta})+i\delta} \\
&+ \sum_{\mathbf{k}}^{\mathrm{BZ}}\sum_{n}^{\mathrm{unocc}}\sum_{\mathbf{k}'}^{\mathrm{BZ}}\sum_{n'}^{\mathrm{occ}} \frac{\Psi^*_{\mathbf{k}n\beta}(\mathbf{r}_2)\Psi_{\mathbf{k}n\beta}(\mathbf{r}_4)\Psi_{\mathbf{k}'n'\alpha}(\mathbf{r}_1)\Psi^*_{\mathbf{k}'n'\alpha}(\mathbf{r}_3)}{-\omega-(\varepsilon_{\mathbf{k}n\beta}-\varepsilon_{\mathbf{k}'n'\alpha})+i\delta}.
\end{aligned} \tag{85}
$$

(— NEED diagram FIGURE HERE for $K_{\alpha\beta}$ —)

Here $t=t_1=t_2$ and $t'=t_3=t_4$. To understand this, note the correspondence between real-time and ω space;

$$\frac{1}{\omega-\varepsilon+i\delta} \quad\leftrightarrow\quad i\theta(t-t')\exp(-i\varepsilon(t-t')) \tag{86}$$

$$\frac{1}{-\omega-\varepsilon+i\delta} \quad\leftrightarrow\quad i\theta(t'-t)\exp(-i\varepsilon(t'-t)). \tag{87}$$

It is easy to make the product $G_\alpha(1,3)G_\beta(4,2)$ in real space and real time represantaiton. This Eq. (88) is a general time-ordered linear response function for a non-interacting system. Note that this is reduced to be Eq. (20) for $\mathbf{r}_1=\mathbf{r}_2$ and $\mathbf{r}_3=\mathbf{r}_4$. This is the same as $K^{\alpha\beta}$ in Eq.(15) in Ref.[21] by Sasioglu. By the Fourier transformation of Eq. (88), we have

$$
\begin{aligned}
-K^{\alpha\beta}(\mathbf{q},\omega) &= \sum_{\mathbf{k}}^{\mathrm{BZ}}\sum_{n}^{\mathrm{occ}}\sum_{n'}^{\mathrm{unocc}} \frac{\Psi^*_{\mathbf{k}n\beta}(\mathbf{r}_2)\Psi_{\mathbf{k}n\beta}(\mathbf{r}_4)\Psi_{\mathbf{k}+\mathbf{q}n'\alpha}(\mathbf{r}_1)\Psi^*_{\mathbf{k}+\mathbf{q}n'\alpha}(\mathbf{r}_3)}{\omega-(\varepsilon_{\mathbf{k}+\mathbf{q}n'\alpha}-\varepsilon_{\mathbf{k}n\beta})+i\delta} \\
&+ \sum_{\mathbf{k}}^{\mathrm{BZ}}\sum_{n}^{\mathrm{unocc}}\sum_{n'}^{\mathrm{occ}} \frac{\Psi^*_{\mathbf{k}n\beta}(\mathbf{r}_2)\Psi_{\mathbf{k}n\beta}(\mathbf{r}_4)\Psi_{\mathbf{k}+\mathbf{q}n'\alpha}(\mathbf{r}_1)\Psi^*_{\mathbf{k}+\mathbf{q}n'\alpha}(\mathbf{r}_3)}{-\omega-(\varepsilon_{\mathbf{k}n\beta}-\varepsilon_{\mathbf{k}+\mathbf{q}n'\alpha})+i\delta}.
\end{aligned} \tag{88}
$$

When we expand eigenfunctions by atom-centered localized functions as

$$\Psi^{\mathbf{k}}_{\mathbf{k}n}(\mathbf{r}) = \sum_{\mathbf{R}i} a^{\alpha}_{\mathbf{R}i} w^{\mathbf{k}}_{\mathbf{R}i\alpha}(\mathbf{r}), (\textbf{correct? Notation check}) \tag{89}$$

in the restricted Model Hilbert space, $K^{\alpha\beta}$ is represented as $K^{\alpha\beta}_{\mathbf{R}ij,\mathbf{R}'kl}(\mathbf{q},\omega)$, where $\mathbf{R}i,\mathbf{R}j,\mathbf{R}'k,\mathbf{R}'l$ are orbital indexes in the unit cell. As we have $W=\langle \mathbf{R}i\mathbf{R}j|W|\mathbf{R}'k\mathbf{R}'l\rangle$ in the mRPA method (or cRPA), we can calculate $\chi_{+-}=K/(1-WK)$ in the model space. **(need detailed equation with indexes!)**

Spectrum funciton and Hilbert transformation

Imaginary part of $K^{\alpha\beta}(\mathbf{q},\omega)$ is obtained just by replacement $1/(\omega-\epsilon+i\delta)\to\pi\delta(\omega-\epsilon)$. That is, it is given as

$$\mathrm{Im}[K] = \sum_{\mathbf{k}}\sum_{n}\sum_{n'} M(\mathbf{k},n,\beta;\mathbf{k}+\mathbf{q},n',\alpha)\delta(\omega-(\varepsilon_{\mathbf{k}+\mathbf{q}n'\alpha}-\varepsilon_{\mathbf{k}n\beta})), \tag{90}$$

where we define the matrix element $M=\Psi^*_{\mathbf{k}n\beta}(\mathbf{r}_2)\Psi_{\mathbf{k}n\beta}(\mathbf{r}_4)\Psi_{\mathbf{k}+\mathbf{q}n'\alpha}(\mathbf{r}_1)\Psi^*_{\mathbf{k}+\mathbf{q}n'\alpha}(\mathbf{r}_3)$. By the replacement $\delta(\omega-\varepsilon)\to 1/(\omega-\varepsilon+i\delta)$ by the Hilbert transformation, we can recover real part from the $\mathrm{Im}[K]$. ω can be positive or negative; positive is for spin excitation $\beta\to\alpha$, negative is for $\alpha\to\beta$.

In the tetrahedron method of ecalj, we can calculate Eq. (90) as follows;

$$\mathrm{Im}[K]([\omega_i,\omega_{i+1}]) = \overline{\sum_{\mathbf{k}}}\sum_{n}\sum_{n'} M(\mathbf{k},n,\beta;\mathbf{k}+\mathbf{q},n',\alpha)W_{\mathrm{tet}}(i,\mathbf{k},n,\mathbf{k}+\mathbf{q},n'), \tag{91}$$

where $\overline{\sum_{\mathbf{k}}}$ is for discritized $\mathbf{k}$ points set by ecalj. $\mathrm{Im}[K]([\omega_i,\omega_{i+1}])$ means the imaginary part (weight) in the interval $[\omega_i,\omega_{i+1}]$.

I think the t2g-eg separated motion is one of the interesting theme. (need to write a little more...)

Test of magnon calculation (transversal spin fluctuation)

Look into `ecalj/MATERIALS/Fe_magnon` and `Ni_magnon`. Run a script. We have results of spin

fluctuations.

16.1 How to develop your code? → Hacking hx0fp0.m.F

For code development of magnon spin susceptibility, it will be easy to hack hx0fp0.m.F (See cleaned up version after Nov28,2019). To invoke hx0fp0, we supply an integer swith `ixc`. In principle, it has some modes, (type hx0fp0 without supplying integer). It shows three normal modes (screened Coulomb interaction on real and imag axis for all k point), and three eps modes on real axis for k point given in Q0P (supplied in GWinput as `<QforEPS>,<QforEPSL>`).

Please examine epsmode ixc=222, this is for `eps_lmfh_chipm`, which is for spin suceptibility $\langle \exp i\mathbf{q}\mathbf{r}\chi^0_{+-}|\exp(i\mathbf{r})\rangle$. (For dielectric constant see ixc=202 mode).

- Main idea of hacking is "Replace eigenvalues and eigenfuncitons" for your purpose. (For examples, we use the Wannier-function-based eigenfunctions). You can read **q** points and crystal strucrure settings in hx0fp0.
- Fermi energy. You may have to supply it via your own `readefermi`.
- `readeval` in `hx0fp0.m.F` should be replaced your own eigenvalue generators. `nband` (read from hbe.d) need to be modified.
- No core setting is needed (but this is standard in usual GWinput).
- Matrix elements must be replaced. It is in `zmel` in `x0kf_v4h`. It is generated at `call get_zmelt2`, and used in the main loop `do 25` in `x0kf_v4h.F`. Note `nmbas` is properly supplied.
- To hack it, skip eibz mode since it is confusing (use `eibzmode=F`), you may or may not recover symmetry afterwards.

16.2 Data structure of the tetrahedron method in ecalj

At first, you can consider only the `mtet=F` case. To make it safer, check `mtet=F` (if it is True, error exit by 'call rx'). In `m_tetw.F` we have `gettetwt` called from `hx0fp0.m.F` as

```
      call gettetwt(q,iq,is,isf,nwgt(:,iq),frhis,nwhis,npm,
i     qbas,ginv, ef, nqibz, nband,ekxx1,ekxx2, nctot,ecore,
i     nqbz,qbz,nqbzw,qbzw,  ntetf,idtetf,ib1bz,
i     nbmx,ebmx,mtet,eibzmode) !nov2016
```

. All arguments are input. This returns tetrahedron weight $W_{\rm tet}(i,\mathbf{k},n,\mathbf{k}+\mathbf{q},n')$ in the common data area of `m_tetw`.

input parameters

- `q`: **q** vector and index of **q** vector
- `iq`: unused for `mtet=F`. It may be safer to set -9999 for safe.
- `is,isf`: α and β
- Eigenvalues are supplied by

  ```
  do kx = 1, nqbz
    call readeval(qbz(:,kx),   is,  ekxx1(1:nband, kx) )
    call readeval(q+qbz(:,kx), isf, ekxx2(1:nband, kx) )
  enddo
  ```

  at `hx0fp0.m.F`
- `nwgt`: This is for EIBZ mode. We skip microtetrahedrons when all of `nwgt(kx0:kx3)` are zero (here `kx0,kx1,kx2,kx3` are four corners of **k** vectors). Thus we calculate tetrahedron weigh only for **k** points for `nwgw(k)=1`. In EIBZ mode, we will symmetrize finally obtained correct $K^{\alpha\beta}$. (takao: I like to check this code again...)
  Anyway, you can skip EIBZ, by eibzmode=F. Then `nwgt=1`.
- `frhis(1:1+nwhis),nwhis`: Histogram bins $[\omega_i,\omega_{i+1}]$ are `[frhis(i),frhis(i+1)]`. We set `frhis(1)=0d0`. See `m_freq.F`. We calculate values on $K^{\alpha,\beta}(\omega)$ where ω is specified by `freq_r` (this is center of the bins. See `freq_r` file.)

- If `npm=2`, we calculate negative frequency part. For negative ω, Histogram bins $[\omega_i, \omega_{i+1}]$ are `[-frhis(i),-frhis(i+1)]`. (Opposite spin flip excitation to positive energy.)
- We can set `ncore=0` (`use genallcf_v3`).
- For BZ data (`use m_read_bzdata`), you can set them by `call read_bzdata`.
- `symops(3,3,1:ngrp)` is the rotation part of space group operation. SYMOPS file is created by `echo 0|lmfgw`, and read by qg4gw. Then symops information is written into HAMindex file.

Output data sets are real(8),allocatable :: whw(:)

```
integer,allocatable:: ihw(:,:,:),nhw(:,:,:),jhw(:,:,:),ibjb(:,:,:,:)
integer:: nbnbx,nhwtot
integer,allocatable :: n1b(:,:,:),n2b(:,:,:),nbnb(:,:)
```

These are used as in `x0kf_v4h.F`. Look into it. Here is a simplified version for showing how to use these data set.

```
      do 25 jpm  = 1, npm !
      do 25 ibib = 1, nbnb(k,jpm)
        n1b(ibib,k,jpm) !band index for k
        n2b(ibib,k,jpm) !band index for q+k
        it  =  n1b(ibib,k,jpm)                ! index for n  for q
        itp =  1+  n2b(ibib,k,jpm)-n2bminimum ! index for n' for q+k
        ... M(:,:) is calculated here ...
           call get_zmelt2 --> Get zmel=<MPB psi_k|psi_{k+q}>
           zmel=dconjg(zmel)  ! -->  zmel= <psi_{k+q}| psi_k MPB_q>
           M(igb1,igb2) = dconjg(zmel(igb1,it,itp))* zmel(igb2, it,itp)
           Here M = <M_igb1 psi_it | psi_itp> < psi_itp | psi_it M_ibg2 >
        do iw = ihw(ibib,k,jpm),ihw(ibib,k,jpm)+nhw(ibib,k,jpm)-1
           imagweight = whw(jhw(ibib,k,jpm)+iw-ihw(ibib,k,jpm))
           Kmatrix(:,:,iw,jpm) = Kmatrix(:,:,iw,jpm) + M(:,:)*imagweight
        enddo
25    continue
```

We do $\sum_n \sum_{n'}$ in Eq. (90) as follows; before this `do 25`, we set $\mathbf{k}, \mathbf{k}+\mathbf{q}$. k is index for $\mathbf{k}$ vector. `n2bminimum` is introduced (this is for unoccupied states for `npm=1`. $W_{\rm tet}$ is given as `imagweight`, `iw` is the index for the histogram bin.

16.3 Requirement of the spin symmetic Hamiltonian

(this idea is moved to another text modelSX.pdf.)

I think we can set up a model spin-symmetric screened exchange energy, which is consistent with the linear response theory. Thus we have two ways, one is calculate Heisenberg's J by the difference of total energy. The other is by the linear response theory as in the magnon project.

17 Usual tetrahedron method

The standard tetrahedron method is used for determining the Fermi energy and so on in `ecalj/lm7K/*`. Probably, easiest way to know its usage is in the PDOS mode (try `job_pdos`), which is calculated in `lmf-MPIK`, mainly in `ecalj/lm7K/fp/bndfp.F`.
In `bndfp.F`, search `--- Loop over tetrahedra ---`, There the `itet` loop is for the tetrahedron sum. See

```
call slinz(wt,eigen,eminp+ef0,emaxp+ef0,pdosalla(1,isp,ichan,ibas),ndos)
```

in the loop. This is to accumelate `pdosalla(1:ndos,....)`. where energy window `[eminpu+ef0:emaxp+ef0]` is divided by `ndos`. `wt` is the matrix element times weight for the tetrahedron (usually no dependence for tetrahedron). `eigen(1:4)` is the four corner of eigenvalues given as `eigen(1:4) = evlall(ib,isp,idtete(1:`
You can see the tetrahedron is originally generated by the `subroutine tetirr`.

18 Wannier project

One of my idea is in newwannier.pdf (request to me).

1. Non orthgonalized basis
2. Acurately remove double counting.
3. Limitation of FLEX, TPSC

19 Paralellization project

Appendix (touched at 2022jan)

A Harris-Foulkner energy and Kohn-Sham energy

In LDA/GGA, on the way to self-consistency, input density and output density are not the same (not self-consitent). Thus we define two total energy for given input density $n^{\rm in}$, the Harris-Foulkner energy $E_{\rm Harris}$ and the Hohenberg-Kohn energy $E_{\rm HoKohn}$

$$\begin{aligned} E_{\rm Harris} &= E_{\rm k}^{\rm core} + E_{\rm B} - V_{\rm es}[n^{\rm Zc}+n^{\rm in},\mathbf{R}_a]\cdot n^{\rm in} - V_{\rm xc}[n^{\rm c}+n^{\rm in}]\cdot n^{\rm in} \\ &\quad + E_{\rm es}[n^{\rm Zc}+n^{\rm in},\mathbf{R}_a] + E_{\rm xc}[n^{\rm c}+n^{\rm in}], \qquad (92) \\ E_{\rm HoKohn} &= E_{\rm k}^{\rm core} + E_{\rm B} - V_{\rm es}[n^{\rm Zc}+n^{\rm in},\mathbf{R}_a]\cdot n^{\rm out} - V_{\rm xc}[n^{\rm c}+n^{\rm in}]\cdot n^{\rm out} \\ &\quad + E_{\rm es}[n^{\rm Zc}+n^{\rm out},\mathbf{R}_a] + E_{\rm xc}[n^{\rm c}+n^{\rm out}], \qquad (93) \\ E_{\rm B} &= \sum_p^{\rm occupied} \alpha_p^{i*}\langle F_i|H^{\rm in}|F_j\rangle\alpha_p^j, \qquad (94) \end{aligned}$$

See Eqs.(B.1),(B.2)in Ref.[20]. These are slightly wrong because Eq.(31) for $E_{\rm xc}$ is wrong; it is not the functional of total charges (inlcuding nucleus) $n^{\rm Zc}+n^{\rm in}$, but the total electron density $n^{\rm c}+n^{\rm in}$. Based on the 3-component formalism in Ref.[20], $E_{\rm es}$ have explicit dependendence on atomic positions $\mathbf{R}_a$, but $E_{\rm xc}$ does not. The dependence is via the mutipole transformation in Eq.(14). Search `ham_ehf,ham_ehk` in ecalj/lm7K/fp/bndfp.F. The $E_{\rm Harris}$ is given at `call m_mkehkf_etot1(sev, eharris)`, while $E_{\rm HoKohn}$ is given by
`call m_mkehkf_etot2(sev,sumtv, eksham)`. $E_{\rm Harris}$ and $E_{\rm HoKohn}$ are shown in `save.*` file.

The band energy $E_{\rm B}$ is stored in the module variable in bndfp.F; this is shown as `sev` in `save.*`. The kinetic energy $E_{\rm k}^{\rm core}+E_{\rm B}-V_{\rm es}[n^{\rm Zc}+n^{\rm in},\mathbf{R}_a]\cdot n^{\rm out}-V_{\rm xc}[n^{\rm c}+n^{\rm in}]\cdot n^{\rm out}$ contained in the $E_{\rm HoKohn}$ is calculated as `sumtv` in bndfp.F-mkekin.F

In LDA/GGA calculations by lmf-MPIK, `save.*` file contains a line per iteration.

```
c ehf(eV)=-15730.0982239 ehk(eV)=-15730.0982222 sev(eV)=-15.7737393
```

shows $E_{\rm Harris}$ =-15730.0982239 eV and $E_{\rm HoKohn}$ =-15730.0982222 eV, as well as the valence band energy = −15.7737393 eV. In principle, $E_{\rm Harris}$ and $E_{\rm HoKohn}$ should be exactly the same when converged; the difference is the numerical error. `c` at the begining of line means "converged". `h` means the 1st iteration from `atm.*` file (superposition of atomic density).

B Block inversion used for dielectric functions and downfolding

See Christph's and Pick's paper

$$\begin{pmatrix} P & Q \\ R & S \end{pmatrix}\begin{pmatrix} W & -WQS^{-1} \\ -S^{-1}RW & S^{-1}+S^{-1}RWQS^{-1} \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \qquad (95)$$

where P and S are square matrices, and

$$W = (P - QS^{-1}R)^{-1}. \qquad (96)$$

We refer $X = -WQS^{-1}$ and $Y = -S^{-1}RW$.

Proof:

$$\begin{aligned} &(1,1)\text{ component} = PW - QS^{-1}RW = (P-QS^{-1}R)W = 1, \qquad (97) \\ &(1,2)\text{ component} = -PWQS^{-1} + QS^{-1} + QS^{-1}RWQS^{-1} \\ &= (-PW + 1 + QS^{-1}RW)QS^{-1} = (-(P-QS^{-1}R)W+1)QS^{-1} = 0 \qquad (98) \\ &(2,1)\text{ component} = RW - SS^{-1}RW = 0, \qquad (99) \\ &(2,2)\text{ component} = -RWQS^{-1} + SS^{-1} + SS^{-1}RWQS^{-1} = 1 \qquad (100) \end{aligned}$$

(I think this proof is a little too complicated).

C Downfolding

Downfolding is a general concept which often appears in physics of varieties of contexts. It is based on the Block inversion Eq. (95). This appears for the inversion of one-body problem, or divide the Fock space for many-body theory. (divide the Fock space into two Hilbert spaces; one-particle excited states and states with more than one-particles).

For exaple, the Green function is the inversion of the matrix $\omega - H$, where H is divided into to

$$\begin{pmatrix} H_{11} & H_{12} \\ H_{21} & H_{22} \end{pmatrix}. \tag{101}$$

If we have $H_{12} = H_{21} = 0$, we have $G^0_{11} = 1/(\omega - H_{11})$ as the main part of Green function. This is completely separated from from the residual part $G^0_{22} = 1/(\omega - H_{22})$.

When H_{12} and H_{21} are non-zero, we have to take into their effect by perturbation, or by the block inversion of Eq. (95). Then we have

$$\begin{pmatrix} G_{11} & G_{12} \\ G_{21} & G_{22} \end{pmatrix} = \begin{pmatrix} G_{11} & -G_{11}H_{12}G^0_{22} \\ -G^0_{22}H_{21}G_{11} & G^0_{22} + G^0_{22}H_{21}G_{11}H_{12}G^0_{22} \end{pmatrix}, \tag{102}$$

where W in Eq. (95) is G_{11}. Note that the ω dependence is in G^0_{11} and G^0_{22}. Here G_{11} given as (See Eq. (96)).

$$G_{11} = \frac{1}{\omega - H_{11} - H_{12}G^0_{22}H_{21}}. \tag{103}$$

Then $H_{12}G^0_{22}H_{21}$ is identified as the self-energy.

There are possible cases about how to choose Hilbert space Ω_1 and Ω_2 corresponding to the division Eq. (101).

- In the one-body problem, for example, we take Ω_1 as low energy part and Ω_2 as high energy part. or 3d parts and others.
- In a case of many-body theory, Ω_1 is the one-particle Fock space. Then G_{11} is the one-body propagetor, and G_{22} is many-body (two- and more particles-) propagetor.
- We may take Ω_1 as the model space, Ω_2 as the residual space. Then we make like to identify H_{11} as the Hubbard Hamiltonian. However, it is not so simple since H_{11} contains screening effect due to the degree of freedom of Ω_2. We have to consider a little more complicated downfolding procedure.

Warn: not be confused with the division of many-body Hamiltonian H into $H_0 + (H - H_0)$ for perturbation.

C.1 Causality and analytic property

The equation $(\omega - H)G(\omega) = 1$ is not well defined. We need to consider original equation of motion in real time as $(i\frac{\partial}{\partial t} - H)G(t - t') = \delta(t - t')$ where we have to take into account the bondary condition of retarded response responding to the source term of impulse at the right-hand side.

Causality means "cause gives result". This is represented by the step function, for example, as $\theta(t - t')\exp(i\omega_0(t - t'))$, whose FT gives $1/(\omega - \omega_0 - i\delta)$. Thus the position of pole (upper or lower plane) is important to determine the direction of time (real time representation).

Sum rule is related to the causality but a little different. For example, sum rule for Imaginary part of G_{11} is controlled only by the beheavior $G_{11}(\omega)$ at $|\omega| \to \infty$. Thus, as long as $H_{12}G^0_{22}H_{21} \to 0$ for $|\omega| \to \infty$, the sum rule is satisfied.

D Spherical Harmonics and Real harmonics in ecalj

In ecalj, fortran codes are mainly based on the real harmonics $y_{lm}(\hat{\mathbf{r}})$, instead of the usual sperical (complex) harmonics $Y_{lm}(\hat{\mathbf{r}})$. The coefficients of eigenfunctions and so on are ordered as, e.g. $(m = -2, m = -1, m = 0, m = 1, m = 2)$ for $l = 2$. For example, `LMXA=4`, we have `(4+1)**2=25` harmonics, ordered as $y_{00}, y_{-11}, y_{01}, y_{11}, y_{-22}, y_{-12}, ..., y_{22}, y_{-33}, ...y_{33}, y_{-44}, ...y_{44}$.

$y_{lm}(\hat{\mathbf{r}})$ is defined from $Y_{lm}(\hat{\mathbf{r}})$. Here $\hat{\mathbf{r}} = (\theta, \phi)$.

$$y_{l0}(\hat{\mathbf{r}}) \equiv Y_{l0}(\hat{\mathbf{r}}). \tag{104}$$

$$y_{lm}(\hat{\mathbf{r}}) \equiv \frac{1}{\sqrt{2}}[(-1)^m Y_{lm}(\hat{\mathbf{r}}) + Y_{l-m}(\hat{\mathbf{r}})]. \tag{105}$$

$$y_{l-m}(\hat{\mathbf{r}}) \equiv \frac{1}{\sqrt{2}i}[(-1)^m Y_{lm}(\hat{\mathbf{r}}) - Y_{l-m}(\hat{\mathbf{r}})]. \tag{106}$$

, where $m > 0$. Equivalently,

$$Y_{l0}(\hat{\mathbf{r}}) \equiv y_{l0}(\hat{\mathbf{r}}). \tag{107}$$

$$Y_{lm}(\hat{\mathbf{r}}) \equiv \frac{(-1)^m}{\sqrt{2}}[y_{lm}(\hat{\mathbf{r}}) + iy_{l-m}(\hat{\mathbf{r}})]. \tag{108}$$

$$Y_{l-m}(\hat{\mathbf{r}}) \equiv \frac{1}{\sqrt{2}}[y_{lm}(\hat{\mathbf{r}}) - iy_{l-m}(\hat{\mathbf{r}})]. \tag{109}$$

.

The definition of $Y_{lm}(\hat{\mathbf{r}})$ are

$$Y_{lm}(\theta, \phi) = (-1)^m \left[\frac{(2l+1)(l-m)!}{4\pi(l+m)!}\right]^{\frac{1}{2}} P_l^m(\cos(\theta)) e^{im\phi}, \tag{110}$$

$$P_l^m(x) = \frac{(1-x^2)^{m/2}}{2^l l!} \frac{d^{l+m}}{dx^{l+m}}(x^2-1)^l \tag{111}$$

.

We take these definitions from
(1)A.R.Edmonds, Angular Momentum in quantum Mechanics, Princeton University Press, 1960,
(2)M.E.Rose, Elementary Theory of angular Momentum, John Wiley & Sons, INC. 1957,
if necessary. The definition of spherical hermonics are the same in these books.

E Wannier function and SOC

At 2022-5-25, we add the SOC matrix calculation for the Wannier function.

E.1 Generate the Wannier functions by projection

It might be better to generate Wannier functions by projection. This means no optimization steps in the maximally localized Wannier procedure. To do the projection, we set `wan_maxit_1st 0`, `wan_maxit_2nd 0` in `GWinput`. One of the big advantage without optimization is that the obtained Wannier functions can keep the crystal symmetry.

We select orbitals at the section `<worb>` in `GWinput`. Corresponding to chosen numbers (if $4f$ is chosen, we have 10 11 ... 16.), we have initial functions $G_{l\sigma}(r)y_{lm}(\hat{\mathbf{r}})$. Here we use real spherical functions $y_{lm}(\hat{\mathbf{r}})$ given in Eq. (106). Outside of MuffinTin(MT), $G_{l\sigma}(r)$ is given by the Gaussian as $G_{l\sigma}(r) = \frac{1}{N}\exp(-(r/r_0)^2)$ where r_0 is fixed to be 2.0 a.u. Search `r0g = 2d0` in `hpsig_MPI.F`. Here we omit atom index for simplicity. Within MT, $G_{l\sigma}(r)$ is given by the linear combination of ϕ and $\dot{\phi}$ as $c_1\phi(r) + c_2\dot{\phi}(r)$. Note that $\phi(r)$ and $\dot{\phi}(r)$ are the solutions of the radial Schrödinger equation. Coefficients c_1 and c_2 are determined so as to match with the Gaussian for value and slope at the MT boundary, see `subroutine getc1c2` in `hpsig_MPI.F`. Note this can be spin-dependent (if `--phispinsym` is used, we can use spin-independent G_l).

Then we can calculate the projection matrix $\langle\psi_{\mathbf{k}n\sigma}|G_{l\sigma}(r)y_{lm}(\hat{\mathbf{r}})\rangle$. Here bands $\psi_{\mathbf{k}n\sigma}$ are in the outer window specified by `wan_out_emax` and `wan_out_emin`. (we assume the case that eigenfunctions are spin diagonal here). The projection is calculated in `hpsig_MPI.F`. (minor point: $\langle\psi_{\mathbf{k}n}$(tail part in MT)$|G_{l\sigma}(r)y_{lm}(\hat{\mathbf{r}})\rangle$ is neglected.)

With the projection, we can make projected Wannier functions as

$$W_{lm\sigma}(\mathbf{r}) = \sum_{\mathbf{k}n} |\psi_{\mathbf{k}n\sigma}(\mathbf{r})\rangle\langle\psi_{\mathbf{k}n\sigma}|G_{l\sigma}(r)y_{lm}(\hat{\mathbf{r}})\rangle \tag{112}$$

Pay attention to the width of the energy window (outer window). If it is very wide, $\sum_{\mathbf{k}n} |\psi_{\mathbf{k}n}(\mathbf{r})\rangle\langle\psi_{\mathbf{k}n}|$ becomes identical matrix; thus no character of eigenfunctions remains. If narrow enough (e.g. only seven bands for $4f$), the Wannier completely include the character of the eigenfunctions; but with the penalty of longer range. The Wannier functions are not uniquely determined. So it might be useless to say something too much. "Maximally localized" is not necessarily so meaningful (or has limited meanings).

FYI: Recall two step optimization procedure of Maximally localized Wannier functions (MLWF). The 1st step is picking up seven degree of freedom for all $\mathbf{q}$ points (4f case). The 2nd step is the unitary transformation within the degree of freedom. Its principle is the minimization of Wannier spread. Cons are (1)Crystal symmetry might be broken, (2) Tail of Wannier can be oscillating. In addition, we have some ambiguity; if outer window become wider and wider, MLWF can be more and more localized.

E.2 SOC matrix for the Wannier functions

We calculate SOC matrix `hammsoc` $\langle F_{\mathbf{k}i\sigma}|H_{\rm SOC}|F_{\mathbf{k}j\sigma}\rangle$ in `sugw.F`
(`lmf-MPIK --jobgw --job --socmatrix` mode around the end of `genMLFhso`) for all $\mathbf{k}$ points in the Brillowin zone. Here $F_{\mathbf{k}i\sigma}$ means the basis functions in the PMT method. Eigenfunction is given as

$$\Psi_{\mathbf{k}n\sigma} = \sum_i F_{\mathbf{k}i\sigma}\alpha^{\mathbf{k}}(i,n,\sigma). \tag{113}$$

Thus the required SOC matrix $\langle W_{lm\sigma}(\mathbf{r})|H_{\rm SOC}|W_{lm'\sigma'}(\mathbf{r})\rangle$ (`hammso` in `hsocmat.F`) is calcualted in `hsocmat` from

1. $\alpha^{\mathbf{k}}(i,n,\sigma)$ (`evec(1:nz,iband,iqbz,isp)` in `hsocmat.F`)
2. $\langle F_{\mathbf{k}i\sigma}|H_{\rm SOC}|F_{\mathbf{k}j\sigma'}\rangle$ (`hso(1:nz,1:nz,iqbz,ispc)` in `hsocmat.F`. `ispc=3` means spin off-diagonal)
3. $\langle\psi_{\mathbf{k}n\sigma}|G_{l\sigma}(r)y_{lm}(\hat{\mathbf{r}})\rangle$ (`dnk(iband,iwf,iqbz,isp)` in `hsocmat.F`)

See `hsocmat.F`. We have `evecw=matmul(evec,dnk)`. From `hammso`, we calculate the trace of the square of $H_{\rm SOC}$ as $\mathrm{Tr}(H^2_{\rm SOC})$ in the space spanned by the Wannier functions. By construction the trace is not dependent on y_{lm} or on Y_{lm} because projected fourteen Wannier functions (when $4f$) spans the same Hilbert space. The trace is equilannt to the sum of square of eigenvalues of $H_{\rm SOC}$.

$\mathrm{Tr}(H^2_{\rm SOC})$, as well as $\langle W_{lm\sigma}|H_{\rm SOC}|W_{lm'\sigma'}\rangle$, can be Wannier dependent because the Wannier is not unique. But we expect the dependence is not so large.

Important check point to evaluate the size of SOC is whether we can reproduce the SOC splitting in the original bands (not the bands of the 14-space model but the bands of full QSGW).

F Crystal symmetry for GW calculation

space group operation (space group rotation):

In the preparation state of GW calculation, we have 3x3 matrices `symops(3,3,ig),ig=1,ngrp` for point group part of the space group operations. It is written in `call m_hamindex_init,m_hamindex.F:L128`) at `lmv7.F` (lmfgw-MPIK job=0 mode). For example, in the case of GaAs, it shows 24 matrices in it.

In GW main routines such as `hsfp0.sc.m.F`, we do `call genallcf_v3`. This set `symgg (=symops)` in the module `m_genallcf_v3` as well as `invg` pointing the . In the module `m_zmel.F`, we do `call mptauof`. This set all required symmetry operations (find accompanied translation vector and atom-site mapping information) from point group operations. It gives informations of space group symmetry (which atom is mapped to which atom, and so on). See document at the begining of `subroutine mptauof`.

In `lmv7.F`(main routine), we have `m_hamindex` called for lmfgw mode=0. This gives a file Hamindex, containing Hamindex.

The space group symmetry operation can be checked by "lmchk" in advance. But be careful. It also includes inversion symmetry, which means $|\psi_{-\mathbf{k}}(\mathbf{r})|^2 = |\psi_{\mathbf{k}}(\mathbf{r})|^2$ (this is because of time-reversal symmetry as for real local $V(\mathbf{r})$ (DFT case)). Need attention as for LDA+U case of orbital moments appears (no time-reversal case).

The given crystal symmetry can be controlled by `SYMGRP` category in `GWinput`. You can supply generators of space group (see explanation in GWinput), or set `find`. If it is `find`, we assume electronic structure has the symmetry of crystal structure. Note that symmetry of electronic structure can be lower than the crystal symmetry. For example, NiO is deformed along (111) direction in realitiy, but we may like to use structure without such deformation for calcualtions. In such a case, we may need to supply generators for lower symmetry.

AF symmetry:
Recently, we add a new option `SYMGRP_AF`. We have to set

```
/home/usr2/h70252a/ecaljAFtest/NdTEST
~/SWJ/Nd2Cu04
...
    ATOM=Ndup  POS=  0.0000  {a}/2   {c}/2-{znd}*{c}  AF=1
    ATOM=Nddn  POS=  {a}/2   0.0000  {znd}*{c}       AF=-1
SYMGRPAF r2x
SYMGRP r4z::(-1/2,1/2,0)
...
```

Here, AF=1,AF=-1 give AF pairs. SYMGRP is to keep z axis. SYMGRPAF is an antiferro magnetic symmetry operation (we add r2x+spin inversion), then the ATOM AF=1 is mapped to AF=-1. Ask to T.Kotani as for AF symmetry setting, because it is relatively new.

G IBZ and EIBZ scheme

We include EIBZ procedure given in III.G. in Ref.[7]. It is for the polarization function `hx0fp0,hx0fp0_sc` and for self-energy `hsfp0_sc`.

IBZ
Recall IBZ first. We have a set of space-group operation $S_A = \{A_i | i = 1, ...N_A\}$ (except translation by reciprocal vectors). A_i is represented by space rotation and translation. This S_A can include time-reversal operation. $\mathbf{q}' = A_i(\mathbf{q})$ belongs to BZ. (Simple multiplication of rotation matrix to $\mathbf{q}$ may (or may not) need pulling back to (1st) BZ). For a set of mesh points "$\{\mathbf{q}\}$ in the BZ", A_i gives an one-to-one mapping between them. "$\{\mathbf{q}\}$ in IBZ" is a sub set of 1stBZ.

Consider a set of functions dependent of $\mathbf{q}$ as $\{g_{\mathbf{q}}(\mathbf{r}, n)\}$. We can calculate X defined as a sum of $F[g_{\mathbf{q}}]$ as

$$X = \sum_{\mathbf{q}\in \mathrm{BZ}} F[g_{\mathbf{q}}] = \sum_{A_i \in S_A} \sum_{\mathbf{q}\in \mathrm{IBZ}} \frac{N_{\mathbf{q}}}{N_A} F[g_{A_i(\mathbf{q})}] \tag{114}$$

Here IBZ is a set of $\mathbf{q}$ from which we can generate all the mesh points in the BZ. $N_{\mathbf{q}}$ is the number of mesh points generated from $\mathbf{q}$ by S_A (the number of the set $\mathbf{q}^*$). For $\mathbf{q} = 0$, $N_{\mathbf{q}} = 1$. For general $\mathbf{q}$ (lowest symmetric points), $N_{\mathbf{q}} = N_A$.

If $A_i[g_{\mathbf{q}}] = g_{A_i(\mathbf{q})}(\mathbf{r}, n)$ is satisfied, we have

$$X = \sum_{\mathbf{q}\in \mathrm{BZ}} F[g_{\mathbf{q}}] = \sum_{A_i \in S_A} \sum_{\mathbf{q}\in \mathrm{IBZ}} \frac{N_{\mathbf{q}}}{N_A} F[g_{A_i(\mathbf{q})}] = \sum_{A_i \in S_A} \sum_{\mathbf{q}\in \mathrm{IBZ}} \frac{N_{\mathbf{q}}}{N_A} F[A_i[g_{\mathbf{q}}]]. \tag{115}$$

This shows how we evaluate X only from $\{g_{\mathbf{q}}(\mathbf{r}, n) | \mathbf{q} \in \mathrm{IBZ}\}$.
(NOTE: eigenfunctions calculated by diagonalization satisfy $A_i[g_{\mathbf{q}}] = g_{A_i(\mathbf{q})}(\mathbf{r}, n)$ except phase factor. We consider cases that the phase factor is irrelevant.)

EIBZ
We consider sub group of S_A with keeping given $\mathbf{k}$. This sub group is written as $S_A^{\mathbf{k}} = \{A_i^{\mathbf{k}} | i =$

$1, 2, ..., N_A^{\mathbf{k}}\}$. For this sub group, we apply the same logic of IBZ, resulting

$$X = \sum_{\mathbf{q}\in \mathrm{BZ}} F[g_{\mathbf{q}}] = \sum_{A_i^{\mathbf{k}}\in S_A^{\mathbf{k}}} \sum_{\mathbf{q}\in \mathrm{EIBZ}(\mathbf{k})} \frac{N_{\mathbf{q}}^{\mathbf{k}}}{N_A^{\mathbf{k}}} F[A_i^{\mathbf{k}}[g_{\mathbf{q}}]] \tag{116}$$

Here EIBZ($\mathbf{k}$) is a set of $\mathbf{q}$ from which we can generate all the mesh points in the BZ. $N_{\mathbf{q}}^{\mathbf{k}}$ is the number of mesh points generated from $N_{\mathbf{q}}$ by $S_A^{\mathbf{k}}$. Note that $\mathbf{k}$ is fixed in Eq. (116). Since $S_A^{\mathbf{k}} \subset S_A$, IBZ $\subset$ EIBZ($\mathbf{k}$). That is, equivalent q poinst in IBZ might be diffrenciated in EIBZ($\mathbf{k}$).

See Eq.(50) in Ref.[7]. `call Seteibz -->call eibzgen` returns $S_A^{\mathbf{k}}$ stored in the modele `m_eibz`.

Note normazization is

$$N = \sum_{\mathbf{q}} 1 = \frac{1}{N_A^{\mathbf{k}}} \sum_{A_i^{\mathbf{k}}\in S_A^{\mathbf{k}}} \sum_{\mathbf{q}\in \mathrm{EIBZ}(\mathbf{k})} N_{\mathbf{q}}^{\mathbf{k}} = \sum_{\mathbf{q}\in \mathrm{EIBZ}(\mathbf{k})} N_{\mathbf{q}}^{\mathbf{k}} \tag{117}$$

For poralization

We apply Eq. (116) to $F^{\mathbf{k}}[g_{\mathbf{q}}]$, resulting

$$X(\mathbf{k}) = \frac{1}{N_A^{\mathbf{k}}} \sum_{A_i^{\mathbf{k}}\in S_A^{\mathbf{k}}} \left(\sum_{\mathbf{q}\in \mathrm{EIBZ}(\mathbf{k})} N_{\mathbf{q}}^{\mathbf{k}} F^{\mathbf{k}}[A_i^{\mathbf{k}}[g_{\mathbf{q}}]] \right). \tag{118}$$

This means that we have to calculate weighted average only for $g_{\mathbf{q}} \in$ EIBZ($\mathbf{k}$), and then finally symmetrized (symmetrization is in `call x0kf_v4hz_symmetrize` in `hx0fp0.sc.m.F,hx0fp0.m.F` or newly developing `hrcxq`.

We apply Eq. (118) to Eq.(22) in Ref.[7] resulting Eq.(51) in Ref.[7]. Then we need to rotate MPB for given space group rotation $A_i^{\mathbf{k}}$. Note that $A_i^{\mathbf{k}}$ keeps $\mathbf{k}$ of MPB. This is in the next section.

Note that `call eibzgen` stores not only $S_A^{\mathbf{q}}$(`eibzsym`), but also the weight $N_{\mathbf{q}}^{\mathbf{k}}$ (`nwgt`) in the module `m_eibz`. (role of $\mathbf{q}$ and $\mathbf{k}$ may be turn around in `m_eibz`)

For self-energy

In `hsfp0.sc.m.F`(not in `hsfp0.m.F`), we have symmetrization as Eq.(52) in Ref.[7]. But used formula is a little different.

We use Eq. (117) as it is. As in the same manner of poralization function, we call `eibzgen` at first. To obtain weighted sum of self-energy at $\mathbf{q}$, we calculate weigted sum for $\mathbf{k} \in$ EIBZ($\mathbf{q}$). (note the sum is controlled by `nrkip` which is the weights copied from `nwgt` (MPI parallelization is used). When `nrkip=0`, we skip corresponding do loop.)

Symmetrizer is `call zsecsym`, called from the main program of `hsfp0.sc.m.F`. Its core part is

```
  zsect(ii1:ie1,ii2:ie2)= zsect(ii1:ie1,ii2:ie2)
&   + matmul( dconjg(transpose(rmatjj(1:ne1,1:ne1,iblk1))),
&     matmul(zsec(ii1:ie1,ii2:ie2,iqxx),
&     rmatjj(1:ne2,1:ne2,iblk2)) )
```

. This is the self-energy rotated by the rotation matrix `rmatjj`, which is generated by `call rotwvigg` in it. (NOTE: we carefully treat degenerated bands to keep symmetry well. It makes block decomposition(due to degeneracy) of self-energy matrix. A block is `zsect(ii1:ie1,ii2:ie2)`, where band indeces `ii1:ie1` are degenerated bands).

H Rotation of eigenfuncitons and MPB by the space-group operations

H.1 space-group rotation of PMT eigenfunction

We sometimes have to map eigenfuncions by space group operations.

An space-group operation is specified by (g, Δ), which contains a 3×3 space-rotaiton matrix g together with a translation vector Δ. Recall that $\Delta = 0$ for the symmorphic cases.

This (g, Δ) makes a rotation $\mathbf{r} \to \mathbf{r}'$ as

$$\mathbf{r}' = g(\mathbf{r}) + \Delta, \tag{119}$$

which is equilalent to

$$\mathbf{r} = g^{-1}(\mathbf{r}') + \Delta^{-1}, \tag{120}$$

$$\Delta^{-1} = -g^{-1}(\Delta) \tag{121}$$

Further, we calculate which atomic site is mapped to which atomic site for give (g, Δ). It is given as

$$g(\mathbf{R}) + \Delta = \mathbf{R}' + \Delta\mathbf{T}_{\mathbf{R}}. \tag{122}$$

In `ecalj`, the array `miat(ipos=1,npos)` gives $\mathbf{R}'$ in the primitve cell. `tiat(3,ipos)` gives the translation of multiples of primitive cell vectors $\Delta\mathbf{T}_{\mathbf{R}}$. Here $\mathbf{R}$ is the atom in the unit cell.

The rotation of an function $F \to F'$ by (g, Δ) is defined so that $F'(\mathbf{r}') = F(\mathbf{r})$. Here $F'(\mathbf{r}')$ is written as $g[F](\mathbf{r}')$. Thus we have

$$g[F](\mathbf{r}) = F(g^{-1}(\mathbf{r}) + \Delta^{-1}) \tag{123}$$

Let us apply this to the bloch sum function $A^{\mathbf{k}}_{\mathbf{R}uL}(\mathbf{r}) \equiv \sum_{\mathbf{T}} A_{\mathbf{R}uL}(\mathbf{r} - \mathbf{R} - \mathbf{T}) \exp(i\mathbf{k}\mathbf{T})$ in Eq. (4). (Here we get the angular momentum index L. u is for radial part of index.) Then we have

$$\begin{aligned}
g[A^{\mathbf{k}}_{\mathbf{R}uL}](\mathbf{r}) &= A^{\mathbf{k}}_{\mathbf{R}uL}(g^{-1}(\mathbf{r}) + \Delta^{-1}) = \sum_{\mathbf{T}} A_{\mathbf{R}uL}(g^{-1}(\mathbf{r}) + \Delta^{-1} - \mathbf{R} - \mathbf{T}) \exp(i\mathbf{k}\mathbf{T}) \\
&= \sum_{\mathbf{T}} A_{\mathbf{R}uL}(g^{-1}(\mathbf{r} + g(\Delta^{-1}) - g(\mathbf{R}) - g(\mathbf{T}))) \exp(i\mathbf{k}\mathbf{T}) \\
&= \sum_{\mathbf{T}} \sum_{L'} A_{\mathbf{R}uL'}(\mathbf{r} + g(\Delta^{-1}) - g(\mathbf{R}) - g(\mathbf{T})) D_{L'L}(g) \exp(i\mathbf{k}\mathbf{T}) \\
&= \sum_{\mathbf{T}} \sum_{L'} A_{\mathbf{R}'uL'}(\mathbf{r} - \mathbf{R}' - \Delta\mathbf{T}_{\mathbf{R}} - g(\mathbf{T})) D_{L'L}(g) \exp(i\mathbf{k}\mathbf{T}) \\
&= \sum_{\mathbf{T}''} \sum_{L'} A_{\mathbf{R}'uL'}(\mathbf{r} - \mathbf{R}' - \mathbf{T}'') D_{L'L}(g) \exp(ig(\mathbf{k})\mathbf{T}'') \exp(-ig(\mathbf{k})\Delta\mathbf{T}_{\mathbf{R}}) \\
&= \sum_{L'} A^{g(\mathbf{k})}_{\mathbf{R}'uL'}(\mathbf{r}) D_{L'L}(g) \exp(-ig(\mathbf{k})\Delta\mathbf{T}_{\mathbf{R}}),
\end{aligned} \tag{124}$$

where we use $A_{\mathbf{R}'uL}(\mathbf{r}) = A_{\mathbf{R}uL}(\mathbf{r})$ since R and R' sites should be equilalent sites. We use cancellation $\Delta = -g(\Delta^{-1})$. $D_{L'L}(g)$ is the rotaion matrix of the real spherical harmonics for g.

H.2 space-group rotation of PMT eigenfunction

We have eigenfunction rotation routine `subroutine rotwvigg` called from `zsecsym.F`.
MTO part:
With this Eq. (124), coeffecients of eigenfunctions for the MTO part of eigenfunctions are rotated (mapped) by the space group operation g. It is coded in `subroutine rotwvigg`(eigenfunction rotation routine) as

```
...
 phase = [(exp(-img2pi*sum(qtarget*tiat(:,ibas,igg))),ibas=1,nbas)]
 do iorb=1,norbmto !orbital-blocks are specified by ibas, l, and k.
   ibas = ibastab(iorb)
   l    = ltab(iorb)
   k    = ktab(iorb)
   init1 = offl(iorb)+1
   iend1 = offl(iorb)+2*l+1
   init2 = offlrev(miat(ibas,igg),l,k)+1
   iend2 = offlrev(miat(ibas,igg),l,k)+2*l+1
   evecout(init2:iend2,:)= matmul(dlmm(-l:l,-l:l,l,igg),evec(init1:iend1,:))*phase(ibas)
 enddo
...
```

Here

`ndimh = nlmto + napw`: the size of PMT Hamiltonian
`evec(1:nlmto,1:nband)` : the coeffecients of eigenfunctions on MTO
`igg`: index specifying the space-group operation g
`qtarget`: qtarget=$g(\mathbf{k})$
`ibas`: site index within the primitive cell.
`iorb` is the index of MTO part of block ($2l+1$ elements for $m=-l,-l+1,...,l$).
`l` is the angularmomentum index. l of L.
`k` is the radial index.
`miat`: site index ibas is mapped to `miat(ibas,igg)`
`tiat`: $\Delta\mathbf{T_R}$
`offl`: offset for the `iorb` block
`offlrev`: offset for the (ibas,l,k) block
The Block `init1:iend1` is mapped to the block `init2:iend2`.

In order to call rotwvigg, we have to call `readhamindex()` so as to set up variables `ltab,ktab,offl,oflrev,...` in `m_hamindex` module in advance.

Orbital index tables (`*_tbl` is read from a file `@MNLA_CPHI` (See Sec.12.1) by `readmnla_cphi`. The last column in `@MNLA_CPHI` contains the orbital-block index. The orbital-blocks are specified by l and k, its size is $2l+1$. The number of i is exactly the number of lines of `@MNLA_CPHI`.

APW part:
In `subroutine rotwvigg`(eigenfunction rotation routine) we also have part for the APW. Formula is

$$\begin{aligned} & g[\exp(i(\mathbf{k}+\mathbf{G})\mathbf{r})] = \exp(i(\mathbf{k}+\mathbf{G})(g^{-1}(\mathbf{r})+\Delta^{-1})) = \exp(i(g(\mathbf{k})+g(\mathbf{G}))(\mathbf{r}+g(\Delta^{-1})) \\ & = \exp(i(g(\mathbf{k}+\mathbf{G})\mathbf{r})\exp(i(\mathbf{k}+\mathbf{G})\Delta^{-1}) \\ & = \exp(i(\bar{g}(\mathbf{k})+g(\mathbf{G})+(g(\mathbf{k})-\bar{g}(\mathbf{k})))\mathbf{r})\exp(-i(\mathbf{k}+\mathbf{G})\Delta). \end{aligned} \tag{125}$$

(Our definition of PW (IPW) has 'no nomalization factor'.) Here $g(\mathbf{k})$ is pulled back to be $\bar{g}(\mathbf{k})$ in the 1stBZ. Their difference is $g(\mathbf{k})-\bar{g}(\mathbf{k})$. We define `barG` as $\bar{G}=g(\mathbf{G})+(g(\mathbf{k})-\bar{g}(\mathbf{k}))$. Thus it is coded as
(here we use $\mathbf{q}$ instead of $\mathbf{k}$, sorry for confusion).

```
...
  igg: index for space group operation.
nlmto: the size of MTO part.

  ikt  = getikt(q)       !index for q
  ikt2 = getikt(qtarget) !index for bargq = \bar{g}(q)
  do ig = 1,napw_in !number of APW
    qpg  = q + matmul( qlat(:,:),igv2(:,ig,ikt))  ! q+G
    qpgr = matmul(symops(:,:,igg),qpg)            !g(q+G)
    nnn= nint(matmul(platt,qpgr-qtarget)) !integer sets for barG= g(G)+ g(q)-bargq
    ig2 = ngvecprev(nnn(1),nnn(2),nnn(3))   ! index for barG
    evecout(nlmto+ig2,:)= evec(nlmto+ig,:) * exp( -img2pi*sum(qpgr*shtvg(:,igg)) )
  enddo
...
```

H.3 space-group rotation of MTIPW expansion of eigenfunction

In ecalj, we re-expand eigenfuncitons in the manner of MTIPW form, that is, MTpart+IPWpart as in Eq. (8) in the GW calculations. In the form of MTIPW, we have to map eigenfunctions by space-group operations. As in the manner of `rotwvigg`, we have `rotmto` and `rotipw`. These are for rotation MTpart and IPW parts.

H.4 space-group rotation of Mixed Product Basis

We need all the matrix elements $\langle \Psi_{\mathbf{q}+\mathbf{k}n'} | \Psi_{\mathbf{k}n} E^{\mathbf{q}}_{\nu} \rangle$. for given inputs of $\mathbf{q}, \mathbf{k}, n, n'$. Here $E^{\mathbf{q}}_{\nu}$ is the Coulomb-diagonalized mixed product basis [7].

In the procedure of EIBZ (See Eq. (118) around), we have to rotate MPB. Module `module m_rotMPB2 (in m_rotMPB.F)` is for generating the rotation matrix `zrotm`. With `zrotm`, we rotate MPB in `call X0kf_v4hz_symmetrize`, where we multiple conversion matrix `\zcousq` for conversion from MPB to $\{\} E^{\mathbf{q}}_{\nu}\}$ as

`rcxq(:,:,iw,jpm) = matmul(zcousqc,matmul(rcxq_core,zcousq))` (see code in `x0kf_v4hz_symmetrize`).

`zrotm` is generated by calling `rotmto2,rotipw2`. These are essentially similar with `rotmto,rotipw`, but the basis set in the MT region, and the basis set for the interstitial region are different from the case of eigenfunctions. Thus we supply `ibas_tbl` and so on as

```
 use m_pbindex,only: norbt, ibas_tbl,l_tbl,k_tbl,offset_tbl,offset_rev_tbl,
&     max_ibas_tbl,max_l_tbl,max_k_tbl,max_offset_tbl
```

in `module m_rotMPB2`.

H.5 space-group rotation of Hamiltonian (obsolate)

Let us consider the rotation of Hamiltonian. For (g, Δ), we have $\langle A^{\mathbf{k}}_i | H | A^{\mathbf{k}}_j \rangle = \langle g[A^{\mathbf{k}}_i] | H | g[A^{\mathbf{k}}_j] \rangle$ because Hamiltonian unchanged by (g, Δ). Thus we have

$$\langle A^{\mathbf{k}}_i | H | A^{\mathbf{k}}_j \rangle = \langle g[A^{\mathbf{k}}_i] | H | g[A^{\mathbf{k}}_j] \rangle = \sum_{i'j'} \langle A^{g(\mathbf{k})}_{i'} z_{i'i} | H | A^{g(\mathbf{k})}_{j'} z_{j'j} \rangle = \sum_{i'j'} z^*_{i'i} \langle A^{g(\mathbf{k})}_{i'} | H | A^{g(\mathbf{k})}_{j'} \rangle z_{j'j}$$

$$= z^\dagger \langle A^{g(\mathbf{k})}_{i'} | H | A^{g(\mathbf{k})}_{j'} \rangle z \tag{126}$$

Thus we have

$$\langle A^{g(\mathbf{k})}_{i'} | H | A^{g(\mathbf{k})}_{j'} \rangle = z \langle A^{\mathbf{k}}_i | H | A^{\mathbf{k}}_j \rangle z^\dagger \tag{127}$$

I xxxxxxxxxxx MPI parallelization

The MPI parallelizaiton is still very limited. MPI in the current version is controlled by "rankdivider" in `hsfp0.sc.m.F,hx0fp0.sc.m.F,hx0fp0.m.F`. In GW calculations, we have to take sum for some types of indexes. And some indexes are external ones. The rankdivider try to divide these indexes to groups and pass these regions to each cores. Each cores sum up for given regions only. And then we do take total sum to obtain final results before symmetrization.(I think symmetrization may be too time consuming, it may be bette to skip EIBZ?).

I think we need to extend the rank-divider idea for better MPI parallelization.

J real space sphere integral

See `module m_SphericalDesignInt`. With the spherical t-design, we can perform real-space integral for a sphere. It is very useful for debugging.

K Expansion of non-local functions, need fixing xxxxxxxxx

... xxxxx under construction xxxxx...

We expand the Coulomb interaction $v(\mathbf{r},\mathbf{r}') = e^2/|\mathbf{r}-\mathbf{r}'|$ as

$$\begin{cases} v(\mathbf{r},\mathbf{r}') = \dfrac{1}{N_c}\displaystyle\sum_{\mathbf{k}}\sum_{IJ} \tilde{M}_I^{\mathbf{k}}(\mathbf{r}) v_{IJ}(\mathbf{k}) \{\tilde{M}_J^{\mathbf{k}}(\mathbf{r}')\}^* \\ v_{IJ}(\mathbf{k}) = \dfrac{1}{N_c}\displaystyle\int_V d^3r \int_V d^3r' \{M_I^{\mathbf{k}}(\mathbf{r})\}^* v(\mathbf{r},\mathbf{r}') M_J^{\mathbf{k}}(\mathbf{r}') \end{cases} \tag{128}$$

This expansion is general for the two-point non-local functions. However, for convenience, we expand the polarization function D as

$$\begin{cases} D(\mathbf{r},\mathbf{r}',\omega) = \dfrac{1}{N_c}\displaystyle\sum_{\mathbf{k}}\sum_{IJ} M_I^{\mathbf{k}}(\mathbf{r}) D_{IJ}(\mathbf{k},\omega) \{M_J^{\mathbf{k}}(\mathbf{r}')\}^* \\ D_{IJ}(\mathbf{k},\omega) = \dfrac{1}{N_c}\displaystyle\int_V d^3r \int_V d^3r' \{\tilde{M}_I^{\mathbf{k}}(\mathbf{r})\}^* D(\mathbf{r},\mathbf{r}',\omega) \tilde{M}_J^{\mathbf{k}}(\mathbf{r}') \end{cases} \tag{129}$$

and the dielectric function ϵ (and also the inverse dielectric function ϵ^{-1}) as

$$\begin{cases} \epsilon(\mathbf{r},\mathbf{r}',\omega) = \dfrac{1}{N_c}\displaystyle\sum_{\mathbf{k}}\sum_{IJ} \tilde{M}_I^{\mathbf{k}}(\mathbf{r}) \epsilon_{IJ}(\mathbf{k},\omega) \{M_J^{\mathbf{k}}(\mathbf{r}')\}^* \\ \epsilon_{IJ}(\mathbf{k},\omega) = \dfrac{1}{N_c}\displaystyle\int_V d^3r \int_V d^3r' \{M_I^{\mathbf{k}}(\mathbf{r})\}^* \epsilon(\mathbf{r},\mathbf{r}',\omega) \tilde{M}_J^{\mathbf{k}}(\mathbf{r}'). \end{cases} \tag{130}$$

L Expansion of a plane wave with the mixed basis, need fixing

... xxxxx under construction xxxxx...

If we substitute a plane wave $e^{i\mathbf{k}\cdot\mathbf{r}}/\sqrt{\Omega}$ for $F^{\mathbf{k}}(\mathbf{r})$ in Eq.(28), we have

$$\begin{cases} \dfrac{1}{\sqrt{\Omega}} e^{i\mathbf{k}\cdot\mathbf{r}} = \displaystyle\sum_J M_J^{\mathbf{k}}(\mathbf{r}) \tilde{C}_J^{\mathbf{k}0} \\ \tilde{C}_J^{\mathbf{k}0} = \dfrac{1}{\sqrt{\Omega}} \displaystyle\int_\Omega \{\tilde{M}_J^{\mathbf{k}}(\mathbf{r})\}^* e^{i\mathbf{k}\cdot\mathbf{r}} d^3r. \end{cases} \tag{131}$$

For small $\mathbf{k}$, the maximum eigenvalue of the Coulomb matrix should be $v(\mathbf{k}) \equiv 4\pi e^2/|\mathbf{k}|^2$ and the corresponding eigenvector shoud be equal to $\tilde{C}_J^{\mathbf{k}0}$. So we can get $\tilde{C}_J^{\mathbf{k}0}$ from the eigenvalue problem instead of evaluating the integral of Eq.(131).

In `hvccfp0.m.f`, we get the maximum eigenvalue $\epsilon^0(\mathbf{k})$ and corresponding eigenvector $\tilde{C}_J^{\mathbf{k}0}$ from

$$\sum_J [v_{IJ}(\mathbf{k}) - \epsilon^0(\mathbf{k}) O_{IJ}^{\mathbf{k}}] \tilde{C}_J^{\mathbf{k}0} = 0. \tag{132}$$

Then we check the normalization

$$\sum_{IJ} (\tilde{C}_I^{\mathbf{k}0})^* O_{IJ}^{\mathbf{k}} \tilde{C}_J^{\mathbf{k}0} = 1 \tag{133}$$

and calculate the two quantities

$$\texttt{v(exact)} = \Omega \frac{4\pi e^2}{|\mathbf{k}|^2}, \tag{134}$$

$$\texttt{v(cal)} = \Omega \sum_{IJ} (\tilde{C}_I^{\mathbf{k}0})^* v_{IJ}(\mathbf{k}) \tilde{C}_J^{\mathbf{k}0} = \Omega \epsilon^0(\mathbf{k}), \tag{135}$$

which are shown in the end of the output of `hvccfp0.m.f` (`lvcc` by the script `gw_lmf` or `eps_lmf`) such as follows.

```
--- vcoul(exact)=  0.166657D+05 absq2=  0.5565111898526868D-01
--- vcoul(cal ) =  0.166587D+05 -0.484112D-19
```

You can see the agreement is good enough! The quantity $\tilde{C}_J^{\mathbf{k}0}$ is stored into `Mix0vec`. It is read into the variable `gbvec` in `hx0fp0.m.f`. We also store the next quantity;

$$\begin{aligned} C_J^{\mathbf{k}0} &\equiv \frac{1}{\sqrt{\Omega}} \int_\Omega \{M_J^{\mathbf{k}}(\mathbf{r})\}^* e^{i\mathbf{k}\cdot\mathbf{r}} d^3r \\ &= \sum_I \{O_{IJ}\}^* \frac{1}{\sqrt{\Omega}} \int_\Omega \{\tilde{M}_I^{\mathbf{k}}(\mathbf{r})\}^* e^{i\mathbf{k}\cdot\mathbf{r}} d^3r \\ &= \sum_I O_{JI} \tilde{C}_I^{\mathbf{k}0}. \end{aligned} \tag{136}$$

It is read into the variable `zzr` in `hx0fp0.m.f`.

M ... *xxxx under construction xxxx...*
(Usuda's old note from here)

... xxxx under construction xxxx...
In this note, we denote the primitive lattice vector as $\{\mathbf{a}_i | i=1,2,3\}$ (`=alat*plat(1:3,i)`), the volume of unit cell as $\Omega = |\mathbf{a}_1 \times \mathbf{a}_2 \cdot \mathbf{a}_3|$, and the reciprocal lattice vector as $\{\mathbf{b}_i | i=1,2,3\}$ (`=2*pi*qlat(1:3,i)/alat`).

We assume the periodic boundary condition for quantities as $\Psi(\mathbf{r}) = \Psi(\mathbf{r} + N_1\mathbf{a}_1) = \Psi(\mathbf{r} + N_2\mathbf{a}_2) = \Psi(\mathbf{r} + N_3\mathbf{a}_3)$. Correspondingly, we use a Brillouin zone (BZ) discrete mesh, which is given as

$$\mathbf{k}(i_1, i_2, i_3) = 2\pi\left(\frac{i_1}{N_1}\mathbf{b}_1 + \frac{i_2}{N_2}\mathbf{b}_2 + \frac{i_3}{N_3}\mathbf{b}_3\right) \tag{137}$$

for $i_1 = 0,1,2,...N_1-1$ and so on. Within the volume $V = \Omega N_\mathrm{c} = \Omega N_1 N_2 N_3$, we normalize eigenfunctions and so on. However, it is rather convenient to use the normalization within a unit cell Ω because we know the property

$$\int_V F^{\mathbf{k}}(\mathbf{r}) G^{\mathbf{k}'}(\mathbf{r}) d^3r = \delta_{\mathbf{k}\mathbf{k}'} N_\mathrm{c} \int_\Omega F^{\mathbf{k}}(\mathbf{r}) G^{\mathbf{k}'}(\mathbf{r}) d^3r \tag{138}$$

for any functions $F^{\mathbf{k}}$ and $G^{\mathbf{k}'}$ with the Bloch periodicity specified by $\mathbf{k}$ and $\mathbf{k}'$. In the GW code, we store the cell-normalized eigenfunction $\tilde{\Psi}^{\mathbf{k}n}(\mathbf{r})$ to DATA4GW;

$$\tilde{\Psi}^{\mathbf{k}n}(\mathbf{r}) \equiv \sqrt{N_\mathrm{c}}\Psi^{\mathbf{k}n}(\mathbf{r}) \tag{139}$$

$$\int_\Omega |\tilde{\Psi}^{\mathbf{k}n}(\mathbf{r})|^2 d^3r = 1. \tag{140}$$

This $\tilde{\Psi}^{\mathbf{k}n}(\mathbf{r})$ is expanded as

$$\tilde{\Psi}^{\mathbf{k}n}(\mathbf{r}) = \sum_{au} \alpha_{au}^{\mathbf{k}n} A_{au}^{\mathbf{k}}(\mathbf{r}) + \sum_{\mathbf{G}} \beta_{\mathbf{G}}^{\mathbf{k}n} P_{\mathbf{G}}^{\mathbf{k}}(\mathbf{r}), \tag{141}$$

$$A_{au}^{\mathbf{k}}(\mathbf{r}) \equiv \sum_{\mathbf{T}} A_{au}(\mathbf{r} - \mathbf{R}_a - \mathbf{T}) e^{i\mathbf{k}\cdot\mathbf{T}}, \tag{142}$$

$$\begin{aligned} P_{\mathbf{G}}^{\mathbf{k}}(\mathbf{r}) &\equiv 0 \quad \text{if } \mathbf{r} \in \text{any MT} \\ &\equiv e^{i(\mathbf{k}+\mathbf{G})\cdot\mathbf{r}} \quad \text{otherwise,} \end{aligned} \tag{143}$$

where $A_{au}^{\mathbf{k}}(\mathbf{r})$ is the Bloch sum of the atomic function $A_{au}(\mathbf{r})$ in the a-site muffin-tin (MT) sphere. $P_{\mathbf{G}}^{\mathbf{k}}(\mathbf{r})$ denotes the interstitial plane wave (IPW). Here $\mathbf{T}$ is the lattice translation vector; $\mathbf{R}_a$ is the position of the a-site in the cell; $\mathbf{G}$ denotes the reciprocal vector; u denotes the index to specvectorify the argumentaion basis. $A_{au}^{\mathbf{k}}(\mathbf{r})$ is orthnormlized as

$$\int_{|\mathbf{r}|<V_a} A_{au}(\mathbf{r}) A_{au'}(\mathbf{r}) d^3r = \delta_{uu'}, \tag{144}$$

where V_a is the size of the a-site MT. The normalization is

$$\frac{1}{N_\mathrm{c}} \int_V \{A_{au}^{\mathbf{k}}(\mathbf{r})\}^* A_{a'u'}^{\mathbf{k}'}(\mathbf{r}) d^3r = \delta_{\mathbf{k}\mathbf{k}'}\delta_{aa'}\delta_{uu'} \int_\Omega |A_{au}^{\mathbf{k}}(\mathbf{r})|^2 d^3r = \delta_{\mathbf{k}\mathbf{k}'}\delta_{aa'}\delta_{uu'} \tag{145}$$

$$\frac{1}{N_\mathrm{c}} \int_V \{P_{\mathbf{G}}^{\mathbf{k}}(\mathbf{r})\}^* P_{\mathbf{G}'}^{\mathbf{k}'}(\mathbf{r}) d^3r = \delta_{\mathbf{k}\mathbf{k}'} \int_\Omega \{P_{\mathbf{G}}^{\mathbf{k}}(\mathbf{r})\}^* P_{\mathbf{G}'}^{\mathbf{k}'}(\mathbf{r}) d^3r = \delta_{\mathbf{k}\mathbf{k}'} \int_\Omega P_{\mathbf{G}'-\mathbf{G}}^{\mathbf{0}}(\mathbf{r}) d^3r \tag{146}$$

N ... *xxxx under construction xxxx...*
Dielectric function

N.1 Dielectric function without local-field correction

... xxxx under construction xxxx...
Approximating $\epsilon^{-1}(\mathbf{q},\omega)$ as $1/\epsilon(\mathbf{q},\omega)$ corresponds to neglecting the local-field correction.

$\epsilon(\mathbf{q},\omega)$ is given as

$$\begin{aligned}\epsilon(\mathbf{q},\omega) &= \frac{1}{V}\int_V d^3r \int_V d^3r' e^{-i\mathbf{q}\cdot\mathbf{r}}\epsilon(\mathbf{r},\mathbf{r}',\omega)e^{i\mathbf{q}\cdot\mathbf{r}'} \\ &= 1-\frac{1}{V}\int_V d^3r \int_V d^3r' \int_V d^3r'' e^{-i\mathbf{q}\cdot\mathbf{r}}e^{i\mathbf{q}\cdot\mathbf{r}'}v(\mathbf{r},\mathbf{r}'')D(\mathbf{r}'',\mathbf{r}',\omega) \\ &= 1-v(\mathbf{q})D(\mathbf{q},\omega), \end{aligned} \tag{147}$$

where the relation

$$\int_V v(\mathbf{r},\mathbf{r}'')e^{-i\mathbf{q}\cdot\mathbf{r}}d^3r = v(\mathbf{q})e^{-i\mathbf{q}\cdot\mathbf{r}''} \tag{148}$$

is used and

$$v(\mathbf{q}) = \sum_{IJ}(\tilde{C}_I^{\mathbf{q}0})^* v_{IJ}(\mathbf{q})\tilde{C}_J^{\mathbf{q}0}, \tag{149}$$

$$D(\mathbf{q},\omega) = \sum_{IJ}(C_I^{\mathbf{q}0})^* D_{IJ}(\mathbf{q},\omega)C_J^{\mathbf{q}0}. \tag{150}$$

In `hx0fp0.m.f`, we calculate $v(\mathbf{q})$, $D(\mathbf{q},\omega)$ and $\epsilon(\mathbf{q},\omega)$ by

```
vcmean = sum( dconjg(gbvec) * matmul(vcoul,gbvec) )
x0mean = sum( dconjg(zzr)   * matmul(zxq(:,:,iw),zzr))
         eps(iw,iqixc2) = 1- vcmean * x0mean
```

and the inverse dielectric funcion is given by `1/eps(iw,iqixc2)` . The matrix element of the polarization, $D_{IJ}(\mathbf{q},\omega)$ = `zxq`, is obtained from the subroutine `x0kf`. The results of Re(ϵ), Im(ϵ), Re(ϵ^{-1}) and Im(ϵ^{-1}) are stored in `EPS01.nolfc.dat` .

N.2 Dielectric function with local-field correction

... xxxx under construction xxxxx...
The inverse dielectric function $\epsilon^{-1}(\mathbf{q},\omega)$ is calculated as follows:

$$\begin{aligned}\epsilon^{-1}(\mathbf{q},\omega) &= \frac{1}{V}\int_V d^3r \int_V d^3r' e^{-i\mathbf{q}\cdot\mathbf{r}}\epsilon^{-1}(\mathbf{r},\mathbf{r}',\omega)e^{i\mathbf{q}\cdot\mathbf{r}'} \\ &= \sum_{IJ}\left\{\frac{1}{\sqrt{\Omega}}\int_\Omega \tilde{M}_I^{\mathbf{q}}(\mathbf{r})e^{-i\mathbf{q}\cdot\mathbf{r}}d^3r\right\}\epsilon_{IJ}^{-1}(\mathbf{q},\omega)\left\{\frac{1}{\sqrt{\Omega}}\int_\Omega \{M_J^{\mathbf{q}}(\mathbf{r}')\}^* e^{i\mathbf{q}\cdot\mathbf{r}'}d^3r'\right\} \\ &= \sum_{IJ}(\tilde{C}_I^{\mathbf{q}0})^*\epsilon_{IJ}^{-1}(\mathbf{q},\omega)C_J^{\mathbf{q}0}. \end{aligned} \tag{151}$$

In `hx0fp0.m.f`, we calculate $\epsilon^{-1}(\mathbf{q},\omega)$ by

```
epsi(iw,iqixc2) = sum( dconjg(gbvec) * matmul(zw0, zzr) )
```

and the dielectric function is given by `1/epsi(iw,iqixc2)` . The matrix element of $\epsilon_{IJ}^{-1}(\mathbf{q},\omega)$ = `zw0` is obtained from the subroutine `wcf`. The results of Re(ϵ), Im(ϵ), Re(ϵ^{-1}) and Im(ϵ^{-1}) are stored in `EPS01.dat` .

O *... xxxx under construction xxxx...*

memo

```
ESEAVR (average of sigm at high energy)

Rotation of q by space group (not unique if q is on the BZ boundary).
Discontinuity of bands at BZ boundary

Mechanism of GW calculation for Metal. Drude weight.

Tetrahedron method. Accumulation of imaginary part, and Hilbert transformation. No time-reversal sym

Rseq,Broryden mixing,Anderson mixing (Yellow note by okuda).

zmelt: unified matrix elements generator m_zmel.F

structure constant:

conversion between spherical harmonics and real harmonics

New offset Gamma procedure. Invariant tensor expansion.
Anisotropy problem.

Wave function and MPB rotation

EIBZ symmetrization

bloch: FFT of sigm.

Calculate effective mass:

hvccfp0: v(exact) vs. v(cal) (eigenvalue of v matrix).

Spectrum function mode:

lmfa:

alagr3z: efficient? We may need improvement.

PFLOAT:

(not now?) ropbes.f ropyln.f had a problem due to compilar option.

FTMESH: denser gives better? ehk=ehf?


Atomic position relaxiation:

epsPP mode:
   Need to check it.
   q=0 limit.

With FSMOM, Efermi is not uniquely given in job_band_nspin2*.
It is given by a bndfp-bzwtsf-bzwsf L300 block
   if ((.not. lfill) .or. (metal .and. (nkp .eq. 1))) then
   (bisection method to determine a middle of LUMO and HOMO).
```

```
It can give some energy between LUMO and HOMO.
Small changes of computational condition can give large change.
But no problem.

======
PDOS: sigm_fbz is required.
(when cp sigm,rst,GWinput ->LDA-like result.
 Then cp sigm_fbz ->it fails.
 Need to make new directory, and copy rst,sigm_fbz.)
And how to check it. (whether

======
mixbeta:
takao@TT4:~/ecalj/fpgw$ grep mixbeta */*.F
main/hqpe.sc.m.F:      call getkeyvalue("GWinput","mixbeta",beta,default=1d0,status=ret)
mixing parameter on sigm file.
As the default beta is unitiy, mixsigm and mixsigma files are

=======
Check convergecne on QSGW.
grep rms lqpe*
```

References

[1] Takao Kotani, Hiori Kino, and Hisazumu Akai. Formulation of the augmented plane-wave and muffin-tin orbital method. Journal of the Physical Society of Japan, 84(3):034702, March 2015.

[2] Takao Kotani. Quasiparticle self-consistent GW method based on the augmented plane-wave and muffin-tin orbital method. J. Phys. Soc. Jpn., 83(9):094711 [11 Pages], September 2014. WOS:000340822100029.

[3] Takao Kotani, Mark van Schilfgaarde, and Sergey V. Faleev. Quasiparticle self-consistent gw method: A basis for the independent-particle approximation. Physical Review B, 76(16):165106, 2007.

[4] Takao Kotani and Hiori Kino. Linearized augmented plane-wave and muffin-tin orbital method with the PBE exchange-correlation: Applied to molecules from h-2 through kr-2. J. Phys. Soc. Jpn., 82(12):124714 [8 Pages], December 2013. WOS:000327350700043.

[5] Takao Kotani and Mark van Schilfgaarde. Fusion of the lapw and lmto methods: The augmented plane wave plus muffin-tin orbital method. Phys. Rev. B, 81(12):125117, Mar 2010.

[6] Takao Kotani and Mark van Schilfgaarde. Quasiparticle self-consistent GW method: A basis for the independent-particle approximation. Physical Review B, 76(16), October 2007. WOS:000250620600028.

[7] Christoph Friedrich, Stefan Blügel, and Arno Schindlmayr. Efficient implementation of the {GW} approximation within the all-electron {FLAPW} method. Physical Review B, 81(12), March 2010.

[8] Christoph Freysoldt, Philipp Eggert, Patrick Rinke, Arno Schindlmayr, R.W. Godby, and Matthias Scheffler. Dielectric anisotropy in the GW space–time method. Computer Physics Communications, 176(1):1–13, January 2007.

[9] Takao Kotani and Mark van Schilfgaarde. Fusion of the LAPW and LMTO methods: The augmented plane wave plus muffin-tin orbital method. Physical Review B, 81(12), March 2010. WOS:000276248900054.

[10] Takao Kotani and M. van Schilfgaarde. All-electron gw approximation with the mixed basis expansion based on the full-potential LMTO method. Solid State Commun., 121:461, 2002.

[11] F. Aryasetiawan and O. Gunnarsson. Product-basis method for calculating dielectric matrices. Phys. Rev. B, 49:16214, 1994.

[12] J Rath and A J Freeman. Generalized magnetic susceptibilities in metals: Application of the analytic tetrahedron linear energy method to Sc. Phys. Rev. B, 11:2109, 1975.

[13] D. R. Hamann and David Vanderbilt. Maximally localized wannier functions for gw quasiparticles. Phys. Rev. B, 79:045109, Jan 2009.

[14] A. Svane and O. K. Andersen. Evaluation of four-center integrals with the linear muffin-tin orbital tight-binding method. Physical Review B, 34(8):5512, 1986.

[15] Christoph Freysoldt, Philipp Eggert, Patrick Rinke, Arno Schindlmayr, R.W. Godby, and Matthias Scheffler. Dielectric anisotropy in the GW space–time method. Computer Physics Communications, 176(1):1–13, January 2007.

[16] Robert M. Pick, Morrel H. Cohen, and Richard M. Martin. Microscopic theory of force constants in the adiabatic approximation. Physical Review B, 1(2):910, 1970.

[17] P. Vogl. Microscopic theory of electron-phonon interaction in insulators or semiconductors. Physical Review B, 13(2):694, 1976.

[18] Paolo Giannozzi, Stefano De Gironcoli, Pasquale Pavone, and Stefano Baroni. Ab initio calculation of phonon dispersions in semiconductors. Physical Review B, 43(9):7231, 1991.

[19] J. Sjakste, N. Vast, M. Calandra, and F. Mauri. Wannier interpolation of the electron-phonon matrix elements in polar semiconductors: Polar-optical coupling in GaAs. Physical Review B, 92(5), August 2015.

[20] Takao Kotani, Hiori Kino, and Hisazumu Akai. Formulation of the Augmented Plane-Wave and Muffin-Tin Orbital Method. Journal of the Physical Society of Japan, 84(3):034702, March 2015.

[21] Ersoy Şaşıoğlu, Arno Schindlmayr, Christoph Friedrich, Frank Freimuth, and Stefan Blügel. Wannier-function approach to spin excitations in solids. Physical Review B, 81(5), February 2010.