All-electron GW code manual (lmf6/lmf7 + fpgw033a1)

Takao Kotani

April 18, 2008

Contents

•**Index of I/O fies, shel scripts and executions.**

1 Introduction

Manual for our all-electron GWcode (one-shot GW, QP self-consistent GW, spectrum function, W and magnetic susceptibility). As the inputs for the GWcalculation, we have to supply the eigenfunctions and the eigenvalues in addition to the crystal-structure informations. The eigenfuncions are expanded by the two kinds of basis functions, the atomic-like argumentation functions in the muffin-tin(MT) spheres, and the plane-waves in the interstitial region., say, the interstitial plane-wave (IPW) hereafter. IPW is defined as the usual plane waves in the interstitial region, but zero within MTs'. This GWcode is applicable not only to FP-LAPW but also for FP-LMTO, because its envelope functions are also well expanded by IPW in the interstitial region; but we here suppose to use lmf (Mark's fp-lmto). The Coulomb matrix v, the dynamically screened Coulomb interaction W, and so on, are expanded in a mixed basis set which consists of the two contributions; (i)the local atom-centered functions confined to MT spheres, so-called the product basis; (ii) IPW. The product-basis are calculated from products of solutions to the Schrödinger equation within the MT sphere, and can include any of the core states. Thus, the core functions can be treated on an equal footing with the valence electrons. In addiion, we include full energy-dependence of W. The code is developed starting from the GWcode by **Ferdi Aryasetiawan** for LMTO-ASA. We added some improvements in addition to the key-feature, the mixed-basis.

In this manual, we explain how to use the GWcode; you have to prepare the LDA results as input by the FP-LMTO codes before you do GW. The FP-LMTO code is called as **lmf** and his package contains the driver routines **lmfgw, lmf2gw** for the GWcalculations to prepare required eigenfunctions and eigenvalues for the GWcalculation. The FP-LMTO code is compilied by Mark van Schilfgaarde, originally developed as the **NFP package** by **Michel Methfessel and Mark van Schilfgaarde**. See documents contained in the lm6.14 packages as for it.

The GWcode itself is independent from how you prepare the eigenfunctions and eigenvalues. It is possible to make a driver for other LDA codes (gwinput_v2.f is a key rouitne to readin the eigenfunctions and eigenvalues).

contributors: Mark van Schilfggarde, Sergey Faleev, Manabu Usuda, Takashi Miyake, and Hiori Kino

What can we do with the fpgw033a1 package (*GW*code)?

- Quasi-particle(QP) energy in the 1st-itteration from LDA. (one-shot *GW*)

  Make band plot for LDA and the QP energies.

- Spectrum function of the self-energy Σ.

- Dielectric function, and its inverse. (including local-field effect or not).

- QP self-consistent *GW*

- magnetic susceptibility

- total energy (testing).

But *GW* calculations are very expensive. So you may not apply it directly to your system...

Main Ref

PRB76 165106 (2007). Denoted as Ref.I. EQ.xxx means Equation in Ref.I.

2 Overview of *GW* calculation

See section II of Ref.I. ————————————————-

Core orthogonalization problem:
At you see in EQS.(32) or (33). it should be $\langle \exp(-i\mathbf{qr})|\Pi|\exp(i\mathbf{qr})\rangle \to 0$ at $\mathbf{q} \to 0$ because $\langle \psi_{\mathbf{k}n}|\psi_{\mathbf{k}n'}\rangle = 0$ for occupied n and unoccupied n'. However, core eigenfunction is not completely orthogonal to the valence eigenfunction (in our FP-LMTO scheme). Thus this behevior can not be so perfect. In order to keep the behevior, we orthogonalize the core eigenfunctions. It is by an optional switch `CoreOrth` in the input file `GWinput`(See the description in the explanation of it). However, we find that QPE usually affected little by this option.

Hilbert transformation (Sergey mode):
This is standard calculation now. Our previous version of our GW code, we calculate Π for required ω points directrly with the tetrahedron method. (corresponds to a script **gw_lmf** explained after). Our standard version (a script **gw_lmfh**) is through the Hilbert transformation (Kramers-Krönig relation). Then we rather calculate only the imaginary part of Π at first. And then we get full Π through the Hilbert transformation.
[In other words, we calculate the imaginary part of Π by replacement of $\left(\frac{1}{\omega-\epsilon_{\mathbf{q}+\mathbf{k}n'}+\epsilon_{\mathbf{k}n}+i\delta} - \frac{1}{\omega+\epsilon_{\mathbf{q}+\mathbf{k}n'}-\epsilon_{\mathbf{k}n}-i\delta}\right)$ with $\delta(\omega-\epsilon_{\mathbf{q}+\mathbf{k}n'}+\epsilon_{\mathbf{k}n})$ in Eq.(**??**). Then $\delta(\omega-\epsilon_{\mathbf{q}+\mathbf{k}n'}+\epsilon_{\mathbf{k}n})$ is replaced by the original one (taking convolution).]

3 Brillouin-zone integral for the self-energy; the smearing method and the offset-Γ method.

- **Smearing method**

  [Note that this is not for not for Π—it is usually evaluated by the tetrahedron method.]

  Our smearing method means, we replace δ function with $\bar{\delta}(\omega)$ as shown in EQ.41.

$$\bar{\delta}(\omega) = \frac{1}{E_{\rm smear}} \text{ for } -\frac{E_{\rm smear}}{2} < \omega < \frac{E_{\rm smear}}{2}, \text{ otherwise zero } (\texttt{GaussSmear} = \text{off mode} \quad E_{\rm smear} = \texttt{esmr in GWinput}) \tag{1}$$

or

$$\bar{\delta}(\omega) = \frac{1}{\sqrt{2\pi}\sigma}\exp(-\frac{\omega^2}{2\sigma^2}) \quad (\texttt{GaussSmear} = \text{on mode}, \sigma = \texttt{esmr in GWinput})$$

As for insulator, $E_{\rm smear}$ is irrelevant (due to numerics,$E_{\rm smear} = 0$ is not allowed. You need to set $E_{\rm smear}$ smaller than band gap. But not too small). However, you may need to pay attention to the size of $E_{\rm smear}$ in the case of metal. (The pole distribution around the Fermi energy is shown in a file `DOSACC.lda`). $\rho_{\mathbf{q}nm}(\mathbf{k})$ can have unsmooth behevior as a function of k in BZ due to the Fermi energy cutoff in the case of metal. Larger $E_{\rm smear}$ reduce the unsmoothness. With denser meshing in BZ, you are allowed to use smaller $E_{\rm smear}$.

- **Offset gamma method** (`BZmesh=1`) See Ref.I. We have to take the two limit $E_{\rm smear} \to 0$ and $N_1N_2N_3 \to \infty$. There could be a convergence problem as for the states $\Psi_{\mathbf{q}n}$ whose $\epsilon_{\mathbf{q}n}$ are near $E_{\rm F}$ in the case of low DOS at $E_{\rm F}$. In Fig.1, we showed the convergence test for $\langle\Psi_{\mathbf{q}n}|\Sigma_{\rm x}|\Psi_{\mathbf{q}n}\rangle$ as a function of $E_{\rm smear}$ in the case of CaB_6. The DOS at $E_{\rm Fermi}$ for CaB_6 is quite small, therefore, it is a severe test for the smering method. Through the comparison between 444 and 666 case, we can say a rapid change at $E_{\rm smear} \to 0$ will be virtual because of the finite number of k points. Therefore we can use $E_{\rm smear} \sim 0.05$ Ry in order to avoid such a finite number effect at $E_{\rm smear} \to 0$. Then we can expect 0.1 eV level of accuracy under the assumption of the flat behevior at $E_{\rm smear} \to 0$. Due to the calcellation effects, $\Sigma_{\rm x} + \Sigma_{\rm c}$ can give better convergences.

Offset-Γ method (`BZmesh=2`)

See EQ.53 and after. $W_{\mathbf{Q}}$ is the weight for the offset Γ point $\mathbf{Q}$. We usually use rather very small value, e.g.,0.01 or less. Integration weights $W_{\mathbf{k}}$ is given as $W_{\mathbf{k}} = 1/(N_1N_2N_3)$ except shortest $\mathbf{k}$ points around Γ point. As for the weight $W_{(\text{shortest } \mathbf{k})}$, we choose it so that the sum of all $W_{\mathbf{k}}$ including $\mathbf{Q}$ get to be unity. Then $\mathbf{Q}$ is determined in the same manner in the `BZmesh =1` case.

This scheme will be sometimes rather advantageous than `BZmesh=1`. In the case of `BZmesh=1`, it gets problematic to treat rather anisotropic systems like one-dimentional atomic chain. In the case, we can not determine reasonable $\mathbf{Q}$ for the BZ division for, e.g., ($N_1 = N_2 = 1, N3$ =large number). It can be a serious difficulty to check the convergence.

On the otherhand, there is no problem in `BZmesh=2`.

We can choose $\mathbf{Q}$ close to Γ point; you can use any $\mathbf{Q}$ (if it is close enough to Γ). For smaller $W_{\mathbf{Q}}$, we have smaller $\mathbf{Q}$.

Notes:

(1) We checked that obtained QPE are not dependent on the choice of $W_{\mathbf{Q}}$ at the limit of $W_{\mathbf{Q}} \to 0$ (However, you have to be careful to use too small $W_{\mathbf{Q}}$ so as not to destory orthogonality of eigenfunctions). This scheme just pick up the divergent part of integral correctly.

(2)As for some anisotropic case (e.g. antiferro-magnetic II NiO), we need to use "negative $W_{\mathbf{Q}}$" because the shortest $\mathbf{k}$ on regular mesh is already too short and the integral of auxially function evaluated on the regular mesh of EQ.47. already larger than the exact value. However, `BZmesh=2` should work OK even for such a case.

(3) In cases (e.g.Si), this mesh can be a bit poor to keep the crystal symmetry as for QPE, because not all the mesh points are mapped to the anothe mesh points by some symmetry operation. Then we may need to take denser mesh to reduce the artificial poorness.

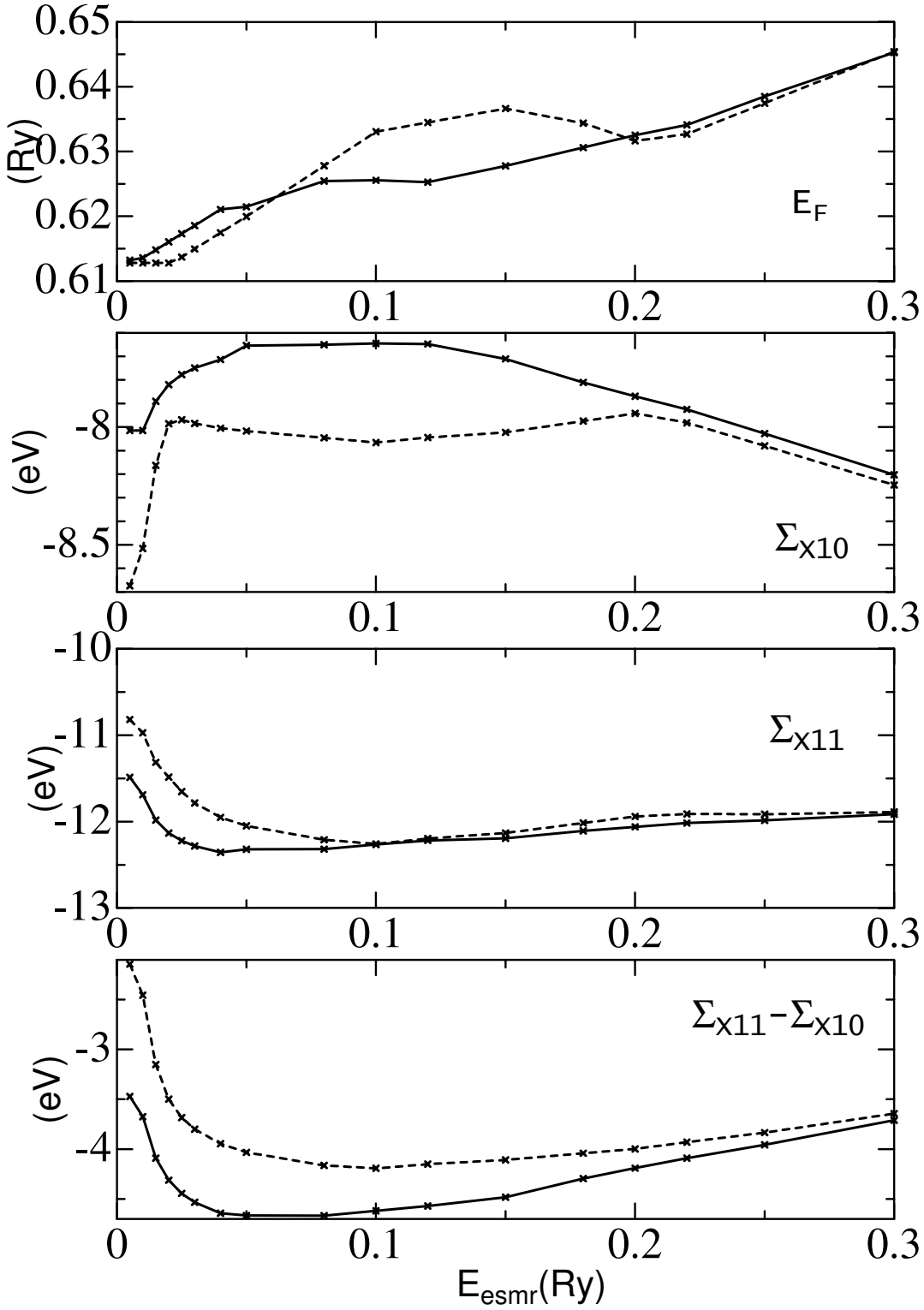
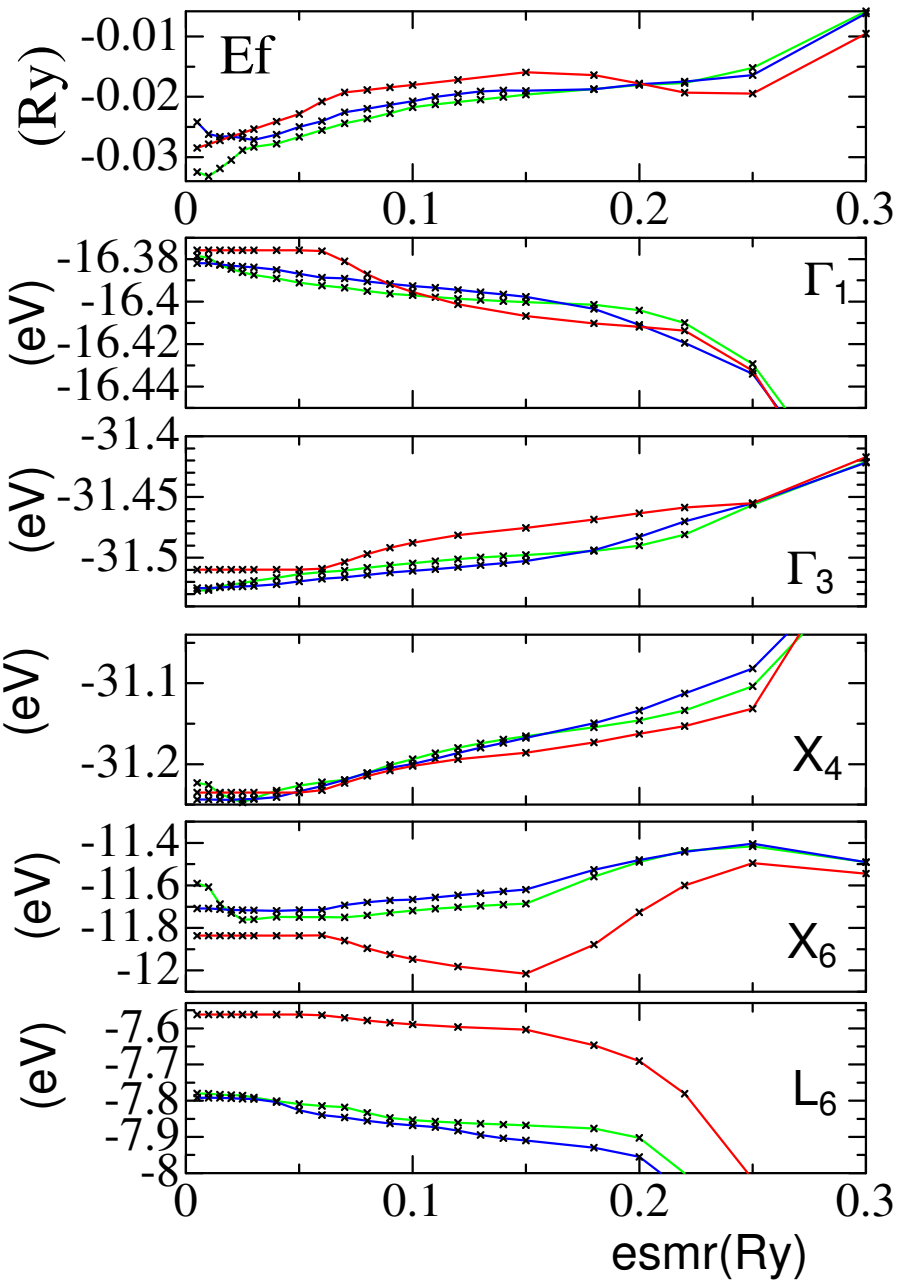


Figure 1: $\langle \Psi_{\mathbf{q}n} | \Sigma_{\mathrm{x}} | \Psi_{\mathbf{q}n} \rangle$ as fucntions of E_{smear} for CaB_6, whose LDA bands are metallic. (`GaussSmear`=off case). The state X_{10} is the top of the valence bands, and X_{11} is the bottom of the conduction bands. Solid line are for 666 case. Broken lines are for 444 case. For (`GaussSmear`=on), we expect somehow smoother behevior(not shown here).



Cu esmr test for 666(red) 101010(green) 141414(blue)

Exchage energy <psi | Σ_x| psi> for each states.

Figure 2: exchange self-energy test for Cu. (`GaussSmear`=off case). (esmr means $E_{\rm smear}$.) You can not use so small $E_{\rm smear}$ so as to avoid the effects of discretization. We can use smaller $E_{\rm smear}$ for denser divisions(meshing) of BZ. For (`GaussSmear`=on), we expect somehow smoother behevior(not shown here).

4 Package installation.

1. `tar -zxvf fpgw033a1.tar.gz` in a directory (at ecal/ or anythere else). See fpgw/fpgw_version_log for changes from older versions.
2. Move to fpgw/exec. Modify `fpgw/exec/make.inc`. Need BLAS and (a part of) LAPACK.
3. Do `make init`. This call a python script "checkmodule" and generate moduledepends.inc, which is used for make.
4. Do `make`. This generates binaries. `make install` just copy binaries to your bin (set name of your bin in make.inc). `make install2` copies scripts to your bin.
5. You also have to copy lmf lmfa lmfgw lmf2gw (these are from lmf(GW driver)) to your bin.

There are scripts—but most of all are for testing purpose. Important ones are
`gw_lmfh`(one-shot GW),
`gwsc`(QSGW),
`eps_lmfh, epsPP_lmfh` (epsion) ,
`epsPP_lmfh_chipm`(spin susceptibility).

Test installation. Copy fpgw/TESTinstallGW to another directory, and do calculations (name of directory corresponds to the name of script; e.g `si:gw_lmfh` means 'do `gw_lmfh si`'.

```
si:gw_lmfh/              Results: QPU
si:gwsc/                        : QPU (you need to wait several itteration)
gas:gwsc/                       : QPU (you need to wait several itteration)
fe:eps_lmfh_chipm/              : ChiPM*
fe:epsPP_lmfh_chipm/            : EPS*
gas:eps_lmfh/                   : EPS*
gas:epsPP_lmfh/                 : EPS*
```

In each directory, you can do `gw_lmfh` and so on. Output of each srcipts are in `out` file. And results should be in agreement with QPU and so (as for gwsc case, agreement can be not so perfect—it depends of how to make convergence). These tests may take minites to hours. `si:gw_lmfh` requires a few minutes.

5 How to execute the *GW* calculation? Overview.

In this section, I will explain how to do the one-shot *GW* calculation from LDA.

———— Warning! ———————————————— (probably I am sloppy about this rule).

- **This font** is for executions or shell scripts.
- **echo 3|hbasfp0** means doing **hbasfp0** with the argument '3' from the standard input. See the end of subsection 5.3.
- This font is for the I/O files by executions or by scripts.
- `This font` is for files, directories, contents of files, or variables used in codes.
- ctrl.si,rst.si and so on mean in the case of Si. You have to replace si with suitable name (extension of ctrl file).
- There are files named *foo*U and *foo*D, which are for up spin and for down spin, respectively; e.g. ,SEXU and SEXD. We sometimes use *foo*U to denote *foo*U and *foo*D.

————————————————————————————

At first, you have to do the self-consistent FP-LMTO LDA calculation. (It is done by **lmfa** and **lmf** in lm package. Staring from ctrl.si (in the case of silicon), you get the converged LDA potential in rst.si.) After you get the LDA result, you can start the *GW* calculation Practically, you have to follow steps below in order to calculate the quesi-particle (QP) energies.

Step 1. Get LDA result. In lmf, you can start LDA calculation from ctrl.si To start *GW*, you need the LDA result rst.si together with ctrl.si.

[NOTE! Due to poorness of our *GW* code, you have to use the same `LMXA` (l in the expansion of eigenfunction in each MT) for all the MT spheres.]

Step 2. Run the script **mkGWIN_lmf2**.

The purpose of this script is to get GWinput.tmp. Other generated files are just useless (or for your info).

Step 3. Edit GWinput.tmp and save it as GWinput

These step 2. and step 3. are just only to get GWinput.

GWinput is the input file describing the computational conditions for *GW* calculation.

Step 4. Run the script **gw_lmfh**.

To invoke **gw_lmfh**, you needs files ctrl.si, rst.si and GWinput (in the case of si). The main output are QPU files and so on; See Section 7. If things OK, it shows as

```
> gw_lmfh si
si
FORTRAN STOP  OK! qg4gw mode=1 normal mode
 OK! lmfgw mode=1
FORTRAN STOP  OK! lmf2gw: end --- DATA4GW_V2 is written
FORTRAN STOP  OK! rdata4gw_v2
FORTRAN STOP  OK! heftet mode=1 EFERMI generated
FORTRAN STOP  OK! hchknw: write nw to NW
FORTRAN STOP  OK! hbasfp0 ix=3 core mode
FORTRAN STOP  OK! hvccfp0
FORTRAN STOP  OK! hsfp0: Core-exchange mode
FORTRAN STOP  OK! hbasfp0 ix=0 normal mode
FORTRAN STOP  OK! hvccfp0
FORTRAN STOP  OK! hsfp0: Sergey's Exchange mode
FORTRAN STOP  OK! hx0fp0 ixc=11 12 Sergey F. mode
FORTRAN STOP  OK! hsfp0: Sergey's Correlation mode
FORTRAN STOP  OK! hqpe
```

This output shows that the script **gw_lmfh** invokes **lmfgw** and so on. As we explain just below, the precedures until "`OK! lmf2gw:  end --- DATA4GW_V2 is written`" corresponds to the end of **(2)Preparation stage**.

Step 5. Run the script **hqpemetal** in the case of metal

This is in order to get the correct Fermi energy by the tetrahedron method.

(To execute **hqpemetal**, you need to calculate all the QP energies at least just above the Fermi energy).

[this is for old users: Note that the newer **hqpemetal** will not work for old results by fpgw020; you need to change nnv in LMTO file.]

In the case of the Antiferro materials, the compuational efforts reduced to be half (only hx0fp0 part; most expensive for one-shot GW) if you prepare a file Anfcond by handbefore you execute **gw_lmfh**. See next section for it. (this function now works only for the case that a symmetry operation is [a translation verctor with spin inversion].)

From the view of computational procedure, the GWcalculation are devided into these stages:

(0)LDA calculation (step 1.)

(1)Pre-Preparation stage (step 2– step 3)

(2)Preparation stage (step 4)

(3)Main stage (step 4)

(4)Post Main stage (step 5)

The script **gw_lmfh** automatically do all the procedures contained in the stage (3) and stage (4).

In anyway, it is necessary to look into these shell scripts, and observe the cosole outputs of each programs called from the script (they are usually reserved to I* files,e.g. Ibas or so. Look into the scripts.) You don't need to understand all items in console outputs; I am sloppy to orgainize it—so, not so meanigful or debugging check write are in these console outputs.

5.1 (1)Pre-Preparation stage to write GWinput

The purpose of this stage is to write GWinput. So you can pass this stage if you have GWinput already. A template GWinput.tmp is generated by **mkGWIN_lmf2**. Files used by **mkGWIN_lmf2**

are

Input files

• ctrl.si : The master control file of the self-consistent FP-LMTO LDA calculation.

• rst.si : This contains self-consistently-determined LDA potential.

Output files

• GWinput.tmp : A file including computational conditions for the GW calculation. In addition, it specifies the **k** poins for which you calculate the QP energy.

When you invoke **mkGWIN_lmf**, it asks you to supply three numbers for BZ integration as

```
== Type three integers n1 n2 n3 for Brillouin Zone meshing for GW! ==
 n1=
```

Then you need to type a number e.g. as "2 Return" for n1. Then you need to repeat it for n2 and n3 as

n1= 2 Return

n2= 2 Return

n3= 2 Return

. These numbers specifies what k poins in BZ is used for BZ integration (In this case, $2 \times 2 \times 2 = 8$ **k** point in the 1st BZ is used. Based on our experiences, we need $4 \times 4 \times 4$ to get band gap for Si with ≈ 0.1 eV accuracy).

Then you have to edit GWinput.tmp and copy it to GWinput. We details the GWinput in later chapter.

5.2 (2)Preparation stage.

In order to start this stage, we need self-consistent LDA potential file and GWinput.

- **echo 0 |lmfgw**: Get some small information files to start **qg4gw**.
- **echo 1 |qg4gw** : Get **k** points used in the GW calculations and the correponding **G** vectors.
- **echo 1 |lmfgw** : Calculate the LDA eigenfunctions, eigenvlaues, and $\langle\psi|V_{\rm xc}^{\rm LDA}|\psi\rangle$ for these **k** in the form of Eq.(**??**).
- **lmf2gw**: store these datas into DATA4GW_V2 and CphiGeig , whose I/O is controlled by a key subroutine **gwinput_v2.f**.

These procedures described in **gw_lmfh** are in the case fo FP-LMTO.

At the end of this stage, we get required eigenfunctions, BZmesh data, and so on, which are required for the successive main stage.

——note ————————————————————

Here, e.g., **echo 1|qg4gw** and so on means that we invoke **qg4gw** with the argument 1 from the standard I/O (not from console). So it is equivalent with

>qg4gw Return

>1 Return

from the console.

But it may not go ahead if you still supply

>\ Return

in cases for **hsfp0, hvccfp0, hx0fp0**, because they have two `read(5,*)` and you have to cause the readin error for the second read to go ahead [1]

5.3 (3)Main stage starting from DATA4GW_V2.

We can start the main stage of *GW* cakckatuib from these files;
- DATA4GW_V2 : Crystal structures and so.
- CphiGeig : Eigenvalues and Eigenfunctions
- QGpsi : q and G vector for the eigenfunction(q means **k** in the previous section),
- QGcou : q and G vector for the Coulomb matrix
- Q0P : q points near q=0 instead of q=0,
- BZDATA : q points date (and tetrahedron weights if necessary) for BZ integrals.
- QIBZ : irreducible q points (This is also contained in BZDATA).
- CLASS : class information for atomic sites.
- SYMOPS : point group operation
- GWinput : computational conditions.

These files are not dependent on how to prepare the eigenfunctions, whether LMTO or LAPW. If you want to do GW calculation with eigenfunction given by other codes, you have to write "a *GW* driver routine" which generates files DATA4GW_V2, CphiGeig and CLASS by yourself. (See later sections for these files).

As for the computational flow of the procedure, see the script **gw_lmfh**. As for this stage, **gw_lmfh** do;

- **rdata4gw_V2**: Read DATA4GW_V2, and decompose it into files required in the followings.
- **heftet** : Get the Fermi energy EFERMI by tetrahedron method. It is used in **hx0fp0**.
- **hchknw** : stores the number of required ω points along real-axis into NW.
  (NW is not essentially used, but is supposed to exist in the followings.)
- **echo 3|hbasfp0**: gives the product basis for Core exchange.
- **echo 0|hvccfp0**: gives the Coulomb matrix for the Core exchange.
- **echo 3|hsfp0** : gives the Core exchange part of the self-energy.
- **echo 0|hbasfp0**: gives the product basis.
- **echo 0|hvccfp0**: gives the Coulomb matrix v.
- **echo 1|hsfp0** : gives the exchange part of the self-energy.
- **echo 1|hx0fp0** : gives the correlated part of the screened Coulomb interaction $W - v$.
- **echo 2|hsfp0** : gives the correlated part of the self-energy.
- **echo 0|hqpe** : gather datas and write down final results into QPU and so on.

Then you can do the script **hqpemetal** in the case of metal in order to get the Fermi energy for the QP energies.

[1] The second read(5,*) is used for the case of parallel computing and you need to cause the readin error for usual single machine computing; (but this mode of pararallel computing is not maintained now).

5.4 Other functions (or scripsts)

In addition to **gw_lmfh**, there are some other additional scripts and functions.

- **gw_lmfh** : The one-shot GWcalculation explained here.

- **gwsc** : Semi self-consistent GWcalculation.
- **epsPP_lmfh, eps_lmfh** : Dielectric function without or with local-field effects.
- run-mode 4 of **hsfp0**: to plot the spectrum function $\Sigma(\omega)$.
- **gwband_lmf** : for plotting band (not well maintained now).
- **epsPP_lmfh_chipm** : non-interacting spin susceptibility. One-degree of freedom like Rigid moment approx. After it ends, you need to do `calj_nlfc_metal` and/or `calj_summary_mat` to get the full spin susceptibility.

Scripts below are for tests

- **extest, extest_repeat**: which is in order to check the dependence of $\Sigma_{\rm x}$ as for $E_{\rm smear}$.
  gw_lmf : The one-shot GWcalculation, older version.
  (Direct calculation of the polarization function without going through the Hilbert-transformation.
- **eps_lmf**: Dielectric function with local-field effects. Direct mode(Not Hilbert-transformation).
- **gwpara_lmf** : A test gw script for parallel computing (testing. it may not work now).
- **eps_lmfh_chipm** : spin susceptibility (full mixed basis). Test purpose.

6 GWinput

The main input files is GWinput. [it is a unified file of old GWIN0, GWIN_V2, and QPNT]. This controls the setting of GW calculation. The file GWinput consists of structures as
keyword1 data1
keyword2 data2
...
In each lines, it consists of keyword and data. Data can be sigle or plural. As for keywords, upper case or Lowercase is not distingushed. All keywords should start from 1st column (no space at head). Order of lines are irrelevant. As for logical variable, you can use anything among (true, ok, .true. yes, on, 1, T) for .true., and anything among (false, ng, .false., no, off, 0, F) for .false.

Or we have "tag sections" in GWinput specified by `<PRODUCT_BASIS>`, `<QPNT>`, `<PBASMAX>`, `<QforEPS>`, and `<QforEPSL>`. (`<PRODUCT_BASIS>` is requires for all kinds of calculations. `<PBASMAX>` is optional. `<QforEPS>` and/or `<QforEPSL>` are required for eps mode). It is like

```
<PRODUCT_BASIS>
 tolerance to remove products
  0.100000D-07 ! =tolopt
 lcutmx(atom)
  3 3
  atom   1
...
</PRODUCT_BASIS>
```

. In these tag sections, you have to keep format for its own (usually numbers are readin by free format `read(5,*)`).

The fundamental readin routine for GWinput is a subroutine `getkeyvalue` defined in `gwsrc/keyvalue.f` written by Dr.Kino. `getkeyvalue` is a general and convenient readin routine in full use of the f90 features. Read a head part of the file and try to do "grep getkeyvalue *.F" in `gwsrc/` or `main/` so as to see how to use it (test routine is `main/kino_input_test.F`.)

So the GWinput consists of three sections
1.General section
2.`<PRODUCT_BASIS>` section
3.`<QPNT>` section
4.`<PBASMAX>` section
5.`<QforEPS>`,`<QforEPSL>` section
We will explain each by each in the followings.

6.1 General section

In genral section, it looks like

```
! ##### From GWIN0 #################
n1n2n3  2 2 2   ! for BZ meshing in GW
QpGcut_psi  2.7 !(See unit_2pioa for unit) |q+G| cutoff for eigenfunction.
QpGcut_cou  2.5 !(See unit_2pioa for unit) |q+G| cutoff for Coulomb and W.
unit_2pioa  off ! on--> unit of QpGcut_* are in 2*pi/alat; off --> a.u.
emax_chi0   4.  !(Ry) emax cutoff for chi0  (Optional)
emax_sigm   2.  !(Ry) emax cutoff for Sigma (Optional)

alpha_OffG  1   !(a.u.) Used in auxially function in the offset-Gamma method.

! ##### FREQUENCIES from GWIN_V2 #################
dw      0.01  !(a.u.) mesh width along real axis.
omg_c   0.05  !(a.u.) Only meaningful for Sergey mode as gw_lmfh.
              ! coaser mesh for higher energy. Width get to be doubled at omg_c.
niw        6  ! Number of frequencies along Im axis. Used for integration to get Sigma_c
              ! E.g. try niw=6 and niw=12
delta -.1D-7  !(a.u.)  Broadening of x0. negative means tetrahedron method.
              ! used by hx0fp0. You get smeard x0 witth abs(delta).
deltaw 0.020  !(a.u.)   Mesh for numerical derivative to get the Z factor
GaussSmear on ! Gaussian or Rectangular smearing for Pole of G^LDA with esmr for hsfp0.
esmr   0.003  !(Ry) used by hsfp0. Keep esmr smaller than band gap for insulators
              ! Poles of G^LDA are treated as if they have width esmr in hsfp0.
              ! Change esmr for metals.  See DOSACC*---especailly around Ef.
```

---

1. `n1n2n3` 3 integers as $N_1, N_2, N3$ (no default); They are ≥ 0.
   Brillouin Zone mesh for integeration is determined by keywords`BZmesh` and `n1n2n3`. In EQS.47(regular mesh) or 53(off-regular mesh).

   `Chi_RegQbz` (on or off)

   Chi_RegQbz = on (default): Use regular mesh (including gamma) for eps calculation.

   Chi_RegQbz = off : Use off-regular mesh (Not including gamma) for eps calculation.

   `BZmesh` (1 or 2)

   BZmesh 1(default) : Use regular mesh (including gamma) for sigma calculation.

   BZmesh 2 : Use off-regular mesh (Not including gamma) for eps calculation.

   So we now have four (2x2) combinations for the 1shot GW calculation (Chi_RegQbz x BZmesh). I think that BZmesh 2 is now not good for self-consistent GW.

2. Plane wave ($\mathbf{q} + \mathbf{G}$) cutoff
   `QpGcut_psi` 1 real (no default)
   `QpGcut_Cou` 1 real (no default)
   `unit_2pioa` 1 logical (no defalt)

We have two cuoff for $\mathbf{q}+\mathbf{G}$. `QpGcut_psi` is the cutoff of $|q+G|$ for the IPW in the expansion of the eigenfunctions. `QpGcut_Cou` is for the IPW of the interactions v, D, W. Its unit is specified by `unit_2pioa` ; "off" means unit in a.u. and "on" means in unit of $\frac{2\pi}{\text{alat}}$. (alat is length scale unit in `ctrl.*`).

3. Cutoff for used bands.
   `emax_chi0`: 1 real (optional,default=∞), in Ry
   `emax_sigm`: 1 real (optional,default=∞), in Ry
   `nband_chi0`: 1 integer (optional,default=∞)
   `nband_sigm`: 1 integer (optional,default=∞)
   These specify how many bands you use in **hx0fp0** (for chi0) and in **hsfp0** (for sigma). Higher bands above them are neglected.

4. Energy mesh related parameters.
   `dw` : 1 real (a.u.). Mesh width along real axis for $W(\omega)$.
   `omg_c` : 1 real (a.u.).
   `dw` and `omg_c` determines ω mesh along real axis for $W(\omega)$. Energy mesh is getting coarser at higher energy. Energy bin width get to be doubled at `omg_c`.
   But `omg_c` is not used in gw_lmf; then energy mesh is fixed as `dw`. We calculate $W(\omega)$ at these energy mesh as $W(\omega=0), W(\omega=\texttt{dw}), W(\omega=2\times\texttt{dw}), W(\omega=3\times\texttt{dw}), ...$ and then use them for the numerically interpolate to determine $W(\omega'=\omega-\epsilon_{\mathbf{q}-\mathbf{k}n})$. See FIG.1 in Ref.I.

   (WARNING! Some of my examples may show as if they are in "(Ry)". But they are Wrong!)

`delta` : 1 real (a.u.). Fix it as -1d-8 or so for gw_lmfh, eps mode and so. This is the size of δ in denominator of Π (EQ.32). But (I think that) you can not use it so as to make broadning for theoretical test (maybe not exactly corresponding to δ).

[Old note. Need check: In gw_lmf, it is used for broadening of x0 when it call hx0fp0. Then `delta` is δ is EQ.32. The sign of `delta` is just used as a flag whether you use the tetrahedron method of dielectric constant [15] or not; minus sign means "Use the tetrahedron method for D"; plus sign means you do it by simple sum. You can usually use this default setting. But it might be possible to use a larger value to smear the fine structures on the energy-dependence of W in cases. This might be necesary if W is so energy-dependent and `dw` is not so small to resolve the structure —but I don't know.]

`niw` : 1 integer.
Number of integration points along the imaginary axis(FIG.1) to get Σ_c. See routines `wint*` called from `sxcf*.F`, which is called from the main routine `hsfp0.m.F` (or `hsfp0.sc.m.F` in the QSGW case). The integration points are $i\omega'(n) = i(1/x(n)-1)$, where $x(n)$ is the usual Gaussian-integration points for the interval [0,1]. In addtion, we give the special analitical treatment for the peakey part at $\omega'=0$. Out tests shows `niw`=6 for Si is good enough for 0.01 eV accuracy. The convergence as for `niw` is quite good. This integration scheme has been devloped by Ferdi Aryasetiawan. The number of points should be the one of 6,10,12,16,20,24,32,40,or 48. It is because we use a `subroutine gauss` in `/gwsrc/mate.F` prepared by Ferdi. We will replace better one in future. See II-F in Ref.I.

`GaussSmear` : 1 logical

`esmr` : 1 real (Ry). Used by hsfp0 (and hsfp0.sc for QSGW).
Poles of the Green fucntion $G^{\rm LDA}$ are treated as if they have width esmr in hsfp0. If `GaussSmear` is on, each pole of $G^{\rm LDA}$ is smeared by a Gaussian function with σ =`esmr` in the calculation of hsfp0. If `GaussSmear` is off, we assume rectangular smearing for the poles. Usually it is necessary to take rather smaller value than band gap for insulators. Try to use 0.003 or so in the case of Si and `GaussSmear`=on.
In the case of insulator, it can be smaller 0.0001 or less (maybe), but it should have some size in the case of metal.

`deltaw`: 1 real (a.u.) only for one-shot case.

`deltaw` is the interval for the numerical derivative $\frac{\partial\Sigma(\omega)}{\partial\omega}$ in EQ.8. We calculate $\langle\psi^{\mathbf{k}n}|\Sigma(\epsilon^{\mathbf{k}n}+$`deltaw`$)|\psi^{\mathbf{k}n}\rangle$ and $\langle\psi^{\mathbf{k}n}|\Sigma(\psi^{\mathbf{k}n}-$`deltaw`$)|\psi^{\mathbf{k}n}\rangle$ in addition to $\langle\psi^{\mathbf{k}n}|\Sigma(\epsilon^{\mathbf{k}n})|\psi^{\mathbf{k}n}\rangle$. From these values, we can calcuate two Z (or second-derivative of $\Sigma(\omega)$), as shown in SECU. It will help to see whether the used `deltaw` is O.K. or not.

5. Offset-gamma point.
`Q0P_Choice 0`:1 integer
`Q0P_Choice` gives how to determine the offset gamma points. Initially we take them as
0: 6 points along plat.(default)
1: 6 points along along Ex Ey Ez.
2: 4 points in plat(1:3,1),plat(1:3,2)
3: 2 points in plat(1:3,3)
Then we choose only inequivalent $\mathbf{q}$ points based on the point group symmetry. We found `Q0P_Choice 0`=3 together with BZmesh=2 works for 1 dimentional atomic chain along z-axis. See q0irre.f and search Q0Pchoice(). Obtained offset gamma points is given in a file `Q0P`.

`alpha_offG`: 1 real (a.u.)
`alpha_offG` corresponds to α in EQ.48. `alpha_offG`=1d0 is usually good in the sense that it seems to be almost a limit at $\alpha\to 0$. So you can usually fix it as `alpha_offG`=1d0, and check the convergence as for `n1n2n3`.

`WgtQ0p` 1 real (in a.u.) (default=0.01). effective only for BZmesh=2
`WgtQ0p` is the total weight for the offset gamma in the case of `BZmesh`=2 (it is the ratio to the weight for regular mesh point as 1/(n1*n2*n3).). It is usually OK to take 0.01 (default). In principle, the final result should not depend on `WgtQ0p`. We observed that QPE changes little from `WgtQ0p`=0.01 through 1d-6 in cases. (For rather small `WgtQ0p`, you have to care that the normalization of eigenfunction is good enough. In order to make the normalization better(see file normchk.dia), you have to use larger `QpGcut_psi`.).

Note:In addition, we found that it was necessary to use `LMXA=6` (l in the expantion of eigenfunction in MT) or so was necessary to keep the normarization of eigenfunction rather accurately for Na.

6. core orthogonalization (default=off)

`CoreOrth` 1 logical
If this is on, we enforce cores orthogonalied to valence ϕ and $\dot{\phi}$ (these appear in II-C in Ref.I). This procedure enforce the correct orthogonal condition, thus we have correct behevior for the dielectric function at $\mathbf{q}\to 0$. However, it may deform core functions too much, especially in the case of shallow 3d (or maybe 4d) cores. So we don't recoommend make it on usually,

even though then the orthogonality condition is somehow broken. Anyway you can check wether it affects to results or not by this switch.

7. QP self-consistent GW.
   `iSigMode` 1 integer (nodefault).
   This is required for semi self-consistent GW calculation by a script **gwsc**.

   1: Use $\Sigma_{nn'}(E_F) + \delta_{nn'}(\Sigma_{nn'}(\epsilon_n) - \Sigma_{nn'}(E_F))$ (EQ.11 mode-B).

   3: Use $\mathrm{Re}\frac{\Sigma_{nn'}(\epsilon_n) + \Sigma_{nn'}(\epsilon_{n'})}{2}$ (EQ.10 mode-A).

   5: Use $\delta_{nn'}\Sigma_{nn}(\epsilon_n)$ (Eigenvalue-only self-consistency).

   See `/gwsrc/sxcf_fal2.sc.F`. In order to do QS$GW$by **gwsc**, you have to set some options in `ctrl.*` so that **lmf** can read the generated Σ by my GW code (sigm file). It is detailed in the GW driver manual in Mark's lmf, and also see Section 14.

8. Others
   `KeepEigen` 1 logical (default=on)
   `KeepPPOVL` 1 logical (default=on)
   These are for memory usage. If KeepEigen is on, eigenfunctions (Eigen) are kept in memory during calculation. If KeepPPOVL is on, the overlapping mamtrix (PPOVL) is stored in a memory. If you have not enough memory in your machine, use them off. Then you can save memory usage. However, then we have too frequent accsess to files. So %CPU might get lower. Be careful to use these options.

   `Verbose` 1 integer (default=0) If 0, it gives minimum standard output. If 40 or higher, it shows too much output. (these verbosity control is not well-organized yet).

   `multitet` 3 intgers. (optional) Now only "2 2 2" is allowed.
   If you set "`multitet 2 2 2`", it affects hx0fp0 (to calculate the polarization function Π) thought the tetrahedron weights. When we set this, each tetrahedron is further devided into $2 \times 2 \times 2 = 8$ micro tetrahedrons. Weights from the tetrahedron are calculted as the sum of contributions from these micro tetrahedrons, where we utilize eigenvalues at corners of these micro tetrahedron.

   In other words, this is a technique to include changes of eigenvalues though BZ efficiently under the assumption that the behevior of eigenfunctions are rather smooth.

   However, we found this is not so efficient. So we don't recommend to set this option.

6.2 <QPNT> section

This section is to specify the q points and bands index for which you calculate the QP energies (QPE). An example is

```
<QPNT>
 --- Specify the q and band indeces,  for which we evaluate the self-energy ---
*** all q -->1, otherwise 0;  up only -->1, otherwise 0
          0          0
*** no. states and band index for calculation.
          3
  15 16 17
*** q-points, which shoud be in qbz.  See KPNTin1BZ.
          2
  1     0.0000000000000000     0.0000000000000000     0.0000000000000000
  2     0.2886751358562925    -0.5000000000000000     0.0000000000000000
  3     0.0000000000000000     0.0000000000000000     0.3073140749846343
</QPNT>
```

Numbers are read by free format read(5,*), thus the numbers should be separeted by space. At the next line to the first ***, you have to give two numbers used as flags. Both of them takes 0 or 1. 1st one is whether you calculate QPE for all q points (in IBZ) or not. If it is 1, you calculate QPE for all q. If it is 0, you calculate them only for q points specified within this file. In the case of metal where you want to calculate the Fermi energy for QPE, you need to calculate all the eigenvalues somehow above the Fermi energy (If you put 1, it is safer but too time-consuming). The second number is whether you calculate QPE for both spins or not. It is usually 0. In the case of antiferro material, it should be 1.

From the next line to the second ***, you have to specify the states for which you calculate teh QPE. In this example, you calculate the 3 bands of QPE for 15th, 16th, and 17th eigenfunctions (they are ordered from the bottom).

From the next line to the third ***, you have to specify the q points. The first numbers of each line are dummy. In this case, you calcualte QPE for two q points. The third q point is neglected because 2 is given at first.

When you generate GWinput.tmp, you see all the possible q points are listed (these q points should be a part of the regular mesh points).

In the QSGWmode (gwsc), this section is neglected (then we calculate all QPE on regular mesh points); so its hsfp0_sc part is quite expensive (usually it takes time more than hx0fp0).

Additional Note ————

`QPNT_nbandrange` num1 num2 (two integers).
This overide setting in <QPNT>. (I think this switch still works).

`AnyQ` on (default is off)
If this is on, you can spefify any Q point which is not on the mesh point. For the purpose, we need to prepare eigenfunctions at extra **k** points. But it is automatic. In order to make the computaiton efficient. Even in this case, from the computational view, it is better to choose **q** on the two times finer devided mesh (or three times finer devided **k** mesh). This is used for Fig.6 in Phys. Rev. B 74, 245125 (2006).

6.3 set QPNT for eps mode

For eps modes (scripts `eps_*`, which are for linear responses. See Sec.17), you have to specify q point in the following ways.

1. `QforEPSIBZ` on
Then all Q point in IBZ are used.

2. Use section as

```
<QforEPS>
0d0 0d0 0.01d0
0d0 0d0 0.02d0
0d0 0d0 0.04d0
0d0 0d0 0.08d0
</QforEPS>
```

In addition, you can specify Q points as

```
<QforEPSL>
0d0 0d0 0d0   1d0   0d0  0d0 8
0d0 0d0 0d0  .5d0  .5d0  0d0 8
</QforEPSL>
```

This is along the line— 8 point along the line (not left-end q; so omitting 0 0 0). The first line means line (0d0 0d0 0d0)—(1d0 0d0 0d0) is devided to 8. So we have 7 points, (0.125 0 0), (0.25 0 0),... (1 0 0).

6.4 <PRODUCT_BASIS> section

This section is to define product basis to expand W and so. Numbers are read by free format read(5,*),thus the numbers should be separeted by space. The line number in this section is meaningful (you can not add comment lines).

```
<PRODUCT_BASIS>
 tolerance to remove products due to poor linear-independency
  0.100000D-04 ! =tolopt; larger gives smaller num. of product basis. See lbas and lbasC, which ar
 lcutmx(atom) = maximum l-cutoff for the product basis.  =4 is required for atoms with valence d,
  4  3
  atom   l  nnvv  nnc ! nnvv: num. of radial functions (valence) on the augmentation-waves, nnc: n
    1    0    2    3
    1    1    2    2
    1    2    3    0
    1    3    2    0
    1    4    2    0
    2    0    2    1
    2    1    2    0
    2    2    2    0
    2    3    2    0
    2    4    2    0
  atom   l    n  occ unocc  ! Valence(1=yes,0=no)
    1    0    1    1    1   ! 4S_p  -----
    1    0    2    1    0   ! 4S_d
    1    1    1    1    1   ! 4P_p
    1    1    2    0    0   ! 4P_d
    1    2    1    1    1   ! 4D_p
    1    2    2    0    0   ! 4D_d
    1    2    3    1    1   ! 3D_l
    1    3    1    0    1   ! 4f_p
    1    3    2    0    0   ! 4f_d
    1    4    1    0    0   ! 5g_p
    1    4    2    0    0   ! 5g_d
    2    0    1    1    1   ! 2S_p  -----
    2    0    2    0    0   ! 2S_d
    2    1    1    1    1   ! 2P_p
    2    1    2    0    0   ! 2P_d
    2    2    1    1    1   ! 3d_p
    2    2    2    0    0   ! 3d_d
    2    3    1    0    1   ! 4f_p
    2    3    2    0    0   ! 4f_d
    2    4    1    0    0   ! 5g_p
    2    4    2    0    0   ! 5g_d
  atom   l    n  occ unocc  ForX0 ForSxc ! Core (1=yes, 0=no)
    1    0    1    0    0      0    0    ! 1S -----
    1    0    2    0    0      0    0    ! 2S
    1    0    3    0    0      0    0    ! 3S
    1    1    1    0    0      0    0    ! 2P
    1    1    2    0    0      0    0    ! 3P
    2    0    1    0    0      0    0    ! 1S -----
</PRODUCT_BASIS>
```

• This section is read in the free format in fortran. So, e.g., `0.01` works as same as `0.10000D-01`. The line order is important (you have to keep the order given by GWinput.tmp). Be careful atom atom id—lmf may re-order it and pass it to gw code. Look into LMTO file (generated by **mkGWIN_lmf2**); which contains crystal structure information after such re-ordering by lmf. I used `!` to make clear that things after `!` are comments. But `!` is not meaningful – just the expected numbers of datas separeted by blank(s) are read for each line from the begining of lines.

• The real number in second line `tolerance` is to remove the poorly linear-independent product basis. If multiple numbers are specified, it means `tolerance` for each atoms.

You can also use `<PBASMAX>` section to override this setting. It is given as

```
<PBASMAX>
1  5 5 5 3 3
2  5 5 3 2 3
3  3 3 2 2 2
</PBASMAX>
```

The first numebr is for atom index (fixed), and other are product basis for each l channel.

• The integer numbers in 4th line `lcutmx` gives the maximum angular momentum l for the procduct basis for each atomic site. In our experience, `lcutmx`=4 is required when the semi-core (or valence) $3d$ electons exist and we want to calculate the QP energies of them.

• Keep a block starting from " atom l nnvv nnc ..." as it originally generated in GWinput.tmp. It just shows that how many kinds of radial functions for cores and valence electrons for ecah atom and l. `nnvv`=2 in the case of ϕ and $\dot{\phi}$; `nnvv`=3 in the case to add the local orbital in addition.

• There are two blocks after the line "`atom l n occ unocc :Valence(1=yes, 0=no)`' and after "`atom l n occ unocc ForX0 ForSxc ! Core (1=yes, 0=no)`'. These are used to choose atomic basis to construct the product basis. The product basis are generated from the products of two atomic basis.

GWinput.tmp generated by **mkGWIN_lmf2** contains labels on each orbitals as `4S_p`, `4S_d`, `4P_p`... Here `4S_p` is for ϕ_{4s}; `4S_d` for $\dot{\phi}_{4s}$; `3D_l` for $\phi^{\rm local}_{3d}$. Capital letter just after the principle-quantum number means the orbital is used as 'Head of MTO'; lowercase means just used only as the 'tail of MTO'.

The swithces for columns labeled as `occ` and `unocc`. take 0 (not included) or 1 (inclueded). With the switch, we can construct two groups of orbitals, `occ` and `unocc`. In this sample GWIN_V2 as for atom 1, $\{\phi_{4s}, \dot{\phi}_{4s}, \phi_{4p}, \phi_{4d}, \phi^{\rm local}_{3d}, \phi^{\rm core}_{3s}, \phi^{\rm core}_{3p}\}$ consist the group `occ`, and $\{\phi_{4s}, \phi_{4p}, \phi_{4d}, \phi^{\rm local}_{3d}, \phi_{4f}\}$ consists the group `unocc`. So the any product of combinations $\{\phi_{4s}, \dot{\phi}_{4s}, \phi_{4p}, \phi_{4d}, \phi^{\rm local}_{3d}, \phi^{\rm core}_{3s}, \phi^{\rm core}_{3p}\}\times \{\phi_{4s}, \phi_{4p}, \phi_{4d}, \phi^{\rm local}_{3d}, \phi_{4f}\}$ are inclueded as for the basis of the product basis. As for atom 2, $\{\phi_{2s}, \phi_{2p}, \phi_{3d}\} \times \{\phi_{2s}, \phi_{2p}, \phi_{3d}, \phi_{4f}\}$ are included.

• Each line of the last section of `Product BASIS` forms

```
atom   l    n  occ unocc   ForX0 ForSxc :CoreState(1=yes, 0=no)
  1    2    1    A    x      B    C
```

At first you have to understand the concept of CORE1 and CORE2 in EQ.35 Ref.I. However, in our recent calculations, we do not use "CORE2" generally. So, in such a case, set `A=B=C=0`. And treat shallow cores (above Efermi−2Ry or so) as valence electron by "local orbital method" in lmf.

[(Note: you can skip here if you don't use CORE2.)

Each of `A,x,B,C` takes 0 or 1. There are some possible combination of these switches;

1. If you take `( A x B C )= (1 0 1 1)`, then the core is included in core2. In other words, this core is treated in the same manner of the valence electron.
2. If you take `( A x B C )= (0 0 0 0)`, then the core is included in core1. The (exchange only) self-energy related to this core is included in `SEXcore`.
   `C` is the key switch which determine whether it is included in core1 or core2. There could be another option.
3. If you take `( A x B C )= (1 0 0 1)`. This core is in core2. But it is not included in the calculation of D and W. This core is only included for SEX and SEC calculations.

These three kinds of choices are reasonable ones but we can consider some another choice. In the following, we show how these switches (`A,B,C`) affect executions called from **gw_lmfh** (esentially as same as **gw_lmf**).

- **hbasfp0**(mode 3) :Product basis for exchange due to core.
  We include the C=0 cores as a part of the product basis as if A=1 x=0.
- **hsfp0**(mode 3): exchange mode for core.
  Σ_x only due to the C=0 cores are calculated.
- **hbasfp0** (mode1): Product basis.
  Only see the switch A and x. The product basis is generated from (occupied $\times$ unoccupied), where A=1 core is included as one of the occupied basis.
- **hsfp0** (mode 1): exchange mode.
  Only see the switch C. Σ_x due to valence and due to C=1 cores are calculated.
- **hx0fp** (mode 1): $W - v$ calculation.
  Only see the switch B. W is calculates using all the valence and B=1 cores.
- **hsfp0** (mode 2): correlation mode.
  Only see the switch C. Σ_c due to valence and due to C=1 are calculated.

• After you perform gw_lmfh or anything, you find output files lbas by **hbasfp0** (mode1), and/or lbasc by **hbasfp0** (mode3) for core. These contains inportant informations about how many and how product basis are chosen. E.g. 'grep nbloch lbas' shows how many product basis are used in the calculations.

6.5 ANFcond

This file is used in **hx0fp0** in the calculation of $W - v$ (or rather Π in the program) to specify the antiferro condition.

Note : Now only for the case that (a translation vector + spin flip) is a symmetry operation.

This should be given by hand. For the cases of not antiferro, this file should not exist. Even if ANFcond does not exists for antiferro case, **hx0fp0** works but it requires about two time computational efforts.

```
The existence of this file means the Antiferro condition is used for x0k
Product basis B({\bf r}-{\bf a}) is translated to B({\bf r}-{\bf a}-Af})= B({\bf r}-{\bf a}'-T_0})
 1d0  1d0  1d0        ! Af=Antiferro translation vector in Cartesian.
 1  2
 2  1
 3  4
 4  3
```

The first line specifies the Antiferro translation vector. From the second line, we specify that atom i in the primitive cell is mapped to what atom $j(i)$ in the cell with the opposite spin by the translation. In this case, $j(1) = 2, j(2) = 1, j(3) = 4, j(4) = 3$. You have to be careful as for the true atomic position used in the GW calculations can be different from the given atomic positions in ctrl.MnO. The true atomic positions is written in LMTO.

In the case of one-shot GW (gw_lmf and gw_lmfh), it may be better to set "up only" QPE, so that you only calcuate QPE of up spins at the same time.

In the case of gwsc, we just calculate QPE for up spins automatically (QPNT section is neglected).

7 Main Output Files

7.1 QPU

This is the central output[2] in human format. It is like this

```
 ================================================================
  quasiparticle energies MAJORITY
 ================================================================
E_shift=  0.4263273221017709D+00  0.6075150850568627D+00  0.7046628446164018D+00 eV

    q      state SEx   SExcore SEc    vxc   dSE  dSEnoZ  eLDA    eQP  eQPnoZ   eHF  Z   2Z*Simg ReS(elda)
0.0 0.0 0.0 1  -29.56  -1.97  10.40 -20.22 -0.52  -0.90 -19.08 -19.42 -19.71 -30.81 0.58 0.95   -21.12
0.0 0.0 0.0 2  -30.52  -2.24  10.09 -21.53 -0.70  -1.14 -18.06 -18.58 -18.93 -29.72 0.61 0.96   -22.66
0.0 0.0 0.0 3  -20.67  -1.87   5.97 -16.85  0.19   0.28  -7.20  -6.83  -6.65 -13.32 0.67 0.66   -16.57
  ...
```

From the 6h line, we have the eigenvalue datas. All of the unit of energy is in eV. We should note that the zerolevel of these values `eLDA eQP eQPnoZ` can be changed by hqpe. This `eLDA - E_shift` are the eigenvlaues realtive to a Fermi energy determined by the smering method. Detailed value of `eLDA` is in TOTE2.UP. Detailed value of `eLDA- E_shift` is in TOTE.UP.

`q` : $\mathbf{k}$ vector
`state`: Band index n, which is from the lowest eigenvalue (not include cores).
`SEx`: $= = \langle\Psi_{\mathbf{k}n}|\Sigma_{\rm x}^{\rm core2+valence}(\mathbf{r},\mathbf{r}')|\Psi_{\mathbf{k}n}\rangle$
`SExcore`: $= \langle\Psi_{\mathbf{k}n}|\Sigma_{\rm x}^{\rm core1}(\mathbf{r},\mathbf{r}')|\Psi_{\mathbf{k}n}\rangle$
`SEc`: $= \langle\Psi_{\mathbf{k}n}|\Sigma_{\rm c}^{\rm core2+valence}(\mathbf{r},\mathbf{r}',\epsilon_n(\mathbf{k}))|\Psi_{\mathbf{k}n}\rangle$
`vxc`: LDA exchange correlation energy. $\langle\Psi_{\mathbf{k}n}|V_{\rm xc}^{\rm LDA}([n_{\rm total}],\mathbf{r})|\Psi_{\mathbf{k}n}\rangle$
`dSE`: $Z_{n\mathbf{k}}\times$ `dSEnoZ`
`dSEnoZ`: $\langle\Psi_{\mathbf{k}n}|\Sigma_{\rm x}^{\rm core1}(\mathbf{r},\mathbf{r}')+\Sigma_{\rm xc}^{\rm core2+valence}(\mathbf{r},\mathbf{r}',\epsilon_n(\mathbf{k}))|\Psi_{\mathbf{k}n}\rangle - \langle\Psi_{\mathbf{k}n}|V_{\rm xc}^{\rm LDA}([n_{\rm total}],\mathbf{r})|\Psi_{\mathbf{k}n}\rangle$
$=$ `SEx + SExcore + SEc - vxc`
`eLDA`: LDA eigenvalues. $\epsilon_n(\mathbf{k})$
`eQP`: QP energy. $\epsilon_n(\mathbf{k})$+`dSE`
`eQPnoZ`: QP energy without Z. $\epsilon_n(\mathbf{k})$+`dSEnoZ`
`eHF`: HF energy of 1st itteration. $\epsilon_n(\mathbf{k})$+`SEx + SExcore -vxc`
`Z`: Z factor. $Z_{n\mathbf{k}}$
`2Z*Simg`: Quesi-particle life time. $2Z_{n\mathbf{k}}\times{\rm Im}\langle\Psi_{\mathbf{k}n}|\Sigma_{\rm c}^{\rm core2+valence}(\mathbf{r},\mathbf{r}',\epsilon_n(\mathbf{k}))|\Psi_{\mathbf{k}n}\rangle$
(Is this really the usual definition of the life time?—don't believe me)
`ReS(elda)`: ${\rm Re}\langle\Psi_{\mathbf{k}n}|\Sigma_{\rm x}^{\rm core1}(\mathbf{r},\mathbf{r}')+\Sigma_{\rm xc}^{\rm core2+valence}(\mathbf{r},\mathbf{r}',\epsilon_n(\mathbf{k}))|\Psi_{\mathbf{k}n}\rangle$

7.2 XCU

LDA exchange-correlation. Detailed data of above `vxc`.

7.3 SEXU

Exchange part of the self-energy due to valence electrons. Detailed data of above `SEx`.

7.4 SEXcoreU

Exchange part of the self-energy due to core. Detailed data of above `SExcore`.

[2]Note that QPU also implies QPD and so on. U is for up D is for down spins.

7.5 SECU

Correlation part of the self-energy. Detailed data of above `SEc`.

7.6 TOTE.UP (TOTE.DN)

This is a central output. It contains LDA and QP energies. These values are realtive to a Fermi energy determined by the smering method. It contains two kind of QP energies `QP QPnoZ`. The first line contains the Fermi energy in Ry determined by the smering method. It is also shown in the end of DOSACC.lda.

7.7 TOTE2.UP (TOTE2.DN)

This is a central output. It contains zerolevel shifts from TOTE.UP. The first line contains the Fermi energy in eV (= the Fermi energy in TOTE.UP but it is in Ry) and three energy shifts `E_shift`, which are the same values in the 4th line of QPU.

Note that all *.chk files are just to check calculations (not read in by successive executions).

7.8 DOSACC.lda

This lists all the eigenvalues in acendent order. States with almost the same eigenvalues are degenerated states. The 4th colmn contains number of electrons up to the eigenvalue.

7.9 DOSACC2.lda

This is similar with DOSACC.lda. But we remove the degeneracy.

7.10 Core_ibas*_l*.chk

Used core eigenfunctions.

7.11 VXCFP.chk

This contains eigenvalues and $\langle\psi_{\mathbf{k}n}|V_{\mathrm{xc}}|\psi_{\mathbf{k}n}\rangle$ in both units, Ry and eV. See below.

7.12 The Fermi energies in this GW code.

We mainly have two kinds of Fermi energy $E^{\mathrm{smear}}_{\mathrm{FEERMI}}$ $E^{\mathrm{tetra}}_{\mathrm{FEERMI}}$.

1. At first eigenvalues given by **lmfgw** is in VXCFP.chk. You can see

   ```
   %head VXCFP.chk
   ### LDA exchange correlation ###
   #   qvec                 ikp iband    eigen       VXC(ntotal)     VXC(nvalence)    eigen(eV)    VXC(ntot
     0.0000  0.0000  0.0000  1  1     -0.96932423     -1.00727912      0.00000000    -13.18843159    -13.70
   ...
   ```

   These are raw values. TOTE contains the eigenvalues but relative to a Fermi energy $E^{\mathrm{smear}}_{\mathrm{FEERMI}}$ which is determined by the smearing method. It is also shown at the top part of output files lsx_sf and lsc_sf. And you also see the value at the end of DOSACC.lda.

   This is the head of TOTE.UP;

```
%head TOTE.UP
          43           8  8.520283353474250E-003
   0.0000000   0.0000000   0.0000000    1   1  -0.1330435686590073D+02 -0.1322984339282777D+02 -0.131922
   0.0000000   0.0000000   0.0000000    2   1  -0.7555264915356062D+00 -0.6267595395613325D+00 -0.600948
...
```

Here $E^{\rm smear}_{\rm FEERMI}$=8.520283353474250E-003. From the second lines, they are LDA eigenvalues and QP energies (Z included and Z=1); they are relative to the $E^{\rm smear}_{\rm FEERMI}$.
-13.18843159 eV - $E^{\rm smear}_{\rm FEERMI}$(which should be translated into in eV) = -0.1330435686590073D+02 eV. Here -13.18843159 is the value in VXCFP.chk shown above.

2. There is the another Fermi energy $E^{\rm tetra}_{\rm FEERMI}$, which is used by mode 11 (or mode 1) of hx0fp0 in **gw_lmfh**. It is determined by heftet and stored in EFERMI.

3. **hqpe** gives TOTE2.UP and QPU. They contains the same values. You can see `eLDA  eQP  eQPnoZ  Z` not only in QPU but also in TOTE2.UP. At top lines of TOTE2.UP, you see

```
%head TOTE2.UP
       43        8  0.1159252712507000D+00  0.7555207081466229D+00  0.6267572296579150D+00  0.600946717
   0.0000000   0.0000000   0.0000000    1   1  -0.1254883615775411D+02 -0.1260308616316985D+02 -0.125913
   0.0000000   0.0000000   0.0000000    2   1  -0.5783388983382487D-05 -0.2309903417430093D-05 -0.161372
   0.0000000   0.0000000   0.0000000    3   1  -0.1369098933889923D-05 -0.6195200397129952D-07  0.204742
   0.0000000   0.0000000   0.0000000    4   1   0.0000000000000000D+00  0.0000000000000000D+00  0.000000
..
```

,where a number in first line $E^{\rm smear}_{\rm FEERMI}$= 0.1159252712507000D+00 eV =8.520283353474250E-003 Ry, the same as the previous one. This is a case when you did hqpe with augment 4 (it means we set the 4th-band eigenvalue zero). Another 3 values in the first line are shifts from TOTE. Shown eshift(eLDA) = 0.7555207081466229D+00 eV. E.g., the second line shows -0.1254883615775411D+02 eV= -0.1330435686590073D+02(in TOTE) + eshift(eLDA) eV.

When you do **hqpemetal**, three shfifts at the first line in TOTE2.UP is determined so as to give the eigenvalues relative to the Fermi energies shown in EFERMI, EFERMI.QP1, and EFERMI.QPz=1. These are Fermi energies by tetrahedron method.

As for **gwband_lmf**, it recalculates eigenvalues for all **q** along SYML. Then
the default "zerolevel" = $E^{\rm smear}_{\rm FEERMI}$ - eshift(lda). Because the eigenvalues given by this bandmode are presumably the same, we have
Shown LDA eigenvalue
= -13.18843159(raw data by band mode—same as that in VXCFP.chk) - zerolevel
= (-13.18843159 - EFERMIsmear) + eshift(lda).
= -0.1330435686590073D+02(this is in TOTE.UP) + eshift(lda)
= -0.1254883615775411D+02(this is in TOTE2.UP).

It means that values in TOTE2.UP recovers. But if raw data by band mode is different from it, these is a trouble. It does not recover the values in TOTE2.UP(=QPU).

As for the QPE, we calculate the difference from LDA values in TOTE2.UP at first, and add the difference to the Shown LDA eigenvalue.

8 Pre-Preparation script:mkGWIN_lmf2 and their I/O Files

—This section might be a little too old. actually this fit to previous version **mkGWIN_lmf**—

The script **mkGWIN_lmf2** is now get to be a bit confused due to the historical reason. However, essentials are simple, and we calls three executions in this script as

echo 0 | **lmfgw si**

echo 1 | **gwinit**

echo -100 | **qg4gw**

. We explain each by each.

8.1 echo 0|lmfgw

— makes SYMOPS LATTC CLASS NLAindx.

Input files

• GWIN0 : This is a file, which contains your supplied n1 n2 n3 when you invoke the script. This file is generated within the script (as "here document").

• ctrl.si : Master input file of lmf calculation. This is in the case of Si.

Output files

• LATTC : contains the information of primitive translation vectors, lmxa and konf.
A sample is,

```
 10.26d0         ! alat       = lattice constant in a.u.
  0d0 .5d0 .5d0 ! plat(1:3,1)= 1st primitive translation vector in alat.
 .5d0 .0d0 .5d0 ! plat(1:3,2)= 2nd ...
 .5d0 .5d0 .0d0 ! plat(1:3,3)= 3rd ...
 -1d10 ! QpGcut_psi = maxmum of |q+G| in a.u.
-----------------------------------------
 2   4 ! nbas lmxax (max l for augmentation)
-----------------------------------------
-- ibas lmxa konf(s) konf(p) konf(d)... ----
    1   4   3   3   3   4   5
    2   4   3   3   3   4   5
```

where negative `QpGcut_psi` in the 5th line means dummy. True `QpGcut_psi` is given in GWIN0. After "`-- ibas lmxa konf(s) konf(p) konf(d)... ----`" , you see numbers "`3 3 3 4 5`" for `ibas=1`. These mean $3s, 3p, 3d, 4f, 5g$, which specify the lowest principle quantum numbers of valence electrons. In other words, cores are $1s, 2s, 2p$ for `ibas=1` in this case.

• SYMOPS : The point group operations. It is written through

```
open(file='SYMOPS')
write(ifi,*) ngrp
do ig = 1, ngrp
  write(ifi,*) ig
  do i=1,3
    write(ifi,"(3d24.16)") symops(i,1:3,ig)
  enddo
enddo
close(ifi)
```

• CLASS : This is like this;

```
1  1
2  1
3  2
4  2
```

The first numbers of each line are atomic-site number in the primitive cell. (It should be the same as the line number). The second numbers of each line are atomic classes for each atomic-site.

• NLAindx : This file contains indexes ($p_{\rm valence}, l, a$) for orbitals in the MT.

• Idima : Size of MTO for each atomic site. (this is used only from **hqpe_sc**—scGW mode).

8.2 gwinit

— Get GWIN_V2.tmp and QPNT.tmp

Input files

- GWIN0 :
- LATTC :
- SYMOPS :
- NLAindx :

Output files

- GWIN_V2.tmp : A part of GWinput.tmp
- QPNT.tmp : A part of GWinput.tmp
- (QPNTforSYML.tmp) : template of QPNT to get best bandplot.

If SYML exist, **gwinit** gives also a templete QPNTforSYML.tmp suitable for such SYML. Here SYML specify how to plot the energy band. See explanation for **bandplot** script.

Note that LATTC SYMOPS CLASS NLAindx are overwritten when you execute **gw_lmfh** because we repeat **echo 0|lmf** at the head of **gw_lmfh**.

8.3 echo -100|qg4gw

— Generate GWinput.tmp

Input files

- GWIN0 : (copy of GWIN0.tmp by **gwinit**)
- GWIN_V2 : (copy of GWIN_V2.tmp by **gwinit**)
- QPNT : (copy of QPNT.tmp by **gwinit**)

Output files

- GWinput : (this is copied to GWinput.tmp)

This command "echo -100|qg4gw" is a file convertor from these two files into GWinput. And it is copied to GWinput.tmp. (**mkGIN_lmf** keeps GWinput if it exist before you invoke it.).

9 One-shot *GW* main script: gw_lmfh and I/O Files

— this may be a little wrong. Let me know if something wrong.—
The main part of the script **gw_lmfh** is

```
############### preparatory gw stage #################
echo 0 |$nfpgw/lmfgw  $argv[1] > llmfgw00
echo 1 |$nfpgw/qg4gw           > lqg4gw

#eigenvalues for micro-tetrahedron method. This if-endif block is only for multitet mode.
if(-e Qmtet) then
  mv Qmtet Qeigval
  echo 5 |$nfpgw/lmfgw  $argv[1] > llmfgw_eigval
  mv eigval eigmtet
endif


echo 1 |$nfpgw/lmfgw  $argv[1] > llmfgw01
@ exinfo = `tail -3 llmfgw01 |grep Exit |head -1 |awk '{print $2}'`
if($exinfo == 0 ) then
  echo " OK! lmfgw mode=1 "
else
  echo `tail -3 llmfgw01 `
endif
echo $argv[1]|$nfpgw/lmf2gw    > llmf2gw


#################### Main GW stage ######################
$nfpgw/rdata4gw_v2      >lrdata4gw_v2


# -- get EFERMI for hx0fp0
echo 1|$nfpgw/heftet      >leftet

# -- hchknw only calculate NW, which contains the number of nw corresponding to <QPNT> -----
echo 0|$nfpgw/hchknw         >lchknw


###### Core1 exchange self-energy ######
# -- product basis for core
echo 3|$nfpgw/hbasfp0 >lbasC
# -- Coulobm matrix
echo 0|$nfpgw/hvccfp0        >lvccC


# -- the self energy from core1
echo 3|$nfpgw/hsfp0   >lsxC


###### Valence part of the self-energy ######
echo 0|$nfpgw/hbasfp0  >lbas
# -- Coulobm matrix
echo 0|$nfpgw/hvccfp0  >lvcc

# -- Sergey.F the exchange self energy from valence core2+valence elctrons
echo 11|$nfpgw/hsfp0   >lsx_sf

# -- Sergey.F the screened coulom interaction
echo 11|$nfpgw/hx0fp0  >lx0_sf
# -- Sergey. F the correlation self-energy from valence core2+valence elctrons
echo 12|$nfpgw/hsfp0   >lsc_sf

# -- Make summary
echo 0|$nfpgw/hqpe    >lqpe
```

,where **echo 0|lmfgw** was already explained in the previous section; `$nfpgw` contains path to the execution binaries. As said in xxx, it consists of 2 stages. In order to see how *GW* work, it will be useful to do these step by step by hand.

9.1 echo 0| qg4gw

— makes **q** points, and **G** vectors for these **q**. (**q** was **k** in previous sections.) Main routine of qg4gw is `main/qg4gw.m.f`. See `exe/makefile`. Input files

• GWIN0 :
• LATTC :
• SYMOPS :

Output files

• QGpsi : (bin) q and G vector for the eigenfunction.
• QGcou : (bin) q and G vector for the Coulomb matrix
• Q0P : offset-Γ points which are the replacement of the q=0 points. See section**??**.
• QIBZ : q points in the Irreducible BZ.
• BZDATA : (bin) BZ data for integration (include tetrahedrons if necessary). See e.g. `main/hx0fp0.f` and search "`call read_BZDATA`", which is a readin routine of this file defined in `rwbzdata.f`.
• KPTin1BZ.mkqg.chk : list of q in the 1st BZ.

9.2 echo 1|lmfgw

— Calculate eigenfunctions, eigenvalues and $\langle\psi|H_{\rm KS}|\psi\rangle$

Input files

• ctrl.si :
• rst.si : Restart file of the nfp calculation. It contains the LDA potential.
• QGpsi,QGcou,Q0P :

Output files

• gwa.si : (bin) atomic data
• gwb.si : (bin) band data
• gw1.si : (bin) $\langle\psi|H_{\rm KS}|\psi\rangle$
• gw2.si : (bin) $\langle\psi|H_{\rm KS}-V_{\rm xc}(n_{\rm total})|\psi\rangle$.
• vxc.si,evec.si : (bin) Only used for scGW mode (hqpe.sc.m.f as "v_xc" and "evec").

vxc.si contains $\langle\psi|V_{\rm xc}(n_{\rm total})|\psi\rangle$ including off-diaonal part. evec.si contains eigenfunctions.

• normchk.si : norm check (only for check) This is like this

```
> head -20 normchk.si
#       IPW          IPW(diag)   Onsite(tot)   Onsite(phi)      Total
      0.436015      0.805123      0.563972      0.562573      0.999988
      0.339134      0.620353      0.660515      0.656881      0.999649
      0.339133      0.620353      0.660516      0.656882      0.999649
      0.339133      0.620353      0.660516      0.656882      0.999649
      0.507738      0.648515      0.492040      0.487673      0.999778
...
```

This check is sometimes important for debugging and to determine the cutoff parameter `QGcut_psi`. The first line (corresponding to 1st band of 1st q point) means that total normalization almost unity = 0.999988 = 0.436015 + 0.563972. Because we expand the MTO by IPW, the normalization is a bit different from unity, especially for higher bands. You can see that it get closer to unity for larger QGcut_psi, though it does not reach to unity because of some contribution of the higher angular momentum contribution within MT. [Values of `Onsite(phi)` are wrong in the case when you use local orbital.]

Due to historical reason, data in vxc.si and exec.si and others contains duplicated data.

9.3 lmf2gw

— All the required informations are stored into DATA4GW_V2 and CphiGeig.

Input files

- gwa.si :
- gwb.si :
- gw1.si :
- gw2.si :
- Q0P :
- CLASS :

Output files

- DATA4GW_V2 : (bin) Main data for GW calculations.
  I/O of DATA4GW_V2 is controlled by `gwinput.f`, which contains detailed informtaions.
- CphiGeig : (bin) Eigenfunctions for GW calculations.
- VXCFP.chk : Eigenvalue and Vxc check (only used for check)
  It is like this;

```
### LDA exchange correlation ###
#   qvec              ikp iband    eigen        VXC(ntotal)    VXC(nvalence)      eigen(eV)    VXC(ntotal)(eV)  VXC(nvalence)
0.0000  0.0000  0.0000  1  1     -0.68505346     -0.91850436      0.00000000     -9.32070032    -12.49698668      0.00000000
0.0000  0.0000  0.0000  1  2      0.19292662     -0.99853478      0.00000000      2.62492096    -13.58586453      0.00000000
0.0000  0.0000  0.0000  1  3      0.19292763     -0.99853469      0.00000000      2.62493477    -13.58586334      0.00000000
0.0000  0.0000  0.0000  1  4      0.19292777     -0.99853461      0.00000000      2.62493664    -13.58586222      0.00000000
 ...
```

Here `VXC(nvalence)` is not used now. The eigenvalue in `eigen` is in Ry.

— This is the end of (2) the preparation stage. —

From here, (3) the main stage. it is independent how you prepared the eigenfunctions.

9.4 rdata4gw_v2

— Read DATA4GW_V2 and some files, and decompose it into files required in the following *GW* steps.

Input files

- GWinput :
- DATA4GW_V2 :
- CphiGeig :
- QGpsi :
- QGcou :
- Q0P :
- QIBZ :
- SYMOPS : points group operations.

Output files

- hbe.d : datasize
- Core_ibas*_l*.chk : core eigenfucntions just for check.
- LMTO : basic date for the crystal.
- EVU : (bin) valence eigen value
- ECORE : core data and core eigenvalues
- CPHI : (bin) Coefficients of eigenfunctions as for the atomic-like argumentation waves in MTs'.
- GEIG : (bin) Coefficients of eigenfunctions as for IPW.
- PHIVC : (bin) All the radial functions.
- @MNLA_CPHI : index set for CPHI. This is not reffered just a check write.

• @MNLA_core : index set for core. This is not reffered just a check write.
• VXCFP : (bin) this is for diagonal elements of $V_{\rm xc}^{\rm LDA}(n_{\rm total})$.
• PPOVL : (bin) Overlap matrix of IPW. not exactly the the overlap matrix. see around line 500 in rdata4gw.m.f
• HVCCIN : (bin) Required inputs for hvccfp0. Informations in this files.
• NQIBZ : q point info. Only used for paralell test mode.
• normchk.dia : Norm check. These numbers should be almost the same as those in normchk.si

These files are input for the folloing steps. The name of file *foo*U means that it relates to up-spin. We have *foo*D files in the case of spin-poralized calculation with `nspin=2`.

9.5 echo 1|heftet

— Get the Fermi energy EFERMI by tetrahedron method. It is used in **hx0fp0**.

Input files

• EVU :
• BZDATA :
• GWinput :
• ECORE : (dummy)
• SYMOPS : (dummy)
• LMTO :
• hbe.d :

Output files

• EFERMI : contains Fermi energy given by the tetrahedron method. It is used in **hx0fp0** but not in **hsfp0**.
• DOSACC.lda,DOSACC2.lda : They are lists of the all the eigenvalues from the bottom. DOSACC2.lda is a list to show only the un-degenerated eigenvlaues. They are just check write. But it is an indicater for you to determi `esmr` in GWinput.

9.6 hchknw

— Calculte the required number of ω points along real axis.
This NW is not essentially used in **gw_lmfh** (but required as a dummy file). Only used in **gw_lmf**.

Input files

• BZDATA :
• GWinput :
• ECORE : (dummy)
• SYMOPS : (dummy)

Output files

• NW : contains number of ω points.

9.7 echo 3|hbasfp0

— Make product basis.

Mode 3 is for the core states. It generate a product basis on each MT suitable to expand to calculate the exchange part due to core. See explanations for the input file of GWinput.

Input files

• LMTO :
• PHIVC :
• GWinput :

Output files

• BASFP* : (bin) Product basis functions

• PPBRD_V2_* : (bin) Radial integrals on each MT, symbollycally written as $\int \phi(r)\phi(r)B(r)dr$

• PHIV.chk : Valence radial functions (for check).

9.8 echo 0| hvccfp0

— Calculate the Coulomb matrix in the Mixed basis

Input files

• HVCCIN :

• Q0P :

• BASFP* :

Output files

• VCCFP : The Coulomb matrix expanded in the mixed basis

• Mix0vec : This is used only for dielectric-constant calculation (mode 2 or 3 of **hx0fp0**). This contains the expansion of the plane wave $\exp(i\mathbf{qr})$ in the mixed basis. See Usuda's note.

9.9 echo 3|hsfp0

— Exchange part of the self-energy for the core

Input files

• GWIN_V2,LMTO,ECORE :

• CLASS :

• hbe.d :

• Q0P :

• PPBRD_V2_* :

• CPHI :

• GEIG :

• VCCFP :

• PPOVL :

Output files

• SEXcoreU : The core part of the exchange self-energy for **q** and band index specified in <QPNT>. See 7.

9.10 echo 0|hsfp0

— Make product basis for the valence part.

9.11 echo 1|hsfp0

— Exchange part of the self-energy for the valence part.

Output files

• XCU : The LDA exchange self-energy for **q** and band index specified in <QPNT>. See 7.

• SEXU : The valece part of the exchange self-energy for **q** and band index specified in <QPNT>. See 7.

9.12 echo 11|hx0fp0

— Screened Coulomb interaction W(sergey mode)

Input files

• GWinput, LMTO, ECORE, EVU :

• NW : dummy
• hbe.d :
• Q0P :
• PPBRD_V2_* :
• CPHI,GEIG :
• PPOVL :
• VCCFP :
• ANFcond : (optional) This file is to specify antiferro condition. This should not exist for other cases. This file should be given by hand.

Output files

• WV.d : size of the dielectric function
• WVR : (bin) $W - v$ in the expansion of mixed basis along the real axis
• WVI : (bin) $W - v$ in the expansion of mixed basis along the imaginary axis

9.13 echo 12|hsfp0

— Corrlation part of the self-energy(sergey mode)

Input files

• GWinput, LMTO, ECORE, SYMOPS : These are readin by `genallcf_v3`.
• CLASS, hbe.d, EVU, Q0P :
• PPBRD_V2_* :
• CPHI,GEIG :
• PPOVL :
• WV.d, WVR, WVI :

Output files

• SECU : The corrlation part of the self-energy for $\mathbf{q}$ and band index specified in <QPNT>. See 7.

9.14 echo 0|hqpe

— Summarize the output

Input files

• SEXcoreU,XCU,SEXU,SECU : See 7.

Output files

• QPU : The QP energies and related value summary in human interface. See 7.
• TOTE : The detailed values of the QP energies. See 7.
• TOTE2 : The detailed values of the QP energy. See 7. This is used for **bndplot**.

NOTE: For example, if you do `{echo 4$|$hqpe}{hqpe}`, it just shift zero level of QPE, so that 4th line (counted from top) eigenvalue (in QPU) is to be zero.

10 hqpemetal: Executions and their I/O Files

This script is to determine the Fermi energy after the GW calculation. In order to calculate the Fermi energy for QP, you have to calculate all the QP energies. So you have to set <QPNT> as

```
 --- Specify the q and band indeces,  for which we evaluate the self-energy ---

*** all q -->1, otherwise 0;  up only -->1, otherwise 0
          1           0
...
```

in order to calculate all q points. In addition, you have to be careful that you have calculated all the QP energies greater than the resulting Fermi energy.

The script **hqpemetal** is

```
#!/bin/csh -f
set n = $0
set nfpgw = ${n:h}
echo $nfpgw

#echo $argv[1]
#setenv LMJOB $argv[1]

echo 2|$nfpgw/heftet >lheftet2
echo 3|$nfpgw/heftet >lheftet3
echo 4|$nfpgw/heftet >lheftet4

echo -1|$nfpgw/hqpe  >lqpemetal
```

The main input to **heftet** are TOTE.UP (and TOTE.DN in the case of nspin=2). TOTE.UP contains eigenvalues of LDA, QP energies (both Z included and Z=1 cases). The command `echo 2|$nfpgw/heftet`, mode 2 of heftet, gives the Fermi energy **EFERMI.check** for LDA eigenvalues. It should be the same as **EFERMI**, which had already generated by **heftet** by **gw_lmf**. The command `echo 3|\verb|$nfpgw/heftet`, mode 3 of heftet, and the command `echo 4|$nfpgw/heftet`, mode 4 of heftet, give the Fermi energies EFERMI.QP and EFERMI.QPz=1.

With EFERMI, EFERMI.QP1, EFERMI.QPz=1, together with SEXU and so on, `echo -1|$nfpgw/hqpe` can gives QPU and TOTE2.UP, which include the resulting eigenvalues and QP energies relative to these Fermi energies.

11 gwband_lmf and its I/O Files

— Recently, I don't maintain this so much, so this may not work well. Or need maintenance —. This script is in order to generate eigenvalues and QP energies plot. The script is

```
#!/bin/csh
# --------------------------------
# Get band plot of GW calculations.
#
# Required inputs are
#  SEXU SECU SEXcoreU
#  ctrl.* *.rst *  SYML
#-----------------------------------------
set n = $0
set nfpgw = ${n:h}
echo $nfpgw

#echo $argv[1]

if( (! -e SYML) ||-z SYML) then
  echo --- No SYML file \(it might be size zero\)!
  exit
endif
echo ' '
echo ' 'hqpe \(or hqpemetal\) to fix the zero level is supposed to be already done. OK\?
echo ' '
echo ' '----- We use SYML --from here------------
cat SYML
echo ' '
echo ' '------------------- to  here------------
echo 0 | $nfpgw/lmfgw $argv[1]  >lng00
echo 3 | $nfpgw/qg4gw    >lqg4gw
echo 4 | $nfpgw/lmfgw $argv[1]  >llmfgw04
foreach ext (UP DN)
if(-e LBAND.$ext) then
  echo LBAND.$ext
  cp LBAND.$ext LBAND
  if(-e TOTE2.$ext) cp TOTE2.$ext TOTE2
  $nfpgw/hbndout  >lbndout.$ext
  $nfpgw/bandplot      #This is a script calling bandfp which generate ps file.
  foreach fin (BandLDA BandQP1 BandQP2 BandGWpoint BandQpoint)
    if(-e $fin ) mv $fin $fin.$ext
  end
  foreach fout (BandLDA BandQP1 BandQP2)
    if(-e $fout.ps) mv $fout.ps $fout.$ext.ps
  end
end
if(-e LBAND) rm LBAND
if(-e TOTE2) rm TOTE2
```

At first you have to do hqpemetal, which gives TOTE2.UP in the case of metal. Or do hqpe and give what number of bands as the zerolevel. TOTE2.UP contains the LDA eigenvalues and the QP energies, in addition to the energy shifts. See Sec.7.

• SYML : The required input which specify the symmetry lines for plotting bands. An example is

```
51    0.5 0.5 0.5  0 0 0  T F   L   \gG
51    0 0 0        1 0 0  F F   \gG  X
```

, where "T F" and "F F" means whether we assume the dispersions as flat at these ends. E.g. "T F" in the first line means, we extrapolate the dSE under the assumption that dSE is flat at "0.5 0.5 0.5", but not assume it at "0 0 0". This option was necessary for some cases.

The **gwband_lmf** shown above contains

```
echo 0 | $nfpgw/lmfgw $argv[1]   >lmfgw00
echo 3 | $nfpgw/qg4gw            >lqg4gw
echo 4 | $nfpgw/lmfgw $argv[1]   >lmfgw04
$nfpgw/hbndout                   >lbndout
```

The first line `echo 0|lmfgw` is just to generate LATTC, SYMOPS and CLASS. The new mode echo `echo 3|qg4gw` gives QGpsi which is along the symmetry lines. Then the new mode `echo 4|lmfgw` calculates the LDA eigenvalues along the symmetry lines in SYML with cheking the continuity of each dispersions by calculating $\langle\psi_{\mathbf{k}n}|\{\psi_{\mathbf{k}+\delta\mathbf{k}n'}\}\rangle$, where $\{\psi_{\mathbf{k}+\delta\mathbf{k}n'}\}$ denotes the eigenfuntions map backed to $\mathbf{k}$. The data is written into LBAND (in the dispersion-branch index ordered).

"`$nfpgw/hbndout >lbndout`" generates BandGWpoint, BandQpoint, BandLDA, BandQP1, BandQP2.

- BandLDA : contains LDA eigenvalues along the symmetry lines.
- BandQP1 : conatains the QP energies with Z.
- BandQP2 : conatains the QP energies with Z=1.

These values should the same as the ones shown in QPU files at each $\mathbf{k}$ points in it.

- BandGWpoint : is important because it contains informations which dates in QPU are used for the interpolation.

```
   x-axis      LDA (eV)       QP(eV)         QPnoZ        symline  branch      qx          qy          qz
 0.173205    -5.7017400    -5.5007494    -4.9360496        1        1     0.400000    0.400000    0.400000
```

`symline` means the index of the symmetry lines. `branch` means the dispersion-branch index.

The numbers of points used in the interpolation along the symline is dependent on the devision in SYML. We just use the points if the devided $\mathbf{k}$ points are in agrement with the ones in QPU. So if you use a poor divisions along symmetry line, you can use little number for interpolation. E.g. if you use 50 (this means 49 divisions) instead of 51 as above for Γ to X in the case of Si 444, you will just have Γ and X points for interpolation. So be careful!

You don't need to calculate QP energies for $\mathbf{k}$ points not along the symline because they are not used for interpolation. For given SYML, the recommended <QPNT> section, which includes the q points along lines specified in SYML, is generated as QPNT.forSYML.tmp when you invoke **mkGWIN_lmf** with the SYML file in the same directory.

If **hbndout** can not find all the QP energies required for the interpolation on a branch along a symline, it will give up the interpolation for the branch.

```
$nfpgw/bandplot      #This is a script calling bandfp which generate ps file.
```

where **bandplot** is a script calling `ecal/plot/bandp`, which is a bandplotting routine using PGPLOT in `http://www.astro.caltech.edu/~tjp/pgplot/`

—- This is an example console output when it successfully finish. —-

```
[kotani@dob2 Nitest]$ ~/ecal_sc/fpgw/exec/gwband_lmf ni
/home/kotani/ecal_sc/fpgw/exec
ni

 hqpe (or hqpemetal) to fix the zero level is supposed to be already done. OK?

 ----- We use SYML --from here------------
 51    0.5 0.5000 0.500  0 0.00 0.0  T T  L \gG
 51    0   0.0000 0.000  1 0.00 0.0  T T  \gG X

 ------------------- to here------------
argv: Subscript out of range.                <---- this is not a problem
FORTRAN STOP  OK! qg4gw mode=3 band-plot mode
argv: Subscript out of range.                <---- this is not a problem
LBAND.UP
FORTRAN STOP  OK! bndout for SYML
/home/kotani/ecal_sc/fpgw/exec
0 BandLDA
  What data LDA(0) QP1(1) QP2(2)?
```

```
   0.000000E+00 L
  0.8660250     \gG
  0.8660250     \gG
   1.866025     X
 nl=          68
  goto plotting
  OK! end of plotting
1 BandQP1
  What data LDA(0) QP1(1) QP2(2)?
   0.000000E+00 L
  0.8660250     \gG
  0.8660250     \gG
   1.866025     X
 nl=          68
  goto plotting
  OK! end of plotting
2 BandQP2
  What data LDA(0) QP1(1) QP2(2)?
   0.000000E+00 L
  0.8660250     \gG
  0.8660250     \gG
   1.866025     X
 nl=          68
  goto plotting
  OK! end of plotting
LBAND.DN
FORTRAN STOP  OK! bndout for SYML
/home/kotani/ecal_sc/fpgw/exec
0 BandLDA
  What data LDA(0) QP1(1) QP2(2)?
   0.000000E+00 L
  0.8660250     \gG
  0.8660250     \gG
   1.866025     X
 nl=          68
  goto plotting
  OK! end of plotting
1 BandQP1
  What data LDA(0) QP1(1) QP2(2)?
   0.000000E+00 L
  0.8660250     \gG
  0.8660250     \gG
   1.866025     X
 nl=          68
  goto plotting
  OK! end of plotting
2 BandQP2
  What data LDA(0) QP1(1) QP2(2)?
   0.000000E+00 L
  0.8660250     \gG
  0.8660250     \gG
   1.866025     X
 nl=          68
  goto plotting
  OK! end of plotting
```

Even when TOTE.* are not exist, **gwband_lmf** can generate LDA band plot.

12 Spectrum function of $\Sigma_c(\omega)$ (diagonal part). run-mode 4 of hsfp0

— recently, I have a little improved this function. But not documented yet (just a little memo in fpgw/fpgw_version_log). So you may need to ask me if you need to plot this —.

* Read util/dossig3.F in order to calculate spectrum dos from SEComg.UP and DN.

In order to calculate $\langle\Psi_{\mathbf{q}n}|\Sigma_c(\omega)|\Psi_{\mathbf{q}n}\rangle$, you first need to add a new section which start a label ***** (five astarisk) in <QPNT> section. For example, <QPNT> section for this mode is like this;

```
 --- Specify the q and band indeces,  for which we evaluate the self-energy ---

*** all q -->1, otherwise 0;  up only -->1, otherwise 0
          0          0
*** no. states and band index for calculation.
          2
  4  5
*** q-points, which shoud be in qbz.  See KPNTin1BZ.
          3
  1     0.0000000000000000     0.0000000000000000     0.0000000000000000
  2    -0.5000000000000000     0.5000000000000000     0.5000000000000000
  3     0.0000000000000000     0.0000000000000000     1.0000000000000000


***** ---Specify the q and band indeces for which we evaluate the omega dependence of self-energy ---
   0.01 999   (Ry) ! dwplot omegamaxin(optional)  : dwplot is mesh for plotting.
                    : this omegamaxin is range of plotting -omegamaxin to omegamaxin.
                    : If omegamaxin is too large or not exist, the omegarange of W by hx0fp0 is used.
*** all q -->1, otherwise 0;  up only -->1, otherwise 0
          0          0
*** no. states and band index for calculation.
          1
  4  5
*** q-points, which shoud be in qbz.  See KPNTin1BZ.
          3
  1     0.0000000000000000     0.0000000000000000     0.0000000000000000
  2    -0.5000000000000000     0.5000000000000000     0.5000000000000000
  3     0.0000000000000000     0.0000000000000000     1.0000000000000000
```

Then you have to invoke "**echo 4** | **hsfp0** >*some file*". The program mode 4 of hsfp0 recognize the label ***** as a starting line of this part. As you see, the program first read the numbers 0.01 and 999 at the next line of *****. Then it reads numbers of the next line of ***. This part after ***** has the same structure as the original ones except that you needs to specify the range to plot, dwplot and omegamaxin. The maximun plotting range of ω is limited by the range of ω of $W(\omega)$ given in WVR —the range is almost maximum of $|\epsilon_{\mathbf{k}n} - E_{\rm Fermi}|$, where $\epsilon_{\mathbf{k}n}$ is for the states specified in the first part of <QPNT>. Even when you set omegamaxin very large enough, the rangeof ω is limited by the range of WVR. When you invoke **hsfp0**, you have to supply 4 from console and then you need to supply \ to make reain error for second read(5,*). The obtained plotting data is in SEComg.UP[DN]. whose each line is like this;

```
  iw    band  iq  isp           q              LDA(eV)  omega(eV)  ReSigma(eV) ImSigma(eV)
  -31    4    1    1  0.00000  0.0000  0.0000 -0.3072   -4.21779   3.83216     0.22379
```

. See the next page of plotting example (however, this is from LDA (not consistent, so not good example).

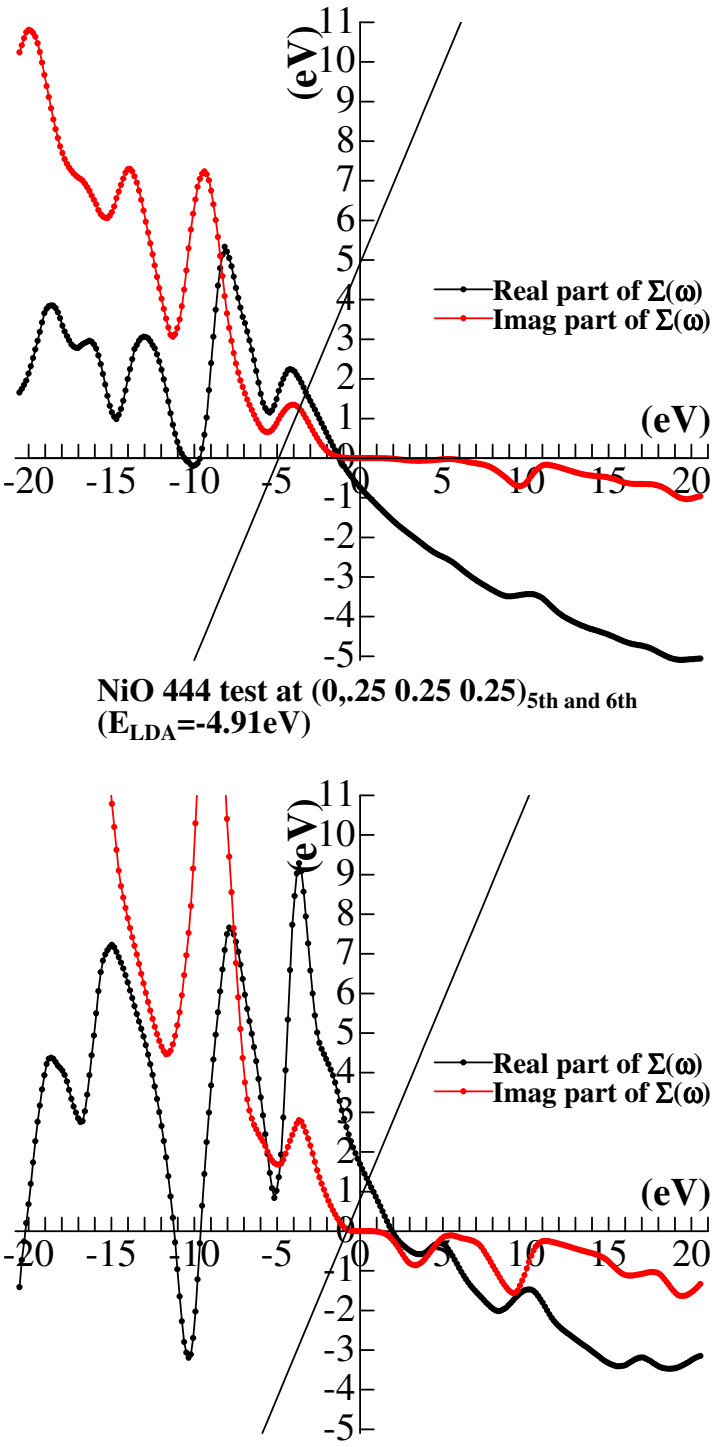


Figure 3: A test sample of mode 4 of hsfp0 to plo $\Sigma(\omega)$. Antiferro NiO case.

13 extest: : To check the dependence of Ex on esmr (smering parameter)

esmr (= $E_{\rm smear}$ in Section 3) is the parameter which is used in **hsfp0** to calculate the self-energy. It is set in GWinput. In **hsfp0**, the poles of the Green function is assumed to have the witdth esmr. (Rectangular smearing or Gaussian). As for the case of insulator, it is not essentially necessary — You can choose it as small enough (too small value might causes a numerical problems — If $E_{\rm smear} = 0$, there is a "hitting (numerical) problem" because given ω and pole of G is exactly the same. So please fix it as 0.01Ry,0.001Ry or so.) However, in the case of metal, the size fo esmr can affects the resutls. Theoretially, if you take enough number of **k** points, you can use very small esmr.

In order to determine esmr for given number of **k** points, I made a test script **extext**. You can run it after you get the Coulomb matrix by **hvcc**. It means that you have to execute **gw_lmf** but you can stop it just after it make the Coulomb matrix:

```
> ~/ecal/fpgw/exec/gw_lmf cu
 OK! lmf2gw: end --- DATA4GW_V2 is written
 OK! rdata4gw
 OK! heftet mode=1 EFERMI generated
 OK! hchknw: write nw to NW          <--- ***
 OK! hbasfp0 ix=3 core mode          <--- ***
 OK! hvccfp0                         <--- ***
 OK! hsfp0: Core-exchange mode       <--- ***
 OK! hbasfp0 ix=0 normal mode
 OK! hvccfp0
```

————————Here you can stop! (you don't need to run some programs shown as *** in order to execute extest.)

Then you can invoke **extest** as

```
>extest foo
. Then you can see the console outputs as
 esmr(Ry) efermi(Ry) Sx1(eV)  Sx2 ...
    0.005 -0.028478  -16.3758  -30.3606  -30.3606  -30.3606  -31.5098  ...
    0.010 -0.027853  -16.3758  -30.3606  -30.3606  -30.3606  -31.5098  ...
    0.015 -0.027228  -16.3758  -30.3606  -30.3606  -30.3606  -31.5098  ...
```

It takes a bit long (the same as time as echo 1—hsfp0), but other lines appeares soon successively. The first column is esmr, the second for the determined Fermi energy, and $\Sigma_{\rm x}$ for eigenfunctions given in <QPNT> section.

You already saw the plots which I made by the script in Section 3.

In principle, we have to estimate the limit of the numbe of **k** points $\to \infty$; then we can estimate esmr so that it is not so small so as not to include the strange beheviors near esmr=0 due to the **k**-points-number cutoff.

You see that we have smoother behevior for larger number of **k** points; $14\times14\times14$ case of Cu in section 3 gives rather smooth for esmr> .25 or so for $E_{\rm fermi}$. This is a case of GaussSmear=off. You may need to repeat it with GaussSmear=on— In Si case, I saw that 1/3 of esmr of GaussSmear=off gives similar effect.

The larger dependence for $\langle\Psi_{\mathbf{k}n}|\Sigma_{\rm x}|\Psi_{\mathbf{k}n}\rangle$ is for states near the $E_{\rm Fermi}$; X_4 X_6, L_6 in the Cu case. The completely flat behevior at esmr $\to 0$ in the case of $6\times6\times6$ comes from the fact that the intervals of states at EF are smaller than these esmr. Apparently $6\times6\times6$ case miss the exchange enegy about 0.2 eV for L_6 though the error will be cancelled by the correlation part. The curves for $6\times6\times6$ diverges from others at even large value esmr=0.25 Ry in X6. However the difference

gets small for esmr< 0.06. So you might be able to use rather small value of esmr even for $6\times6\times6$ case for Cu. As for $10\times10\times10$ case, you can see a strange behevior in X_6 for esmr< 0.025 Ry. So it might be better you to use larger emsr to avoid the behevior. Apparanly it is just for exchange, and we have to do total check runs for convergence of QPU though extest will help to give some informations what size of esmr will be reasonable.

It is easy and rather quick if you want to repeat extest. Please use **extest_repeat**. If you want further plotting points, you edit **extest_repeat** (add values of esmr in foreach loop). In order to remove too many files and directories generated, please do `rm -rf EX*`.

14 gwsc: QP self-consistent *GW*

With a script *gwsc*, you can do QS*GW*calculation as in [3]. The procedure is as follows.

1. Get LDA results. `rst.si`

2. Invoke **gwsc1shot**.
   This does only a first itteration to generate sigm containing $\Sigma - V_{\rm xc}^{\rm LDA}$. At the end of **gwsc1shot**, you will get sigm. Before invoke it, you can set, e.g.,
   "`emax_sigm  5.00 !(Ry) emax cutoff for Sigma (Optional)`"
   in GWinput so as to reduce the computational time. However, this 1st itteration is to check the behevior for $\Sigma - V_{\rm xc}^{\rm LDA}$ as fucntion of $\epsilon_{\mathbf{k}n}$. So it meybe better to take larger emax_sigm as possible (maybe larger than 5 (Ry) or above).

3. Write optional section in ctrl.*.
   At first, you have to add HAM—RDSIG and HAM—SIGP tokens to ctrl.* e.g. as;

   ```
   HAM     RDSIG=12 SIGP:3,0,0,0,2.5,0,.06,0
   ```

   These are explained in lmto doc as `ecal_sc/lm-6.14/doc/gw.txt`. See the note. I give a sketch here.——— RDSIG=12 means to add the self-energy $\Sigma - V_{\rm xc}^{\rm LDA}$ to Hamiltonian if sigm.* exists. In addition, RDSIG=12 means that we just take approximated (extraporated) diagonal-only $\Sigma - V_{\rm xc}^{\rm LDA}$ for high-energy bands. The part `SIGP:modsgp,nmin,emin,nmax,emax,asig,bsig,efit` is how to approximate the diagonal part of high-energy bands. emax,asig,bsig are important quantities. Especially emax is the cutoff above which we only take extrapolated diagonal parts of $\Sigma - V_{\rm xc}^{\rm LDA}$.
   After you add these tokens, invoke **lmf** (See the script gwsc— don't forget rename sigm as sigm.*). Then you can see infomations below in standard(conosle) output [3] as;

```
 hambls: approximate sigma for states n=6 and below; and for energies E(lda)>2
state E(lda)     sig_ii     constraint  use
...
...
 33   1.402066   0.135934               0.135934
 34   1.622702   0.138711               0.138711
 35   1.649643   0.153268               0.153268
 36   1.650243   0.173259               0.173259
 37   1.650244   0.173259               0.173259
 38   1.919147   0.197660               0.197660
 39   1.919162   0.197648               0.197648
 40   1.935087   0.167100               0.167100
 41   1.935099   0.167125               0.167125
 42   1.979897   0.179666               0.179666
 43   2.110782   0.233979   0.368863    0.368863 <-- 43th
 44   2.110789   0.233979   0.368863    0.368863 <-- 44th
 45   2.736957   0.000000   0.418957    0.418957
 46   2.736958   0.000000   0.418957    0.418957
 47   2.737101   0.000000   0.418968    0.418968
 48   2.976955   0.000000   0.438156    0.438156
 49   3.041153   0.000000   0.443292    0.443292
```

[3] set larger VERBOSE to see this info, maybe 50 or so—this info is for each **k**

```
...                          ^^^^^^^^---Above 43th, use extrapolated sig_ii
...                                     (shown at "constraint" column)
...              ^^^^^^^^---Up to 44th, sig_ii are calculated
```

This shows results from a key part for the extrapolation to determine diagonal parts of higher bands. The third column `sig_ii` is for the calculated $\Sigma - V_{\rm xc}^{\rm LDA}$. In this example, only $\Sigma - V_{\rm xc}^{\rm LDA}$ up to 44th state are calculated because we used "`emax_sigm  2.500 !(Ry)`" in GWinput in this case.

On the other hand, "`constraint`" section is from 43th. This means that we assign "higher bands for extrapolation" from 43th. This is controlled by `emax` in `SIGP`. The extrapolated values are given by `asig,bsig`. In this case, these values in "`constraint`" correspond to `asig=0 bsig=.06`.

At the same time you see lines as

```
hambls: sig(low,high) = 0.0000,0.2353  fit : 0.1447 + -0.0238 * E(lda) (1768 points)
```

in the same standard output of **lmf**. Here
`fit : 0.1447 + -0.0238 * E(lda)` gives linear fitting of `sig_ii` as function of `E(lda)`. These are important informations to supply reasonable `asig,bsig`.

If you want to use these values, you set asig=0.1447, bsig=-0.0238 in `SIGP` token. (however, this case is a fit up to 2.500 (Ry). So not so good. Negative bsig is a bit strange—it may be better to choose fitting region... See gw.txt).

At the end, you have to give reasonable parameters for `SIGP` including `asig,bsig`. Note that you have to set `emax` in `SIGP` and `emax_sigm` in GWinput so as to have some overlap, so that `sig_ii` should be determined directly or the extrapolation. We expect that these numbers affects little to final results anyway. With our experiences `emax` in `SIGP` $\sim$ 2.0Ry is good for NiO and so.

Anyway see `ecal_sc/lm-6.14/doc/gw.txt` for details.

4. Set `iSigMode` in GWinput. and invoke **gwsc**. In each itteration step, you will have .{*itt-number*}run files. E.g, QPU.1run, QPU.2run... are generated at each itteration step.

15 Check list for convergence

The compuational resource is rather limited but results could be dependent on cutoff parameters in GWinput, and on the LDA result as inputs. So you have to schedule the convergence check carefully and efficiently, mainly on these points below.

- LDA eigenfunctions and number of unoccupied states.

  A good way to check the convergence of LDA result is the comparison of the energy bands and the density of states generated by FP-LMTO and some other accurate method. Within FP-LMTO, you can check the convergence when you enlarge the LMTO basis sets, Head and Tail; but the LMTO itself has difficulties to persue the convergence for higher unoccupied states.

  Main problem could be how many unoccpied states you consider. Our experience shows that we need to add rather large number of unoccupied states for good convergence even if these accuracy would be not so good [4]

[4] This may be related to a completeness of the basis set; the completeness could be important from the view of 'Coulomb hole' picture.

[In GWinput, you can set the number of unoccupied states which you take into account by `emax_chi0`, `emax_sigm`, `nband_chi0`, and `nband_sigm`. But we now usually unset them except the case of scGW calculation for `emax_sigm`.]

Actually this point is most difficult for convergence check. You have to set up carefully some LDA results as input to check the convergence.

- Number of k points `n1n2n3`.
- Product basis section. At least, `lcutmx`=4 will be necessary for atoms with d electrons.
- Cores. Usually we nees to treat only the shalow cores as **core2**, others as **core1**. But you need check. As for semi-core like *d* bands, it is better to treat them by local orbital.
- `esmr`. You have to be careful if metal. You can test it with **extest**. It might be a probelm to choose too small `esmr`.
- `dw, omg_c` `niw`

  It will be worth to try to check how much the results changed due to them. But usually `dw=0.01, omg_c=0.05` is not so bad. As for `niw=6` seems to be not so bad usually, but it is safer to check the convergence on it (test cases with `niw=10,12,16`).
- `deltaw`.

  $\sim$ 0.01 a.u. will be not so bad. See two Z values shown in SXCU. It is better to try to check how about the depencence on this.
- `CoreOrth` on. Try to test it. If it affects so much. The D function might be too poor due to the poor orthogonality condition between core and valence.

16 EXX+RPA total energy

There is a mode to calculate total energy, origianlly started by Dr.Miyake. I tested it so much, but it is numerially not so satisfactory; see Ref.I.

17 Linear response calculations

We now have these `eps*` scripts in the following.

- `eps_lmfh` epsilon with local field correction.
- `epsPP_lmfh`

  epsilon without local field correction. $1 - \langle e^{i\mathbf{qr}}|v|e^{i\mathbf{qr}}\rangle\langle e^{i\mathbf{qr}}|\left(\chi^0\right)|e^{i\mathbf{qr}}\rangle$
- `epsPP_lmfh_chipm`

  For spin susceptibility. This essentially calculate non-interacting spin susceptibility. Then it is used for the calculation of full spin susceptibiity with `util/calj_*.F` programs (small quick programs). See spin wave paper. See spin susceptibility section Sec.**??**.
- `eps_lmfh_chipm`

  This gives full non-inteacting spin susceptibility. Testing. We have to determine U (stoner I) for the determination of full spin susceptibility. TDLDA? or so?
- `(This is old mode --- removed not) epsPP_lmfh_chipm_q`

  For spin susceptibility. spin susceptibility $\langle e^{iqr}|\chi(q,\omega)|e^{iqr}\rangle$ In this script, You have to assign that isp=1 is majority, isp=2 is minority. This is with long wave approximation.

• `*_lmfh_*` means histogram method. At first, we calculate its imaginary parts with tetrahedron technique. Then we get its real part by Hilbert transformation.
You need to choose `dw,omg_c`. The width of histrgram bins are getting larger when omega gets larger. dw is the size of histogram-bin width at omega=0. At omega=omg_c, its width gets twiced. You have to choose small enough omega for spin wave mode as 0.001 Ry (Or smaller). omg_c is given like 0.05 Ry or so. But sometimes it can be like 1Ry.

• `epsPP` uses a a special product basis set for cases without inversion (actually the problem is just in how to expand exp(iqr) in the mixed basis; the product basis is not from phi and phidot, but from spherical bessel functions).

• `EPSrange, EPSdw` are not used for `*_lmfh_*` scripts.

```
In *_lmfh_* modes( I now use little for *_lmf_* modes), you can use small enough delta.
Use small enough delta (=-1e-8 a.u.) for spin wave modes (also you can use it for
dielectric function and GW).  This is necessary because pole is too smeared
if you use larger delta.
```

18 eps_lmfh, epsPP_lmfh: the dielectric functions

You can invoke the script, e.g. as "**eps_lmfh** si".

Specify **q** point in `<QforEPS>` or so. Mesh for ω is specified by `dw, omg_c`.

The obtained datas are in EPS*.dat and EPS*.nlfc.dat. EPS*.nlfc.dat contains the result without local-field correction EPS*.dat contains the result with local-field correction (this is generated only for `eps_lmfh`. Both of them contains

$\mathbf{q}$(1:3), ω, Re(ϵ) Im(ϵ), Re($1/\epsilon$), In($1/\epsilon$)
in each line.

For the limit $\mathbf{q} \to 0$, be careful! Because $\mathbf{q} \to 0$ gives too large cancelation effects (the denominator and numerator go to zero—it means we need very accurate orthogonalization between occupied and unoccupied states). This is a kind of disadvantage of our method (though there is an advantage— our code can calculate dielectric function even for metal as far as you use large enough number of $\mathbf{k}$ point.)

The calculaten of dielectric functions usually requires so many k point. For example, for si, `n1 n2 n3 = 4 4 4` is too small. It gives too large dielectric constants ~ 19.4 though the converged value should be ~ 13. (we need 10x10x10 or more like 20x20x20 for some reasonable results). For GaAs, we observed that reasonable $\epsilon(\omega)$ requires rather large number of $\mathbf{q}$ points lke 15x15x15 or 20x20x20 for `n1n2n3`. This is too time-consuming to get result (but you can use "very small product basis"(just sp poralization for this purpose; it makes speed up so much). Or, you can calculate "$\epsilon(\omega)$ without LFC". See section for **eps_PP_lmfh**.

————-**!WARNING!**————-

1. This code works OK only for $\mathbf{q}$ is near 0. Be careful for $\mathbf{q} \to 0$ limit. Too small $\mathbf{q}$ can give strange spectrum at high energy (real part is affected by it)

2. `CoreOrth` gives so serious effect for $\epsilon(\omega)$, if you include some cores as "**core2**" in the product basis setting. (This means that you includes transtions from "**core2** to valence" in the calculation of $\epsilon(\omega)$).

Then you have to use "`CoreOrth` on". Without it, you will have rather large imaginary part at rather high energy Such transitions from core to higher valence bands is artifical due to the incomplete orthogonality between core and the higher bands. However, shallower d semi-core might be deformed too much by this option. Try to plot Core_*.chk files, which contains core radial functions. Anyway, it is better to treat shallow core as valence by "local orbital".

18.1 epsPP_lmfh: the dielectric function(No LFC— faster)

You can calculate ϵ without LFC by **epsPP_lmfh**. It is very faster than **eps_lmfh**.

To calculate $\epsilon(\mathbf{q}, \omega)$ without LFC accurately, the best basis set for the expantion of the Coulomb matrix within MT is apparently not the product basis, but the bessel functions corresponding to the plane waves exp($i\mathbf{qr}$). We use such a basis in this mode. However, our experience shows that the changes are little even with the usual product basis (we don't describe this here).

19 How to calculate correct epsilon?

```
There are prolems to calculate correct epsilon.
At first, we talk about epsPP_lmfh, which is No LFC. Main problem are

-----------------
1.Convergence for number of k point(specified by n1n2n3).
  Roughly speaking, 20x20x20 is required for not-so-bad results for Fe and Ni.
  It is better to do 30x30x30 to see convergence check.
  However, in the case of ZB-MnAs (maybe because of simple structure around Ef),
  it requires less q points.
```

figs are for GaAs.
fig001: n1n2n3 convergence for Chi_RegQbz = on case.
fig002: n1n2n3 convergence for Chi_RegQbz = off case.
(Chi_RegQbz in explained in General section in this manual).

As you see, k points convergence looks a little better in Chi_RegQbz=off
(mesh not including gamma). However a little ploblem is that its thereshold around
0.5eV is too high and slowly changing.

fig003: Alouanis'(from Arnaud) vs. ``Chi_RegQbz = on'' vs. ``Chi_RegQbz = off''
As you see, the threshold of the Red line (20x20x20 Chi_RegQbz=on) and Alouani's
are almost the same, but the red line is too oscilating at the low energy part.
On the other hand, ``Chi_RegQbz = off'' in Green broken line is not so satisfactory
at the low energy part.

fig.gas_eps_kconf.pdf shows the convergence behevior of epsilon for

2.$q \to 0$ convergence (this is related to whether Chi_RegQbz=on or off).
If you use very small q like q=0.001 is GaAs, it can cause a problem.
Use q=0.01 or larger (maybe q=0.02 or more is safer).
Very small q can give numerical error for high-energy region.

In fig004, we show the high energy tail part of Im $\epsilon(\omega)$ for GaAs case.
At q=0.01 (this means q= 2*pi/alat * (0 0 0.01)), the imaginary part
is a little too large . Less than 80eV, q=0.02 gives good results when compared with
other high q results, though it still has noise above 80eV.
In fig005, I showed the same results compared with Alouani's (his is up to 40eV).
Both gives rather good agreements. As you see, q=0.06 or above might be necessary
to get reasonable convergence for high energy part abouve 40eV.

We have to be careful for this poorness in high energy part--- it may effect
low-energy Re[$\epsion$] through KK relation. However this can be very small
ehough.
In fig.gas_eps_qconv.jpg, we checked the convergence of eps (\omega=0,q) for q \to 0.
As you see, it gives convergence, however, q=0.01 is a little out of
curve---this should be because of the poorness in the high energy part.
so q=0.02 or q=0.03 is safer, and you can get eps within 1 percenr accuracy.

3. Including Core for dielectric constant is dangerous.
   It can cause very poor results if you include core part in GWinput.
   You need to include core just as valence (with local orbital).

   In fig008, we showed core effects. It starts from \approx 16eV
   (this is core to conduction transition).
   fig007 showd the check about the q point dependence---even with large q,
   it would not change.
   These shows that the core excitation can have larger energy range.

```
   This is in contrast to the valence case
   (then the most of excitaion is limited to less than 10eV).
   We have to be careful for such high-energy exciation... The LMTO basis might
   be not so good for high energy.

4. basis set.
   Use QpGcut_psi \approx 3.0 a.u. or so (as same as GW calculation).
   In the case of epsPP* mode,
   QpGcut_cou can be very small--- In our codes now,
   ngc>=1 should be for all q vector shown in lqg4gw02 (output of echo 2|qg4gw).
   [In principle, it should be only for the q vector for which we calculate epsilon.
    But there is a technical poorness in our code---
    (maybe) a problem here; the plane-wave part of the eigenfunction generated
    in lmfgw is not correctly passed to lmf2gw when ngc=0].


-- eps_lmfh: including LFC ----------------------------------
To include eps with LFC, do eps_lmfh.
But lcutmx=2 seems to be good enough to get 0.5 percent error (maybe better than this).
Test it 10x10x10 or so. (I need to repeat if necessary).
Further you can use smaller QpGcut_cou like 2.2 or so,
with rather smaller product basis (up to p timed d, not including f).

Note: epsPP_lmfh is designed to use good basis to calculate eps
without LFC. This is usually in agreement with what you obtained by eps_lmfh;
however it can give slight difference when you use small product basis.


---Summary --------------------
So in conclusion, I think a best way to do is

1. set q=0.02 [q=2pi/alat(0 0 0.02)] or so for GaAs case.
   If you want to check, do q=0.03 and q=0.06 also.

   ``Chi_RegQbz = off'' is better for matrials like GaAs with direct gap.

2. You can use small QpGcut_cou but all ngc should be one or more.

3. As for the Product basis setting in epsPP* scripts, only
   lcutmx and tolerance (this can be like 0.001 or so) are relevant.
   E.g. set lcutmx=4 or so.

4. Do nk=20 18 16 and take interpolarion to determine eps(omega=0, q=0).

5. To get eps with LFC, set QpGcut_cut as xxx, and set lcutmx=2 where
   (occupied sp) \timex (unoccupied spd) are included.
   But correct EPS*.nolfc.d is rather from epsPP_lmfh script.
```

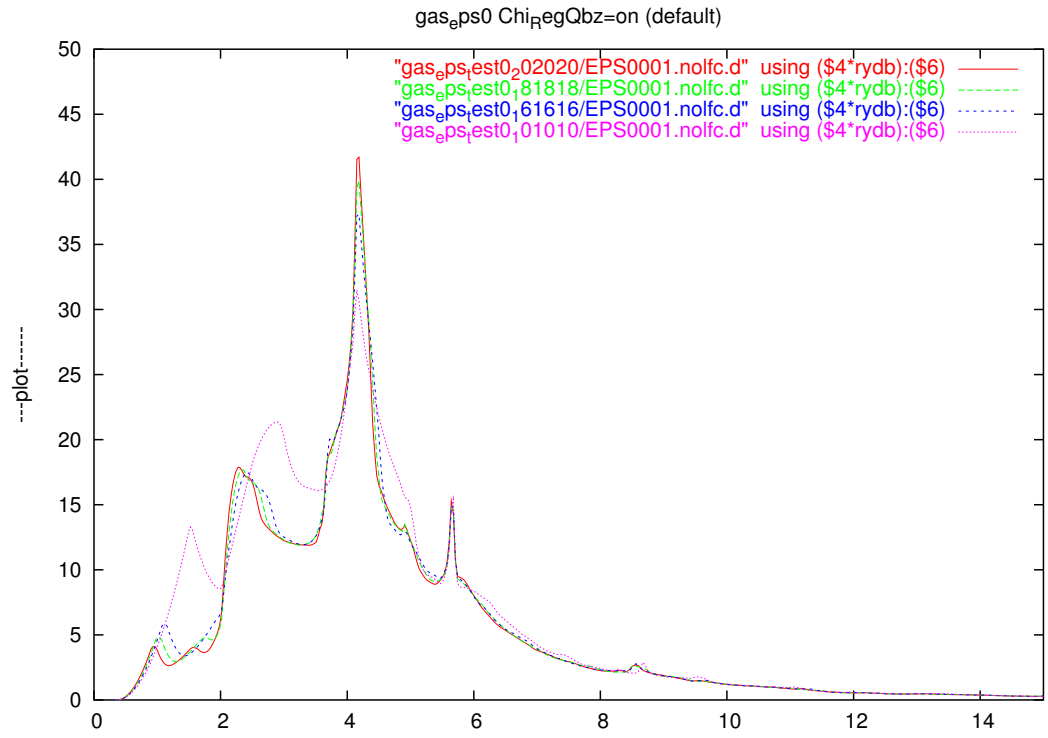


fig001

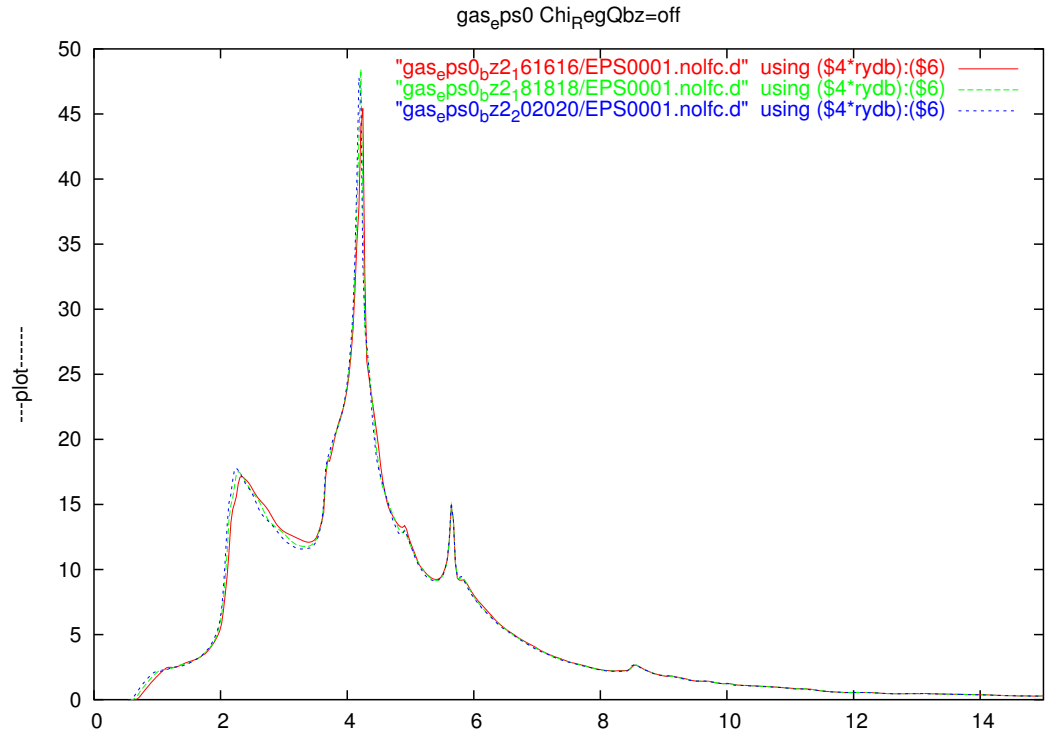


fig002

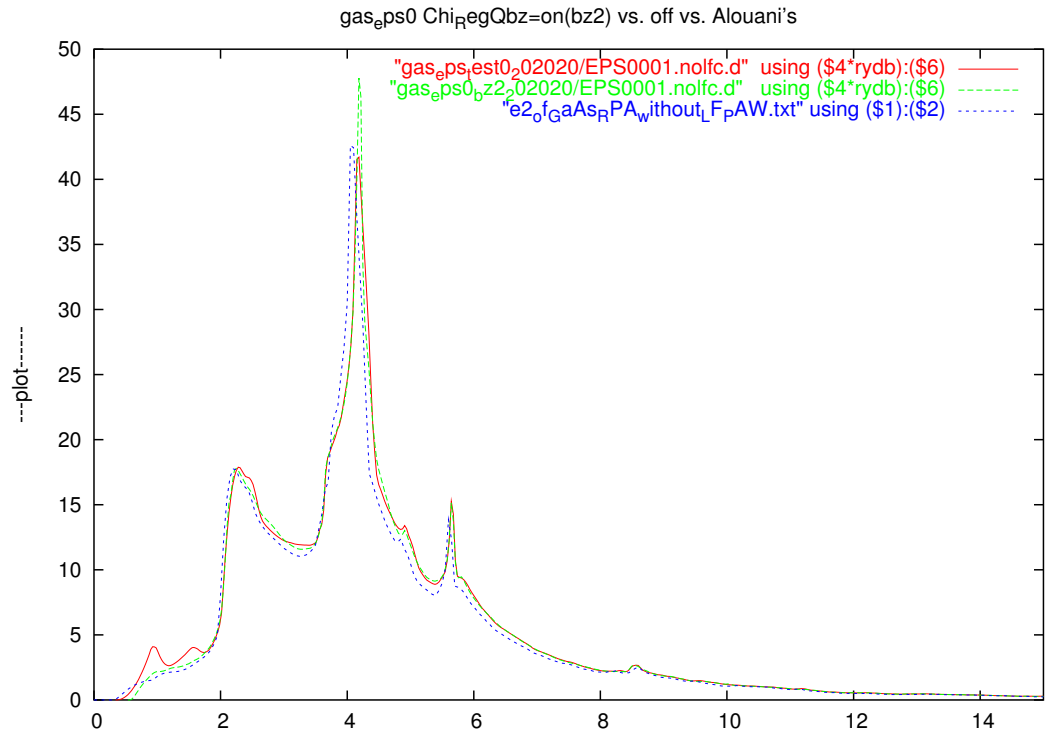


fig003

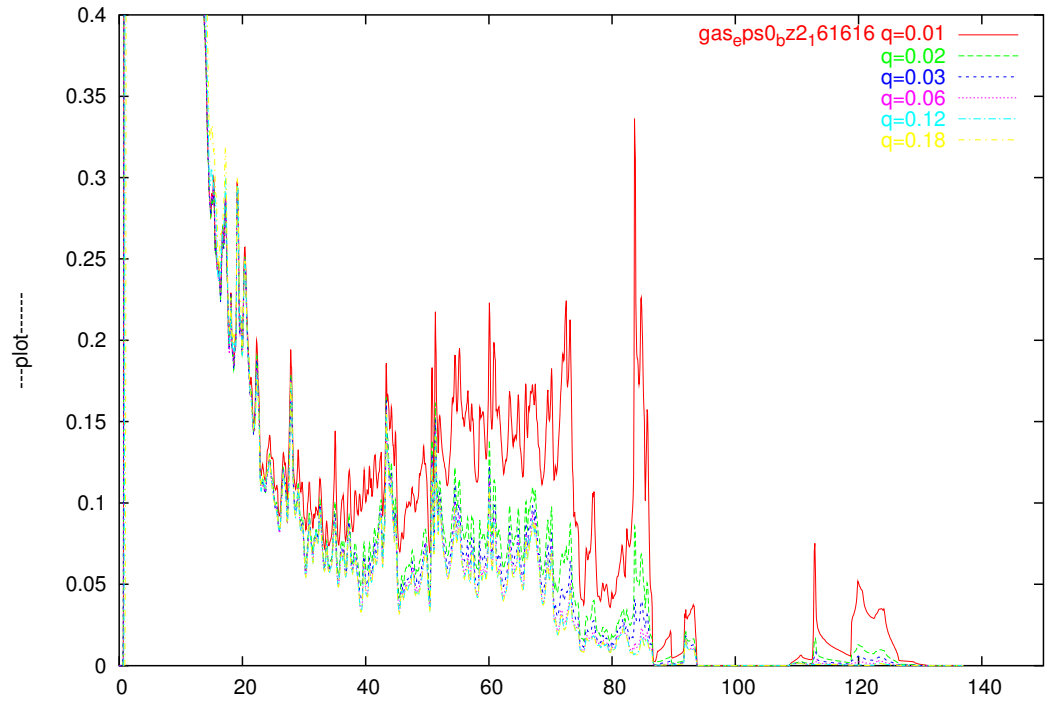


fig004

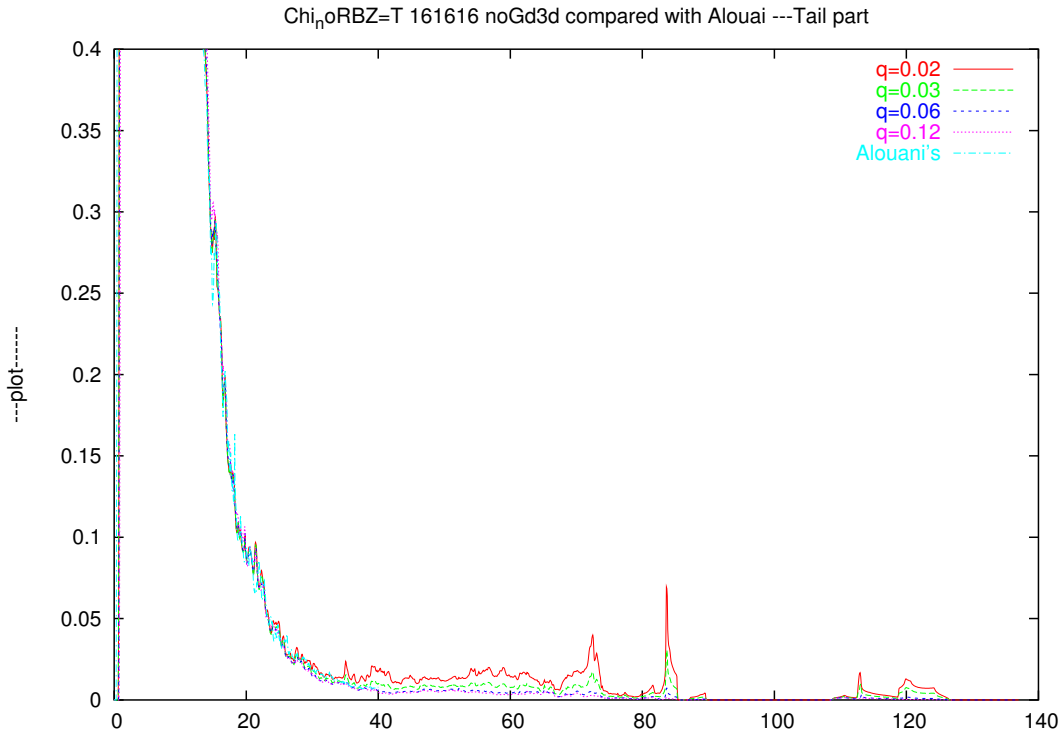


fig005

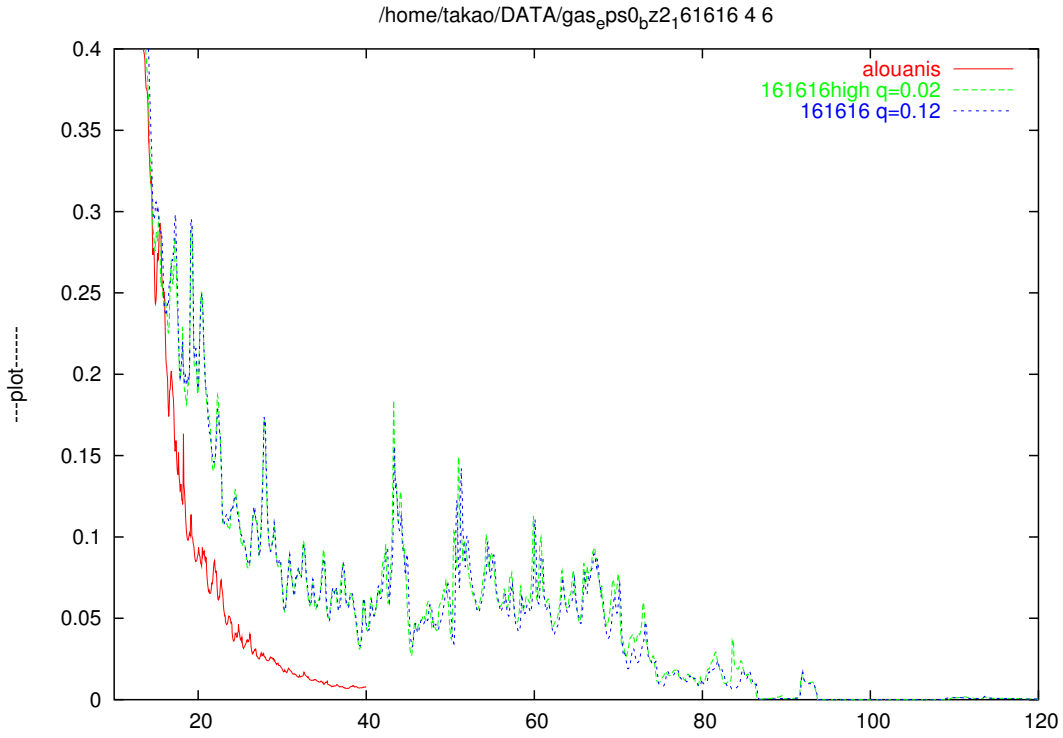


fig007

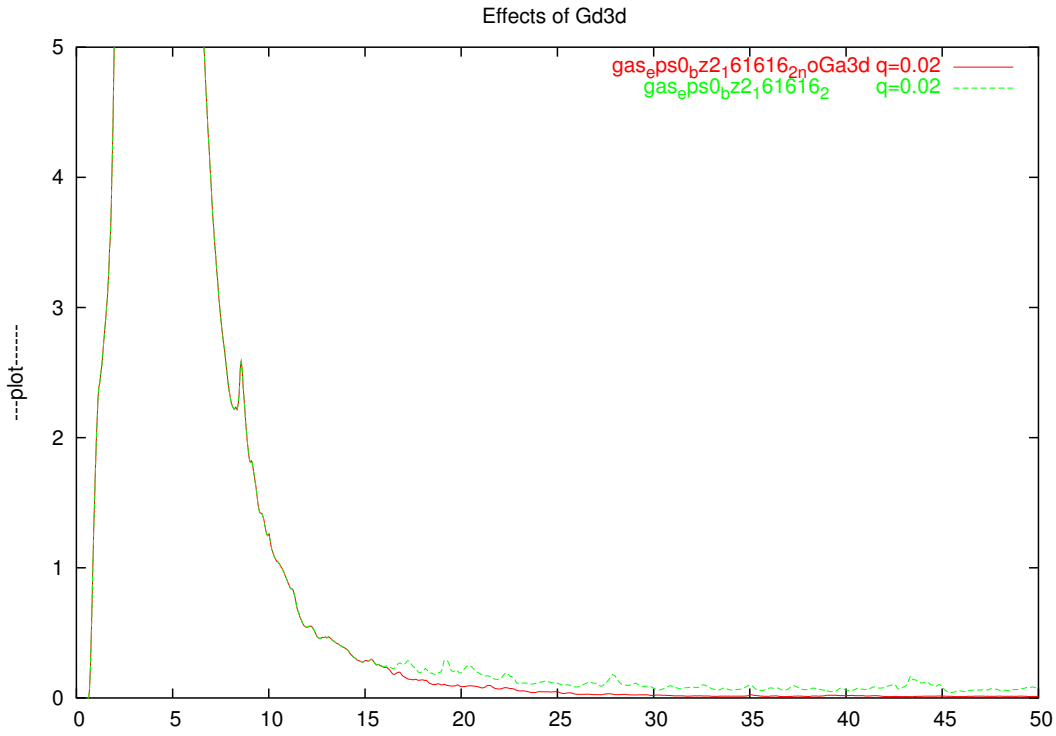

Effects of Gd3d
gas_epsO_bz2_161616_2_noGa3d q=0.02
gas_epsO_bz2_161616_2 q=0.02
---plot-----
0
1
2
3
4
5
0
5
10
15
20
25
30
35
40
45
50

fig008

20 χ^{+-} calculation

[I changed sign of χ! (July2007); This definition may be different from other text book. Be careful] The non-interacting transverse spin susceptibility $\chi^{0+-}(\mathbf{r},\mathbf{r}',t-t')$ is given as

$$\chi^{0+-}(\mathbf{r},\mathbf{r}',t-t') = -i\langle T(S_+(\mathbf{r},t)S_-(\mathbf{r}',t')\rangle = -iG_\uparrow(\mathbf{r},\mathbf{r}',t-t')G_\downarrow(\mathbf{r}',\mathbf{r},t'-t) \tag{2}$$

for non-interacting system (Lindhard-like poralization function). In ω space, this reduced to

$$\begin{aligned}&\chi^{0+-}(\mathbf{r},\mathbf{r}',\omega)\\ &= \sum_{n\downarrow}^{\rm occ}\sum_{n'\uparrow}^{\rm unocc}\frac{\Psi^*_{n\downarrow}(\mathbf{r})\Psi_{n'\uparrow}(\mathbf{r})\Psi^*_{n'\uparrow}(\mathbf{r}')\Psi_{n\downarrow}(\mathbf{r}')}{\omega-(\epsilon_{n'\uparrow}-\epsilon_{n\downarrow})+i\delta} + \sum_{n\downarrow}^{\rm unocc}\sum_{n'\uparrow}^{\rm occ}\frac{\Psi^*_{n\downarrow}(\mathbf{r})\Psi_{n'\uparrow}(\mathbf{r})\Psi^*_{n'\uparrow}(\mathbf{r}')\Psi_{n\downarrow}(\mathbf{r}')}{-\omega-(\epsilon_{n\downarrow}-\epsilon_{n'\uparrow})+i\delta},\end{aligned} \tag{3}$$

Here $\downarrow$ is for majority (`isp=1`) and $\uparrow$ is for minority (`isp=2`), $\begin{pmatrix}\text{isp}=1\\ \text{isp}=2\end{pmatrix} = \begin{pmatrix}\downarrow\\ \uparrow\end{pmatrix}$.

$S^+(\mathbf{r}) = \Psi^*_{\text{isp}=1}(\mathbf{r})\Psi_{\text{isp}=2}(\mathbf{r}) = \Psi^*_\downarrow(\mathbf{r})\Psi_\uparrow(\mathbf{r})$,
$S^-(\mathbf{r}') = \Psi^*_{\text{isp}=2}(\mathbf{r}')\Psi_{\text{isp}=1}(\mathbf{r}') = \Psi^*_\uparrow(\mathbf{r}')\Psi_\downarrow(\mathbf{r}')$,
Notes:

- This definition of χ^{0+-} results in $\chi^{0+-}\to\frac{\rm m}{\omega-\Delta_{\rm ex}}$ at $\mathbf{q}=0$ in the case of shifted-band model as $\epsilon^\uparrow_{\mathbf{k}} = \epsilon^\downarrow_{\mathbf{k}} + \Delta_{\rm ex}$. Here $\Delta_{\rm ex}$ means exchange splitting, and $m = N_\downarrow - N_\uparrow$. For paramagnetic case, this definition gives $\chi_0 = 2\chi^{0+-}$, where χ_0 is usual Lindhard polarization function for density response.
- Recall $e^{-i(\epsilon-\epsilon')t}\theta(t) = e^{-i\epsilon t}\theta(t)\times e^{+i\epsilon' t}\theta(t)$ or equivalently $\frac{1}{\omega-(\epsilon-\epsilon')+i\delta} = \int d\omega' \frac{1}{\omega-\omega'-\epsilon+i\delta}\frac{1}{-\omega'-\epsilon'-i\delta}$.
- If no occupation for minority channel, only the 1st term in Eq.(3) remains. In contract to charge density, χ^{0+-} is not symmetric for $\omega\leftrightarrow-\omega$.

By the way, the physically meaningful quantity is the retarted verion of χ^{0+-}, named as $\chi^{0+-}_{\rm Ret}$, which is given by changing the sign of $+i\delta$ in Eq.(3) so as to make it propotional to $\theta(t)$. Let us assume colinear case (z-axis), and consider adding external transversal magnetic fiels $B_x(\mathbf{r},t)$ and $B_y(\mathbf{r},t)$ for our system. For non-interacting system, this $\chi^{0+-}_{\rm Ret}$ specify the linear repsonse to such B_x and B_y. Instead of them, it is convenient to use the complex field, $b^- = b_x - ib_y$, (I introduce $\mathbf{b}$ in unit of $g\mu_{\rm B}/2$, so that $(g\mu_{\rm B}/2)B_x = b_x$: electron's spin magnetic momenets is $-2g\mu_{\rm B}\mathbf{s}$.) Then the additional Hamiltonian due to this magnetic field is written as

$$\begin{aligned}H_{\rm ext\_mag} &= \int d^3r B(\mathbf{r})\cdot g\mu_{\rm B}\mathbf{s}(\mathbf{r}) = 2\int d^3r \mathbf{b}(\mathbf{r})\cdot\mathbf{s}(\mathbf{r})\\ &= \int d^3r\left[b^+(\mathbf{r})s^-(\mathbf{r}) + b^-(\mathbf{r})s^+(\mathbf{r}) + 2b_z(\mathbf{r})s_z(\mathbf{r})\right],\end{aligned} \tag{4}$$

where $s^-(\mathbf{r}) = s_x(\mathbf{r}) - is_y(\mathbf{r})$ and so on. $s_x(\mathbf{r}) = \sum_{\alpha,\beta}\langle\hat\psi^\dagger_\alpha(\mathbf{r})\frac{1}{2}\sigma^x_{\alpha\beta}\hat\psi_\beta(\mathbf{r})\rangle$ and so on, where $\sigma^x_{\alpha\beta}$ is the Pauli matrix. The induced spin moment Δs^- for non-interacting system is given as

$$\Delta s^-(\mathbf{r},t) = \int dt'd^3r'\chi^{0+-}_{\rm Ret}(\mathbf{r},\mathbf{r}',t-t')b^-(\mathbf{r}',t'). \tag{5}$$

[because of $\theta(t)$ in $\chi^{0+-}_{\rm Ret}$, $t-t'>0$.].

In the case of interacting system, we need to construct χ^{+-}. We define $U(\mathbf{r},\mathbf{r}',\omega)$ as

$$(\chi^{+-})^{-1} = \left(\chi^{0+-}\right)^{-1} - U. \tag{6}$$

This is taken as the definition of $U(\mathbf{r},\mathbf{r}',\omega)$. How to define U is the problem — See my **spin wave paper** in Arxiv. We utilize one degree of freedome per atom (so χ^{+-} is the matrix whose

dimension is the number of magnetic atoms), and sum rule. (Some one may call this approximation as "rigid moment approximation". But it can be misleagind. Be careful about what I did.
—-MEMO————————-
`eps_lmfh_chipm`: Our fpgw code can calculate χ^{0+-} in this form of expansion

$$\chi^{0+-}(\mathbf{r},\mathbf{r}',\omega)=\frac{1}{N}\sum_q \chi^{+-}_{\mathbf{q}}(\mathbf{r},\mathbf{r}',\omega)=\frac{1}{N}\sum_q\sum_I\sum_J M_I^{\mathbf{q}}(\mathbf{r})\chi^{0+-}_{\mathbf{q}IJ}(\omega)(M_J^{\mathbf{q}}(\mathbf{r}'))^*. \tag{7}$$

Here $\{M_I^{\mathbf{q}}(\mathbf{r})\}$ is the complete set with the periodicity specified by $\mathbf{q}$ (Mixed basis). In other words, $\{M_I^{\mathbf{q}}(\mathbf{r})/e^{i\mathbf{qr}}\}$ is the complete set to expand periodic function. However, I have not used this now...;problem is determination of U. How to do it? (sum rule is not enough. we need static response?).

21 Sum rule(moment)

The equation of motion of spin is written as

$$i\dot{\hat{\mathbf{S}}}=[\hat{\mathbf{S}},\hat{H}] \tag{8}$$

$$\begin{aligned}\chi^{+-}(\mathbf{r},\mathbf{r}',t-t') &= -i\langle T\left(S_+(\mathbf{r},t)S_-(\mathbf{r}',t')\right)\rangle \\ &= -i\langle S_+(\mathbf{r},t)S_-(\mathbf{r}',t')\rangle\theta(t-t')+i\langle S_-(\mathbf{r}',t')S_+(\mathbf{r},t)\rangle\theta(t'-t)\end{aligned} \tag{9}$$

Thus

$$\begin{aligned}&\frac{\partial}{\partial t}\chi^{+-}(\mathbf{r},\mathbf{r}',t-t')=-i[S_+(\mathbf{r},t),S_-(\mathbf{r}',t)]\delta(t-t') \\ &-\langle[S_+(\mathbf{r},t),H]S_-(\mathbf{r}',t')\rangle\theta(t-t')+\langle S_-(\mathbf{r}',t')[S_+(\mathbf{r},t),H]\rangle\theta(t'-t),\end{aligned} \tag{10}$$

where $[S_+(\mathbf{r},t),S_-(\mathbf{r}',t)]=2S_z(\mathbf{r},t)\delta(\mathbf{r}-\mathbf{r}')$. As $\int d^3r[S_+(\mathbf{r},t),H]=0$, we have

$$\int d^3r\frac{\partial}{\partial t}\chi^{+-}(\mathbf{r},\mathbf{r}',t-t')=-i\langle[S_+(\mathbf{r},t),S_-(\mathbf{r}',t)]\rangle\delta(t-t')=-2i\langle S_z(\mathbf{r},t)\rangle\delta(t-t') \tag{11}$$

This reads

$$\int d^3r\omega\chi^{+-}(\mathbf{r},\mathbf{r}',\omega)=2\langle S_z(\mathbf{r}',t)\rangle=M_z(\mathbf{r}') \tag{12}$$

• At $\omega\to\infty$, this condition get stronger as

$$\chi^{+-}(\mathbf{r}',\mathbf{r},\omega)\to\frac{M(\mathbf{r})}{\omega}\delta(\mathbf{r}-\mathbf{r}')+O(1/\omega^2). \tag{13}$$

See **spin wave paper**.

22 Rigid moment approximation

This means that "the magnetic moments are very rigid that they changes without changing its form". In other words, $\Delta s^-(\mathbf{r})$ induced by any $b_-(\mathbf{r})$ are propotional to its original moment $s_z(\mathbf{r})$. Rigid rotation in spin space can be expressed by e.g., $U^x(\theta)=\exp\left(\frac{i\sigma^x\theta}{2}\right)$ in the case of x-axis rotation. Then you can easily verify $(U^x(\theta))^\dagger\sigma^zU^x(\theta)\propto\sigma^y$ This means $\Delta s^-(\mathbf{r})\propto s_z(\mathbf{r})$. Note that we did rotation only in spin space. We neglect the mappling of $\mathbf{r}$ when we rotate spin—this will cause little problem when the moment is rather spherical. Be careful; our approximation (one-degree of freedom per magnetic atom) may be a little different from the "rigid moment approximation". I did not want to mix it up, thus I avoided this terminology in my [**spin wave paper**].

23 static $J(q)$ calculation—- Heisenberg Model

(See kotani's SW paper). The total energy of our spin system is assumed to be

$$\mathcal{H} = -\sum_{Rn}\sum_{R'n'} J_{RnR'n'}\mathbf{S}_{Rn}\cdot\mathbf{S}_{R'n'} + g\mu_B\sum_{Rn}\mathbf{S}_{Rn}\cdot\mathbf{B}_{Rn} \tag{14}$$

. Here we take all site indexes Rn and $R'n'$ ($J_{RnRn}=0$. $J_{RnR'n'}=J_{R'n'Rn}$. It we restrict sum as $Rn > R'n'$, factor 2 appears.). $\mathbf{S}_{Rn}$ is the spin at Rn (R is for primitive cell, n specify site in a cell). The equation of motion $-i\hbar\dot{\mathbf{S}}_{Rn} = [\mathcal{H}, \mathbf{S}_{Rn}]$, is reduced to be

$$\hbar\dot{\mathbf{S}}_{Rn} = \mathbf{S}_{Rn}\times\left(2\sum_{R'n'}J_{RnR'n'}\mathbf{S}_{R'n'} - g\mu_B\mathbf{B}_{Rn}\right) \tag{15}$$

We introduce $g\mu_B\mathbf{B} = 2\mathbf{b}$, and $\mathbf{S}_{Rn} = \mathbf{S}^0_{Rn} + \Delta\mathbf{S}_{Rn}$. Then Eq.(15) reduce to

$$\begin{aligned}\hbar\dot{\Delta\mathbf{S}}_{Rn} &= \mathbf{S}^0_{Rn}\times\left(2\sum_{R'n'}J_{RnR'n'}\Delta\mathbf{S}_{R'n'}\right) + \Delta\mathbf{S}_{Rn}\times\left(2\sum_{R'n'}J_{RnR'n'}\mathbf{S}_{R'n'}\right) - 2\mathbf{S}^0_{Rn}\times\mathbf{b}_{Rn}\\ &= \sum_{R'n'}\left(2\mathbf{S}^0_{Rn}J_{RnR'n'}\right)\times\Delta\mathbf{S}_{R'n'} - \left(2\sum_{R'n'}J_{RnR'n'}\mathbf{S}^0_{R'n'}\right)\times\Delta\mathbf{S}_{Rn} - 2\mathbf{S}^0_{Rn}\times\mathbf{b}_{Rn}\end{aligned} \tag{16}$$

Introduce the fourier transformation as $\Delta\mathbf{S}_{Rn} = \frac{1}{N}\sum_{\mathbf{k}}\Delta\mathbf{S}_n(\mathbf{k})e^{i\mathbf{kR}}$. Then Eq.(**??**) reduce to

$$\hbar\dot{\Delta\mathbf{S}}_n(\mathbf{k}) = \sum_{n'}\left(2\mathbf{S}^0_n J_{nn'}(\mathbf{k}) - \left(2\sum_{n''}J_{nn''}(0)\mathbf{S}^0_{n''}\right)\delta_{nn'}\right)\times\Delta\mathbf{S}_{n'}(\mathbf{k}) - 2\mathbf{S}^0_n\times\mathbf{b}_n(\mathbf{k}). \tag{17}$$

Assume $\Delta\mathbf{S}_n(\mathbf{k}) \propto e^{-i\frac{\omega t}{\hbar}}$, we have

$$\sum_{n'}\left(\frac{i\omega\delta_{nn'}}{2} + \mathbf{S}^0_n J_{nn'}(\mathbf{k}) - \left(\sum_{n''}J_{nn''}(0)\mathbf{S}^0_{n''}\right)\delta_{nn'}\right)\times\Delta\mathbf{S}_{n'}(\mathbf{k}) = \mathbf{S}^0_n\times\mathbf{b}_n(\mathbf{k}). \tag{18}$$

Let us consider colinear ground state, then $\mathbf{S}^0_n = S_n\mathbf{e}_z$ (S_n is the size of spin with sign). You have

$$\sum_{n'}\left(\frac{i\omega\delta_{nn'}}{2S_n}\right)\Delta\mathbf{S}_{n'}(\mathbf{k}) + \sum_{n'}\left(J_{nn'}(\mathbf{k}) - \left(\sum_{n''}\frac{1}{S_n}J_{nn''}(0)S_{n''}\right)\delta_{nn'}\right)\mathbf{e}_z\times\Delta\mathbf{S}_{n'}(\mathbf{k}) = \mathbf{e}_z\times\mathbf{b}_n(\mathbf{k}). \tag{19}$$

As $\mathbf{S} = S^+\frac{\mathbf{e}_x-i\mathbf{e}_y}{2} + S^-\frac{\mathbf{e}_x+i\mathbf{e}_y}{2} + S^z\mathbf{e}_z$, and $\mathbf{e}_z\times(\mathbf{e}_x\pm i\mathbf{e}_y) = \mp i(\mathbf{e}_x\pm i\mathbf{e}_y)$ we have,

$$\sum_{n'}\left(\frac{\omega\delta_{nn'}}{2S_n} - \bar{J}_{nn'}(\mathbf{k})\right)S^+_{n'}(\mathbf{k}) = b^+_n(\mathbf{k}). \tag{20}$$

$$\sum_{n'}\left(\frac{\omega\delta_{nn'}}{2S_n} + \bar{J}_{nn'}(\mathbf{k})\right)S^-_{n'}(\mathbf{k}) = b^-_n(\mathbf{k}), \tag{21}$$

where

$$\bar{J}_{nn'}(\mathbf{k}) = J_{nn'}(\mathbf{k}) - \left(\sum_{n''}\frac{1}{S_n}J_{nn''}(0)S_{n''}\right)\delta_{nn'} \tag{22}$$

This $\bar{J}_{nn'}(\mathbf{k})$ and also S_n are stored in `Jmat` file (or `JMAT` line when you run a script `ecal/util/calj\_summary_mat` which calls (`calj_nlfc_mat`). Only the differenc between $\bar{J}_{nn'}(\mathbf{k})$ and $J_{nn'}(\mathbf{k})$ are diagonal parts. These are determined so that $\int d^3k J_{nn}(\mathbf{k}) = 0$.

JJMAT contains another definition of J, which is to reproduce SW spectrum (but it does not work well in cases because the SW peaks are not well identified at high $\mathbf{k}$.)

See my spin wave paper.

24 $J(q)$ and Tc

—- this section is my memo. Not need to read here —-

(This section is not consistent with previous page. Only a case, with an atom in the cell). The total energy of our spin system is assumed to be

$$E_{\rm spin} = -\sum_i \sum_j J_{ij} \mathbf{e}_i \cdot \mathbf{e}_j \tag{23}$$

. Here we take all site indexes i and j ($J_{ii} = 0$. $J_{ij} = J_{ji}$. It we restrict sum as $i > j$, factor 2 appears.). $\mathbf{e}_i$ is the unit vector to specify the spin direction.

If we identify $E_{\rm spin}$ as the Heisenberg hamiltonian with a fixed spin moment $m = N^{\downarrow} - N^{\uparrow}$, the spin wave dispersion is given as

$$\omega_{\mathbf{q}} = \frac{4}{m} \left[J(0) - J(\mathbf{q}) \right]. \tag{24}$$

This $\omega_{\mathbf{q}}$ is different from the true pole of $\langle m | \left(\chi^{+-}(\mathbf{q}, \omega) \right)^{-1} | m \rangle$ except $\mathbf{q} \to 0$.

The critical temperature $T_{\rm c}$ is given as $T_{\rm c} = \frac{2}{3} J(\mathbf{Q}) Q_{\rm factor}$. This $Q_{\rm factor}$ can be $(S+1)/S$, but it seems to be taken as unity usually... $J(Q=0)$ is for ferromagnetic case.

In order to calculate $J(0)$, integrate the left hand side of Eq.(**??**) in the BZ and use $\int \frac{\Omega d^3 q}{2\pi} J(\mathbf{q}) = 0$. It gives

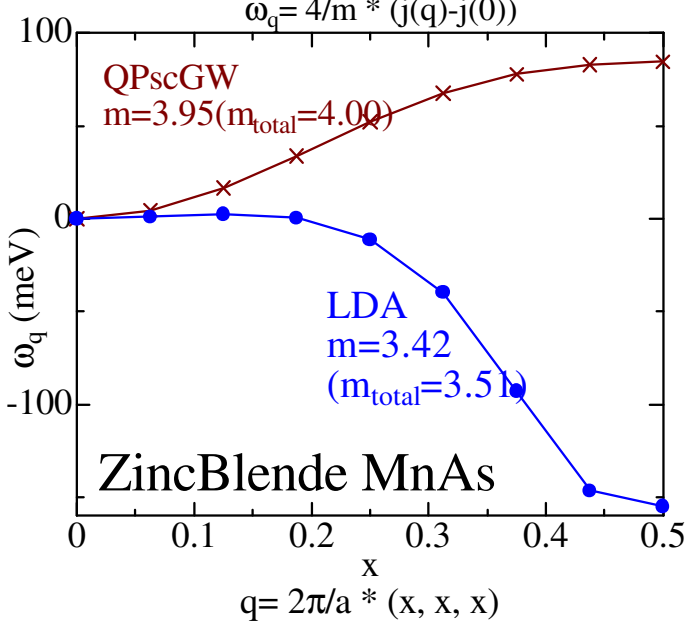
$$T_{\rm c} = \frac{2}{3} J(0) = \frac{2}{3} \int \frac{\Omega d^3 q}{2\pi} \langle m | \left(\chi^{+-}(\mathbf{q}) \right)^{-1} | m \rangle = \frac{2}{3} \frac{m}{4} \int \frac{\Omega d^3 q}{2\pi} \omega_{\mathbf{q}} \tag{25}$$

This equation contains two probrems.

(1) Mapping to a Heisenberg model.
This may cause a problem in the case of transition metals, and so. The spin waves have strong dumping. Further, we don't include the temperature-dependence of the model itself.

(2) Mean field apprximation to solve the Heisenberg model.
In other words, $Q_{\rm factor}$ should be a functional of $J(\mathbf{q})$ for all $\mathbf{q}$. We can devide the problem into classical part and quantum part. A contraversial point is that the integral in Eq.(25) is rather dominanted by the contribution around the BZ boundaries, though we can expect that $T_{\rm c}$ can be rather strongly controlled by low energy $\omega_{\mathbf{q}}$.

— (this is Mark says)—
[MF Quantum Tc] — too high. (S+1)/S * MFC
[Full Quantum Tc] — Exact solution of the Heisenberg Model.
[MF Classical Tc]
[Full Classical Tc] (probably $\sim$ 80 % of MFC)



16x16x16
dw omegac=0.005 0.04 (a.u.)
delta =-1e-8
emax chi0 2.0 Ry

Figure 4: [We have to recalculate this!!!] $\omega_{\mathbf{q}}$ for ZincBlend MnAs by Eq.24. LDA case vs. QSGW case. QSGW makes ZincBlend MnAs as half-metallic. m denotes on-site moment in μ_B. The mean-filed T_c in QSGW is about 600 K.

25 eqs. for Fourer transformation in fpgw program

(I think this section is still meaanigful for my code).

Any site-dependent functions $A(\mathbf{R})$ are written as

$$\bar{A}(\mathbf{k}) = \sum_R A(\mathbf{R}) e^{-i\mathbf{k}\mathbf{R}} \tag{26}$$

$$A(\mathbf{R}) = \frac{1}{N}\sum_{\mathbf{k}} \bar{A}(\mathbf{k}) e^{i\mathbf{k}\mathbf{R}} \to \int \frac{\Omega d^3k}{(2\pi)^3} \bar{A}(\mathbf{k}) e^{i\mathbf{k}\mathbf{R}} \qquad (N\to\infty.) \tag{27}$$

(For any $\alpha(\mathbf{k})$ and $\bar{A}(\mathbf{k})$, $\frac{1}{N}\sum_{\mathbf{k}} \alpha(\mathbf{k})\bar{A}(\mathbf{k}) \to \int \frac{\Omega d^3k}{(2\pi)^3}\alpha(\mathbf{k})\bar{A}(\mathbf{k}) \qquad (N\to\infty).)$

In the case of $A(\mathbf{R}) = 1$ (constant function), $\bar{A}(\mathbf{k}\neq 0) = 0$ and $\bar{A}(\mathbf{k}=0) = N$. At $N\to\infty$, $\bar{A}(\mathbf{k}) = \frac{(2\pi)^3}{\Omega}\delta(\mathbf{k})$. Here $\mathbf{k}$ takes discrete values as $\mathbf{k}_{n_1n_2n_3} = \frac{2\pi}{a}\left(\frac{n_1}{N}\mathbf{b}_1 + \frac{n_2}{N}\mathbf{b}_2 + \frac{n_3}{N}\mathbf{b}_3\right)$, where $N = n_1n_2n_3$. a is the given scale of the system (given in `alat`). $\mathbf{b}_i$ are reciprocal lattice vector (given in `qlat` or `qbas`). $\mathbf{a}_i\cdot\mathbf{b}_j = \delta_{ij}$. ($\mathbf{a}_i$ is given in `plat`). Note $\int \frac{\Omega d^3k}{(2\pi)^3} = 1$.

At first, we assume $\Psi_{\mathbf{k}n}$ is normalized in the macroscopic volume V. In **fpgw** program, we use $\bar{\Psi}_{\mathbf{k}n}$ as $\bar{\Psi}_{\mathbf{k}n} = \sqrt{\frac{V}{\Omega}}\Psi_{\mathbf{k}n}$, where $\Omega = V/N$ denote the volume of primiteive cell. Thus the normalization is

$$\int d^3r |\bar{\Psi}_{\mathbf{k}n}(\mathbf{r})|^2 = 1 \tag{28}$$

1. Then

$$\sum_{\mathbf{k}} \Psi^*_{\mathbf{k}}(\mathbf{r})\Psi_{\mathbf{k}}(\mathbf{r}') = \int \frac{Vd^3k}{(2\pi)^3}\Psi^*_{\mathbf{k}}(\mathbf{r})\Psi_{\mathbf{k}}(\mathbf{r}') = \int \frac{\Omega d^3k}{(2\pi)^3}\bar{\Psi}^*_{\mathbf{k}}(\mathbf{r})\bar{\Psi}_{\mathbf{k}}(\mathbf{r}'). \tag{29}$$

2. The Fourier transformation for Bravias lattice.

$$\delta_{\mathbf{T}\mathbf{T}'} = \int \frac{\Omega d^3k}{(2\pi)^3} e^{i\mathbf{k}(\mathbf{T}-\mathbf{T}')}, \qquad \sum_{\mathbf{T}} e^{i\mathbf{k}(\mathbf{T}-\mathbf{T}')} = \frac{(2\pi)^3}{\Omega}\delta(\mathbf{k}) \tag{30}$$

   $\mathbf{T}$ denotes Bravais lattice as $\mathbf{T}_{i_1i_2i_3} = a\left(i_1\mathbf{a}_1 + i_2\mathbf{a}_2 + i_3\mathbf{a}_3\right)$.

3. At first, we express $\chi(\mathbf{r},\mathbf{r}')$ as the superposions of $\chi_{\mathbf{q}}(\mathbf{r},\mathbf{r}')$ components;

$$\chi(\mathbf{r},\mathbf{r}') = \frac{1}{N}\sum_{\mathbf{q}} \chi_{\mathbf{q}}(\mathbf{r},\mathbf{r}') = \int \frac{\Omega d^3q}{(2\pi)^3}\chi_{\mathbf{q}}(\mathbf{r},\mathbf{r}'). \tag{31}$$

   Here $\chi_{\mathbf{q}}$ is written as

$$\chi_{\mathbf{q}}(\mathbf{r},\mathbf{r}') = \sum_{\mathbf{T}} \chi(\mathbf{r}+\mathbf{T},\mathbf{r}')e^{-i\mathbf{q}\mathbf{T}}, \tag{32}$$

   which satisfy

$$\chi_{\mathbf{q}}(\mathbf{r}+\mathbf{T},\mathbf{r}') = \chi_{\mathbf{q}}(\mathbf{r},\mathbf{r}')e^{i\mathbf{q}\mathbf{T}}, \quad \chi_{\mathbf{q}}(\mathbf{r},\mathbf{r}'+\mathbf{T}) = \chi_{\mathbf{q}}(\mathbf{r},\mathbf{r}')e^{-i\mathbf{q}\mathbf{T}}. \tag{33}$$

   Thus we can construct $\chi(\mathbf{r},\mathbf{r}')$ from $\chi_{\mathbf{q}}(\mathbf{r},\mathbf{r}')$ where $\mathbf{r}$ and $\mathbf{r}'$ is limited in unit cell. From the above equation, we have

$$\chi(\mathbf{r},\mathbf{r}'+\mathbf{T}) = \frac{1}{N}\sum_{\mathbf{q}} \chi_{\mathbf{q}}(\mathbf{r},\mathbf{r}')e^{-i\mathbf{q}\mathbf{T}} = \int \frac{\Omega d^3q}{(2\pi)^3}\chi_{\mathbf{q}}(\mathbf{r},\mathbf{r}')e^{-i\mathbf{q}\mathbf{T}}. \tag{34}$$

4. Then $\chi_{\mathbf{q}}(\mathbf{r},\mathbf{r}')$ is given as

$$\chi_{\mathbf{q}}(\mathbf{r},\mathbf{r}') = \sum_n \sum_{n'} \int \frac{\Omega d^3k'}{(2\pi)^3} \frac{\bar{\Psi}^*_{\mathbf{k}n}(\mathbf{r})\bar{\Psi}_{\mathbf{k}'n'}(\mathbf{r})\bar{\Psi}^*_{\mathbf{k}'n'}(\mathbf{r})\bar{\Psi}_{\mathbf{k}n}(\mathbf{r})}{\ldots} + \ldots, \tag{35}$$

where $\mathbf{k}' = \mathbf{q} + \mathbf{k}$.

5. Bloch basis (mixed basis) $M_{\mathbf{q}}(\mathbf{r})$ satisfy $M_{\mathbf{q}}(\mathbf{r}+\mathbf{T}) = M_{\mathbf{q}}(\mathbf{r})e^{i\mathbf{q}\mathbf{T}}$. $M_{\mathbf{q}}(\mathbf{r})$ are is normalized in Ω.

$$\chi_{\mathbf{q}}(I,J) = \int_\Omega d^3r \int_\Omega d^3r' M^*_{\mathbf{q}I}(\mathbf{r})\chi_{\mathbf{q}}(\mathbf{r},\mathbf{r}')M_{\mathbf{q}J}(\mathbf{r}') \tag{36}$$

$$\chi_{\mathbf{q}}(\mathbf{r},\mathbf{r}') = \sum_{I,J} \int \frac{\Omega d^3k}{(2\pi)^3} \sum_{I',J'} M_{\mathbf{q}I'}(\mathbf{r})O^{-1}_{\mathbf{q}I'I}\chi_{\mathbf{q}}(I,J)O^{-1}_{\mathbf{q}JJ'}M^*_{\mathbf{q}J'}(\mathbf{r}') \tag{37}$$

26 χ^{0+-}

In $\mathbf{q}$ space, χ^{0+-} is written as

$$\begin{aligned} -\chi^{0+-}_{\mathbf{q}}(\mathbf{r},\mathbf{r}',\omega) &= \sum_{\mathbf{k}n\downarrow}^{\mathrm{occ}} \sum_{\mathbf{k}'n'\uparrow}^{\mathrm{unocc}} \frac{\Psi^*_{\mathbf{k}n\downarrow}(\mathbf{r})\Psi_{\mathbf{k}'n'\uparrow}(\mathbf{r})\Psi^*_{\mathbf{k}'n'\uparrow}(\mathbf{r}')\Psi_{\mathbf{k}n\downarrow}(\mathbf{r}')}{\omega - (\epsilon_{\mathbf{k}'n'\uparrow} - \epsilon_{\mathbf{k}n\downarrow}) + i\delta} \\ &+ \sum_{\mathbf{k}n\downarrow}^{\mathrm{unocc}} \sum_{\mathbf{k}'n'\uparrow}^{\mathrm{occ}} \frac{\Psi^*_{\mathbf{k}n\downarrow}(\mathbf{r})\Psi_{\mathbf{k}'n'\uparrow}(\mathbf{r})\Psi^*_{\mathbf{k}'n'\uparrow}(\mathbf{r}')\Psi_{\mathbf{k}n\downarrow}(\mathbf{r}')}{-\omega - (\epsilon_{\mathbf{k}n\downarrow} - \epsilon_{\mathbf{k}'n'\uparrow}) + i\delta}, \end{aligned} \tag{38}$$

where $\mathbf{k}' = \mathbf{q} + \mathbf{k}$.

In the case with the time-reversal symmetry, we have $\Psi_{\mathbf{k}n\downarrow}(\mathbf{r}) = \Psi^*_{-\mathbf{k}n\downarrow}(\mathbf{r})$ and $\Psi_{\mathbf{k}n\uparrow}(\mathbf{r}) = \Psi^*_{-\mathbf{k}n\uparrow}(\mathbf{r})$. Then Eq.(38) is reduced to be

$$\begin{aligned} -\chi^{0+-}_{\mathbf{q}}(\mathbf{r},\mathbf{r}',\omega) &= \sum_{\mathbf{k}n\downarrow}^{\mathrm{occ}} \sum_{\mathbf{k}'n'\uparrow}^{\mathrm{unocc}} \frac{\Psi^*_{\mathbf{k}n\downarrow}(\mathbf{r})\Psi_{\mathbf{k}'n'\uparrow}(\mathbf{r})\Psi^*_{\mathbf{k}'n'\uparrow}(\mathbf{r}')\Psi_{\mathbf{k}n\downarrow}(\mathbf{r}')}{\omega - (\epsilon_{\mathbf{k}'n'\uparrow} - \epsilon_{\mathbf{k}n\downarrow}) + i\delta} \\ &+ \sum_{\mathbf{k}n\uparrow}^{\mathrm{occ}} \sum_{\mathbf{k}'n'\downarrow}^{\mathrm{unocc}} \frac{\Psi^*_{\mathbf{k}n\uparrow}(\mathbf{r})\Psi_{\mathbf{k}'n'\downarrow}(\mathbf{r})\Psi^*_{\mathbf{k}'n'\downarrow}(\mathbf{r}')\Psi_{\mathbf{k}n\uparrow}(\mathbf{r}')}{-\omega - (\epsilon_{\mathbf{k}'n'\downarrow} - \epsilon_{\mathbf{k}n\uparrow}) + i\delta}. \end{aligned} \tag{39}$$

(in the second term of Eq.(38), we need to rename $\mathbf{k}n$ as $-\mathbf{k}'n'$, and $\mathbf{k}'n'$ as $-\mathbf{k}n$. Then we need to call $\mathbf{k}'$ as $-\mathbf{k}'$.) In the current version **fpgw032f06f.tar.gz**, we use this formula in `hx0fp0` program. (I have not yet implimented the case of non-time reversal symmetry). If you assume paramagnetic case, this agree with the minus half of the usual charge-channel polarization function for the case of time-reversal symmetry, see e.g. Ferdi's GW review Eq.7.3. This Eq.(39) can be expaned by the mixed basis $|M_{\mathbf{q}I}\rangle$ as

$$\begin{aligned} -\langle I|\chi^{0+-}(\mathbf{q},\omega)|J\rangle = -\chi^{0+-}_{\mathbf{q}}(I,J,\omega) &= \sum_{\mathbf{k}n\downarrow}^{\mathrm{occ}} \sum_{\mathbf{k}'n'\uparrow}^{\mathrm{unocc}} \frac{\langle M_{\mathbf{q}I}|\Psi^*_{\mathbf{k}n\downarrow}\Psi_{\mathbf{k}'n'\uparrow}\rangle\langle\Psi^*_{\mathbf{k}'n'\uparrow}\Psi_{\mathbf{k}n\downarrow}|M_{\mathbf{q}J}\rangle}{\omega - (\epsilon_{\mathbf{k}'n'\uparrow} - \epsilon_{\mathbf{k}n\downarrow}) + i\delta} \\ &+ \sum_{\mathbf{k}n\uparrow}^{\mathrm{occ}} \sum_{\mathbf{k}'n'\downarrow}^{\mathrm{unocc}} \frac{\langle M_{\mathbf{q}I}|\Psi^*_{\mathbf{k}n\uparrow}\Psi_{\mathbf{k}'n'\downarrow}\rangle\langle\Psi^*_{\mathbf{k}'n'\downarrow}\Psi_{\mathbf{k}n\uparrow}|M_{\mathbf{q}J}\rangle}{-\omega - (\epsilon_{\mathbf{k}'n'\downarrow} - \epsilon_{\mathbf{k}n\uparrow}) + i\delta}. \end{aligned} \tag{40}$$

To calculate required quantities, we need $\langle m|M_{\mathbf{q}J}\rangle$. This is stored in `MixSpin.*` files. In mode 222 of `hx0fp0`, we directly calculate $\langle m|\chi^{0+-}(\mathbf{q},\omega)|m\rangle$. In mode 223, we calculate full matrix of $\langle I|\chi^{0+-}(\mathbf{q},\omega)|J\rangle$ and take its inverse.

27 —- MEMO_log ——

```
* How to calculate Spin Wave.

The script calj_nlfc_metal summarize peak position and width of SW.
It is in ecal/util/
(it calls calj_interp_mat.F and calj_nlfc_mat.F in it).

We can calculate
       chi^{0+-} for nolfc mode  :  epsPP_lmfh_chipm
       chi^{0+-} full matrix mode:  eps_lmfh_chipm
However, I now think eps_lmfh_chipm mode might be not so meaningful,
because we have not find a way to determined U.

(1) So use epsPP_lmfh_chipm now.
   You need sigm.*, ctrl.*, GWinput, rst.* files.
   chi^{0+-}  is a matix whose dimension is the number of magnetic atoms in a cell.
      epsPP_lmfh_chipm : Its main output is ChiPM*.nlfc.mat.

    Set MagAtom section and q vector in GWinput.
    E.g. MagAtom 1 -3
      We treat two magnetic atoms 1st and 3rd.
      The third atom point opposite direction; AF case or so.
      Note; atoms can be re-orderd by lmf. See a file names ad LMTO.

    format of ChiPM*.nlfc.mat
   ------------------------------------------------------------------
    q(1:3)   omega(Ry)     <eiqr|chipm|eiqr>    <eiqr|chipm|eiqr>^{-1}
    3*real      real        complex               complex
   ------------------------------------------------------------------
   but Need to check normlizaion for eiqr= e^{\i {\bf q} \bfr}}


   ( ChiPM*.nlfc.dat is now deleted; it was a date file of <eiqr|chipm0|eiqr> but
     for \omega-independent U)
     format of ChiPM*.nlfc.dat (no \omega-dependent U---so a little wrong).
     ------------------------------------------------------------------
      q(1:3)   omega(Ry)     <eiqr|chipm|eiqr>    <eiqr|chipm|eiqr>^{-1}
      3*real      real        complex               complex
     ------------------------------------------------------------------
     but Need to check normlizaion for eiqr= e^{\i {\bf q} \bfr}}         )


(2) Perform calj_nlfc_metal (MnAs) or calj_summmary_mat(NiO,MnO)
    Then you will have SWE, FMHF JMAT MMAT. uu0uu1 is generated (U matrix).
    Also generate SW spectrum.
```

```
    These script calls calj_nlfc_mat.F and calj_interp_mat.F internally.
    But these scrips are imperfect yet.
    NEED FIXING!!! In anyway, it is necessary to learn these fortran codes
    (and my SW paper) to calculate spin susceptibility.

    You see static moment and I by echo ChiPM0001.nlfc.mat|calj_nlfc_xxx.


(memo: for olde version;
x calj_det.F     : Calculate spin wave from ChiPM*.mat matrix file.
x calj_detc.F    : modified version of calj_det.F
x calj_search0.F : SW (pole search) for ChiPM*.dat ChiPM*.nolfc.dat files.
x calj_interp.F  : make spectrum function by interpolation (for metal).
x
x calj_summary_ferro : for ferro
x calj_summary_aferro: for aferro
x calj_interp: for metal
)




=============== Below is my personal memo. ======================
bandplot

bandngpsin: (or bandng) kotani
  Usage .
   prepare bnds.ni
   Do bandngspin ni.
   ngraph bandp_plot.spin1.ngp
  note:  bandp.ngp is header part in bin.

---------------------------------------
use ngraph for bandplot

ngraph_band  24June2005 kotani
 bandp.ngp contains functions in sh.
 plbnds.f generates plot.ngp *.ddd
 ngraph_band make  bandp_plot.ngp = bandp.ngp+plot.ngp
 ngraph bandp_plot

-----------------------------------
Total dos calculation

METAL=2 TETRA=1 DOS=-5.3 1.7 NPTS=7001 SAVDOS=t
* Metal =on
* Tetra on
* DOS range
```

```
* SAVDOS t
lmf mnas >& llmf_dos
   mmom.mnas is generated
lmdos mmoms

-------------------------------------
LLBAND --- band mode for lmfgw. kotani
1. SYML
    cp ../rst.si .
    cp ctrl.preprocessed.si ctrl.si
    iactive=t ---> iactive=f
2. echo 0|lmfgw si
3. echo 3|qg4gw
4. echo 4|lmfgw si

/panfs/hpc/home/takao/MARK_rawdata/sitest/sc/bas12.lgc.tppc4.tpdc4.gwbas.float/nk8m
                      MARK_rawdata/ctest/sc/bas12.lfc.tpsc4.tppc4.float/nk8m


We need tetra=1 metal=2 for dos plot!


----------------------------------------
pdos calculation

lmf --wsig:fbz cu2o
cp sigm2.cu20 sigm.cu20
SYMOPS i*i
lmf --pdos:mode=2 cu20
lmdos --pdos:mode=2 cu20

pldos dos.cu20 '-lst='1:4;5:9;102:104'
pldos dos.cu2o -lst='1:4;5:9;102:104;1:50,101:125'


echo /|~/plot/pldos -escl=13.605 -ef=0 -lst="9,11,15;13,17;10,12,16;14,18;102:150:2;104:108:2;2:10
fplot -f plot.dos
echo 400,15,-10,15|~/plot/pldos -escl=13.605 -ef=0 -lst="1" -fplot dostot.nio

strange???
In MnAs, total channels are 100 as
25+25+25+25+0+0=100 for each spin.
But lm lmdos shows
 Channels in dos file generated by LMDOS:
 site class label   spin-1                      spin-2
    1    1   C1     1:49:2                      2:50:2
    2    4   C12    51:99:2                     52:100:2
    3    2   A1     101:149:2                   102:150:2
```

```
    4    5   A12    151:199:2                    152:200:2
    5    3   EA1    201:249:2                    202:250:2
    6    6   EA12   251:299:2                    252:300:2
 Exit 0 LMDOS
This maybe bcause empty sphere is automatically omitted.
So total is 200 channel.


----------------------------------------
ErAs case: How to get sigma without MZ?

lmf eras --wsig:fbz
cp sigm2.eras sigm.eras
Change ctrl.eras as
        nk=3
        Remove MZ
        RDSIG =10012
lmf eras --wgis:newkp
mv sigm2.eras sigm.eras
Change ctrl.eras backs up origial except MZ.
        nk=8     (move back to original)
        RDSIG=12 (move back to original)


-------------------------------------------
syml.coo
41 -.5 .5 .5 0 0 0    X G
41 0 0 0 -.25 .75 -.25    G L
41 -.75 .25 .25 -.0625 .875 -.0625   L U
41 -.0625 .875 -.0625 .25 .25 .25    U T
41 .25 .25 .25 0 0 0    T G
0 0 0 0 0 0 0

lmf --band:fn=syml coo >llmf_band
plbnds -fplot -ef=0 -scl=13.605 -spin2 eras
fplot -f plot.plbnds


-------------
Don't forget
  "Position of atom is not necessarily fixed by ctrl"
  Look into LMTO file.


-------------
Fe and Ni
plan mode3, mode5, 1shot_lmfh, 1shot_mode3
Use FRZWF=t for mode5.
```

```
(Be careful convergence of QSGW--- bands around Ef is somehow unstable).


-------------
/home/takao/DATA2/Fe/FeLDA_5.408_tpd4_303030/line110
>>> 12.10/(2*3.1415926/(5.408*0.529177)*0.066666)**2/2.
282.39834188678122


-------------
iarg = iargc()
call getarg(0,str)
call getarg(1,str) !1st argment
call system('ls')  !system call


-------------
*Saguaro is not so simple.
 ---When I has a bug resulting core.
 The error was occured, some steps ahead of the last of the end of check write.

*  " subroutine(number of arguments were different)" caused "segmentation fault".


-------------
*stop occured for
        if(sum(abs(add-nadd))>1d-10) stop "sexc: abs(add-nadd))>1d-10"
     in x0kf_v4h for Gd case.
     This is maybe because of poor accuracy for <QpGforEPS>
     setted in GWinput.

-------------
!!!  segmentation fault when dw=0.0005 --->Need to be 0.001
 Too large memory requirement
```

28 Samples in al1 (old document)

(—- **This is an old document. This is just for history.**—

There are samples in
`http://al1.phys.sci.osaka-u.ac.jp/~kotani/data/` These samples are not necessary to be converged results. Rahter most of all are test examples, maybe **far from the converged results**.

- data/LDA —- calculations in LDA. I now have LDA results on AlAs, AlP, AlSb, BeTe, CaB6,CaO,CdO,CdS,CdTe,C,Fe,GaP,GaSb,GaS, Ge,InAsInP,InSb. We did *GW* for them but these are not in it. ctrl.* in it will be useful. See e.g. `alastest/LDA.bigbas/BandLDA.UP.pdf` (energy band).

- data_020 — old examples by fpgw020. These are examples and current version is not compatible with these results.
- data_026 — old examples by fpgw026. With ctrl.*, rst.*, GWIN0, GWIN_V2 in each directory and corresponding command, you can reproduce these results. (GWIN0 plus GWIN_V2 are automatically converten into GWinput at the head of **qg4gw** called from a script).
- data_030 — latest results.

29 gwpara_lmf for parallel computaion test (old document)

(This was in fpgw025. I still keep this, but I have not cheked it. So I need to have to debug it to make it work.)

This **gwpara_lmf** is a test script for paralleling the computaion. In **gwpara_lmf**, you find these lines.

```
################ PARALLEL1 ##############################
@ count = 1
while(-e QPNT.$count)
  echo  ' '
  echo ' --- ' $count 'th loop starting with ' QPNT.$count
  echo "3 $count"|$nfpgw/hsfp0   >lsxC.$count
  @ count = $count + 1
end
################ End of PARALLEL1 ######################

Here you can see "3 $count". The first 3 means the core-exchange mode;
$count means $nfpgw/hsfp0 use QPNT.$count instead of QPNT.
If you invoke hsfp0 as usual like  echo 3|$nfpgw/hsfp0, no second input
corresponding  to $count causes the reading error and goes to use QPNT.
Or you can supply "3 0" from standard input instead. This while loops
should be done completely parallel without the conflict of the output
files.

The next parallel section is
################ PARALLEL2 ##############################
@ count = 1
while( $count <= $nmachine )
  @ kinit = $kinitlist[$count]
  @ kend =  $kendlist[$count]
  echo ' ---  ' $count ' th cycle.  From ' $kinit ' to ' $kend
  echo "0  $kinit $kend" |$nfpgw/hvccfp0  > lvcc.$count
  echo "1  $kinit $kend" |$nfpgw/hx0fp0   > lx0.$count
  @ count = $count + 1
end
################ End of PARALLEL2 ######################
, which is for hvccfp0 and hx0fp0. As you see, these takes three standard
inputs arguments, e.g., "0  $kinit $kend". As is the same as hsfp0,
echo 0|hvccfp0 and echo 1|hx0fp0 gives the ordinary behevior with the
readin error for the second and the third arguments. in the case of
echo "0  $kinit $kend" |$nfpgw/hvccfp0, hvccfp0 just calculate
the coulomb matrix only between $kinit-th and $kend-th k vector
in the IBZ+Q0P.

The last parallel section is
################ PARALLEL3 ##############################
@ count = 1
while( -e QPNT.$count)
  echo ' --- ' $count 'th loop starting with ' QPNT.$count
  echo "1 $count"|$nfpgw/hsfp0 >lsx.$count
  echo "2 $count"|$nfpgw/hsfp0 >lsc.$count
  @ count = $count + 1
end
################ End of PARARELL3 ######################
, similar with the PARALLEL1.
```

At first in gwpara_lmf, we set the number of machines as `@ nmachine = 2` It is passed to a new routine `parainfo`, which make `X0KDIV` (include the information of deviding the set of **k** into machines), and devide QPNT int QPNT.*. NQIBZ including `nqibz` and `nq0i` is also generagted. The routine mergewv is just in order to merge WVR.* to WVR (and also for WVI).

30 Changes from the previous version fpgw020 to fpgw025 (old document)

(—- **This is an old document. This is just for history.**— fpgw025 is improvement from fpgw030—now things changes from fpgw025.)

The new FP-LMTO as FP.lm6.12 can treat the local orbital (Singh 91) as is explained in http://www.wien2k.at/lapw/index.html. So it can treat one-additional channel per l. They could be important for these cases; (1) we can treat not only $3d$ but also $4d$ at the same time as valence in the case of Cu — $4d$ can be treated by the local orbital. It is not important for occupied states but it could affect on the GW results though the unoccupied states; (2) we can treat semi-core $3d$ states by the local orbital, and $4d$ as usual in the case of GaAs.

The GW code fpgw025 is a generalized version which can treat the multiple atomic-like argumentation waves per l. The previous version fpagw020 could treat just two argumentation waves per l, so-called ϕ and $\dot{\phi}$. On the other hand, fpgw025 can treat any number per l. But the number is limited to up to three (ϕ, $\dot{\phi}$ and the local orbital), when you use FP.lm6.12.

Further I added a minor change for tetwt4.f. (the tetrahedron-weight routine. See subroutine intttvc2 in the code). It is only a point which might change results for given input in comparison with fpgw020. I found that some weight was missed in intttvc in fpgw020— however the changes will affect little.

As the GW procedure, it is essentially the same but GWIN is replaced with GWIN_V2. You have to edit GWIN_V2.tmp generated by **mkGWIN_lmf**, instead of GWIN . With GWIN0, GWIN_V2, ctrl.*, rst.*, QPNT, you can invoke **gw_lmfh** to get the final results of QP energies.

Files, binaries are renamed and unified,

- GWIN → GWIN_V2

  These is some difference in the product basis section. But GWIN_V2.tmp contains comments to understand it. Or see explanations after.

- A new routine **convgwin** can make GWIN_V2 from GWIN.

- DATA4GW → DATA4GW_V2

  Format to store data is a bit modified.

- **rdata4gw → rdata4gw_v2**

  This corresponds to the change above.

- RBU RHBU CBU CHBU RBD RHBD CBD CHBD → CPHI.

  It contains coefficients of eigenfunctions for atomic-like argumentation waves for each l and MT.

- PPBRD* → PPBRD_V2_*

  Radial integrals on each MT, symbolically written as $\int \phi(r)\phi(r)B(r)dr$. These are generated by hbasfp0. Format to store data is a bit modified.

- LMTO contains nnv. It was fixed to 1 in fpgw020. But now it has the new meaning "the maximum number of radial functions for valence". So it may cause a problem if you do new versions of hqpemetal, hx0fp0, hsfp0 or something with the old files. Maybe hqpemetal might be a main possibility which you might do—then just change the number nnv from 1 to 2.

- PHIU PHID PHICU PHICD → PHICV.

  This contains all the radial functions.

In addition, we made some simplifications for the code. Especially now we just use only `ppb` array for the radial integrals $\langle\phi\phi B\rangle$. They are generated by **hbasfp0** and stored into PPBRD_V2_*. Then it is read through ppbafp_v2 in ppbafp.fal.f, and used in hx0fp0.m.f and hsfp0.m.f.

As for this document itself, main modification is in the explanation in GWIN_V2. I also changed file names and so on as I explained above. Further I added a section Sec.15. I also add little modifications in the formalism note in Sec.**??**. In addion, I add some small changes.

• `esmr` is $E_{\rm smear}$ in Section 3. Use 0.01Ry or something (not zero due to the numerical reason) for insulators (set it smaller than band gap. 0.001Ry will also work). But be careful to choose this value in the case of metal.

References

[1] Kotani, Mark, Sergey, PRB76 165106 (2007). Main reference as Ref.I. EQ. means equation in this literature.

[2] Kotani, van Schilfgaarde. Spin wave paper in arxiv.

[3] Sergey V. Faleev, Mark van Schilfgaarde, and Takao Kotani, All-electron self-consistent GW approximation: Application to Si, MnO, and NiO, PRL

[4] L. Hedin, Phys. Rev. **139**, A796 (1965); L. Hedin and S. Lundqvist, in *Solid State Physics*, edited by H. Ehnrenreich, F. Seitz, and D. Turnbull (Academic, New York, 1969), Vol. 23, p. 1.

[5] F. Araysetiawan and O. Gunnarsson, Rep. Prog. Phys. **61**, 237-312 (1998).

[6] W. G. Aulbur, L. Jönsson, and J. W. Wilkins, *Solid State Physics*, edited by H. Ehnrenreich, F. Saepen (Academic, New York, 2000), Vol. 54, p. 1.

[7] E.L. Shirley, Z. Zhu, S.G. Louie, Phys. Rev. B **56**, 6648 (1997).

[8] N. Hamada, M. Hwang and A.J. Freeman, Phys. Rev. B **41**, 3620 (1990).

[9] P.E Blöchl, Phys. Rev. B **50**, 17953 (1994).

[10] B. Arnaud and M. Alouani, Phys. Rev. B **62**, 4464 (2000).

[11] F. Aryasetiawan, Phys.Rev. B **46**, 13051 (1992).

[12] See Refs. in [5, 6] and in F. Aryasetiawan, in *Advances in Condensed Matter Science*, edited by I.V. Anisimov (Gordon and Breach, 2000), Vol 1, p.33.

[13] F. Aryasetiawan, O Gunnarsson, Phys.Rev. B **49**, 16214 (1994)

[14] F. Aryasetiawan and O. Gunnarsson, Phys. Rev. Lett. **74**, 3221 (1995).

[15] J. Rath and A.J. Freeman, Phys. Rev. B **11**, 2109 (1975).

[16] M. Methfessel, M. van Schilfgaarde, and R. A. Casali, "A full-potential LMTO method based on smooth Hankel functions," in *Electronic Structure and Physical Properties of Solids: The Uses of the LMTO Method*, *Lecture Notes in Physics*, **535**. H. Dreysse, ed. (Springer-Verlag, Berlin) 2000.

[17] E.Bott, M.Methfessel, W.Krabs, and P.C.Schmidt, J. Math. Phys. **39**,3393 (1998).

Appendix. Some are useful for developers. But some of them (after Sec.B) might be too old; they may not fit to the latest codes.

A Memo

* Log for development is in `fpgw_version_log`...

* Now fpgw program's shows its version. e.g. type qg4gw(or hxfp0 or any program) and put -9999 into it. Then it shows version number. It is also shown the top of cosole output.

B Used formulas

$$\exp(i\mathbf{kr}) = 4\pi \sum_L i^l j_l(|\mathbf{k}|r) Y_L^*(\widehat{\mathbf{k}}) Y_L(\widehat{\mathbf{r}}) \tag{41}$$

$$\frac{2l+1}{4\pi} P_l(\cos\Theta) = \sum_m Y_L^*(\widehat{\mathbf{r}}_1) Y_L(\widehat{\mathbf{r}}_2) \qquad [\cos\Theta = \widehat{\mathbf{r}}_1 \cdot \widehat{\mathbf{r}}_2]. \tag{42}$$

$$\langle P_{\mathbf{G}}^{\mathbf{k}} | P_{\mathbf{G}'}^{\mathbf{k}} \rangle = \Omega \delta_{\mathbf{G},\mathbf{G}'} - \sum_{a,L} \exp(i(\mathbf{G}-\mathbf{G}')\mathbf{R}_{\mathbf{a}}) \times Y_L(\widehat{\mathbf{k}+\mathbf{G}'}) Y_L(\widehat{\mathbf{k}+\mathbf{G}}) \times \int_0^{R_a} j_l(|\mathbf{k}+\mathbf{G}|r) j_l(|\mathbf{k}+\mathbf{G}'|r) 4\pi^2 r^2 dr, \tag{43}$$

C Sperical Harmonics and Real harmonics used in GW (and lmf).

In our GW code, we user real harmonics $y_{lm}(\hat{\mathbf{r}})$, instead of the usual sperical (complex) harmonics $Y_{lm}(\hat{\mathbf{r}})$ in the real implimentation. The coefficients of eigenfunctions and so on are ordered as, e.g. $(m=-2, m=-1, m=0, m=1, m=2)$ for $l=2$.

$y_{lm}(\hat{\mathbf{r}})$ is defined from $Y_{lm}(\hat{\mathbf{r}})$. (Note $\hat{\mathbf{r}}=(\theta,\phi)$). The definition of the real harmonics is the same as what is used in lmf.

$$y_{l0}(\hat{\mathbf{r}}) \equiv Y_{l0}(\hat{\mathbf{r}}). \tag{44}$$

$$y_{lm}(\hat{\mathbf{r}}) \equiv \frac{1}{\sqrt{2}}[(-1)^m Y_{lm}(\hat{\mathbf{r}}) + Y_{l-m}(\hat{\mathbf{r}})]. \tag{45}$$

$$y_{l-m}(\hat{\mathbf{r}}) \equiv \frac{1}{\sqrt{2}i}[(-1)^m Y_{lm}(\hat{\mathbf{r}}) - Y_{l-m}(\hat{\mathbf{r}})]. \tag{46}$$

, where $m>0$. Or Equivalently,

$$Y_{l0}(\hat{\mathbf{r}}) \equiv y_{l0}(\hat{\mathbf{r}}). \tag{47}$$

$$Y_{lm}(\hat{\mathbf{r}}) \equiv \frac{(-1)^m}{\sqrt{2}}[y_{lm}(\hat{\mathbf{r}}) + iy_{l-m}(\hat{\mathbf{r}})]. \tag{48}$$

$$Y_{l-m}(\hat{\mathbf{r}}) \equiv \frac{1}{\sqrt{2}}[y_{lm}(\hat{\mathbf{r}}) - iy_{l-m}(\hat{\mathbf{r}})]. \tag{49}$$

.

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

$$Y_{lm}(\theta,\phi) = (-1)^m \left[\frac{(2l+1)(l-m)!}{4\pi(l+m)!}\right]^{\frac{1}{2}} P_l^m(\cos(\theta))e^{im\phi}, \tag{50}$$

$$P_l^m(x) = \frac{(1-x^2)^{m/2}}{2^l l!}\frac{d^{l+m}}{dx^{l+m}}(x^2-1)^l \tag{51}$$

.

See
(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.

D Expansion of the eigenfunction

The Bloch sum of the MTO, $\chi^{\mathbf{k}s}(\mathbf{r})$, is expressed by a linear combination of local orbitals $A_{au}(\mathbf{r}) \equiv \{\phi_{aL}(r)Y_L(\hat{\mathbf{r}}), \dot{\phi}_{aL}(r)Y_L(\hat{\mathbf{r}})\}$ within each MT. $\phi_{aL}(r)$ and $\dot{\phi}_{aL}(r)$ denote solutions of the radial Schrödinger equations and their energy derivatives, respectively. ($\dot{\phi}$ does not necessarily to be such energy derivatives). $u \equiv (L, I_{\rm P})$ is the composite index where $I_{\rm P}$ takes 0 for ϕ, or 1 for $\dot{\phi}$. a is the index to specify atom in the primitive cell. The MTO basis is specified by $s \equiv ajL$, where $L \equiv (l, m)$ is the angular momentum index, and j is the additional index (principle quantum number or so). $A_{au}(\mathbf{r})$ makes normalized-orthogonal basis for each MT a. The MTO can be written as

$$\begin{aligned}\chi^{\mathbf{k}s}(\mathbf{r}) &= \sum_{au} C^{\mathbf{k}s}_{au} A^{\mathbf{k}}_{au}(\mathbf{r}) \quad \text{if } \mathbf{r} \in \text{any MT} \\ &= H^{\mathbf{k}s}(\mathbf{r}) \quad \text{otherwise,}\end{aligned} \tag{52}$$

where we use the Bloch sums,

$$A^{\mathbf{k}}_{au}(\mathbf{r}) \equiv \sum_{\mathbf{T}} A_{au}(\mathbf{r} - \mathbf{R_a} - \mathbf{T}) \exp(i\mathbf{kT}), \tag{53}$$

$$H^{\mathbf{k}s}(\mathbf{r}) \equiv \sum_{\mathbf{T}} H_s(\mathbf{r} - \mathbf{R_a} - \mathbf{T}) \exp(i\mathbf{kT}). \tag{54}$$

$\mathbf{R_a}$ is the position of the atom a in the primitive unit cell. $H^{\mathbf{k}s}(\mathbf{r})$ is the envelope functions (we used the smooth Hankel functions in the NFP code). The eigenfunction $\Psi^{\mathbf{k}n}$ is expanded as the linear combination of the MTO as

$$\Psi^{\mathbf{k}n}(\mathbf{r}) = \sum_{s} z^{\mathbf{k}n}_{s} \chi^{\mathbf{k}s}(\mathbf{r}) \tag{55}$$

$$= \sum_{au} \alpha^{\mathbf{k}n}_{au} A^{\mathbf{k}}_{au}(\mathbf{r}) + \sum_{\mathbf{G}} \beta^{\mathbf{k}n}_{\mathbf{G}} P^{\mathbf{k}}_{\mathbf{G}}(\mathbf{r}), \tag{56}$$

where the interstitial plane wave (IPW) $P^{\mathbf{k}}_{\mathbf{G}}(\mathbf{r})$ is defined as

$$\begin{aligned}P^{\mathbf{k}}_{\mathbf{G}}(\mathbf{r}) &= 0 \quad \text{if } \mathbf{r} \in \text{any MT} \\ &= \exp(i(\mathbf{k}+\mathbf{G})\mathbf{r}) \quad \text{otherwise.}\end{aligned} \tag{57}$$

The coefficients are calculated as

$$\alpha^{\mathbf{k}n}_{au} = \sum_{s} C^{\mathbf{k}s}_{au} z^{\mathbf{k}n}_{s} \tag{58}$$

$$\beta^{\mathbf{k}n}_{\mathbf{G}} = \sum_{\mathbf{G}'s} \langle P^{\mathbf{k}}_{\mathbf{G}} | P^{\mathbf{k}}_{\mathbf{G}'} \rangle^{-1} \langle P^{\mathbf{k}}_{\mathbf{G}'} | H^{\mathbf{k}s} \rangle z^{\mathbf{k}n}_{s}, \tag{59}$$

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`.

`lm-6.14/gw/sugw.f` called from `lm-6.14/lmfgw.m.f` is a main part to generate this expansion. Some key quantities in `lm-6.14/gw/sugw.f` are

.

- $z^{\mathbf{k}n}_{s}$ = `zegf(i,j); i=1,ndimh; j=1,ndimh` (`i` is for for basis, and `j` is for band index.)
- $\alpha^{\mathbf{k}n}_{au}$ = `cphi`
- $\langle \phi Y_L$ or $\dot{\phi} Y_L | \chi^{\mathbf{k}s} \rangle$= `phichi`
- `phichi` is constructed from `phihd`, and `bmat` $\times$ `phipkl`.

- `bmat` are generated in `hxp_bl` $\in$ `augm_q`. It is the coefficients for the expansion of $H^{\mathbf{k}s}(\mathbf{r})$ at the another MT center.

[In `ng0.m.f`, `QpGcutHakel` is assumed as $=$ `1.5` $*$ `QpGcut_psi` now. But it is not justified enough. You will be able to utilize more reasonable ones which was used in the LDA calculations.]

$\alpha_{au}^{\mathbf{k}n}$ is calculated by the subroutine `getcoeffas` in `ng0.m.f`. The subroutine `matgg2` $\in$ `mkppovl2` $\in$ `pplmat2` in `pplmat.f` calculates $\langle P_{\mathbf{G}}^{\mathbf{k}}|P_{\mathbf{G}'}^{\mathbf{k}}\rangle$ through

$$\langle P_{\mathbf{G}}^{\mathbf{k}}|P_{\mathbf{G}'}^{\mathbf{k}}\rangle = \Omega\delta_{\mathbf{G},\mathbf{G}'} - \sum_{a,L}\exp(i(\mathbf{G}'-\mathbf{G})\mathbf{R}_{\mathbf{a}}) \times Y_L(\widehat{\mathbf{G}'-\mathbf{G}}) \times \int_a \exp(i(\mathbf{G}'-\mathbf{G})\mathbf{r})d^3r. \tag{60}$$

$\langle P_{\mathbf{G}'}^{\mathbf{k}}|H^{\mathbf{k}s}\rangle$ is also calculated in `pplmat2` through the plane wave expansion of $H^{\mathbf{k}s}$ (Eq.(9.4) of Ref.[17]). Then `pplmat2` gives the the coefficients $\beta_{\mathbf{G}}^{\mathbf{k}n}$.

E Expansion of the Coulomb matrix

For the numerical evaluation of the Coulomb matrix, we can not avoid a cutoff procedure, which means to use the another kind of IPW $\bar{P}_{\mathbf{G}}^{\mathbf{k}}$ in place of $P_{\mathbf{G}}^{\mathbf{k}}$. Here $\bar{P}_{\mathbf{G}}^{\mathbf{k}}$ is defined by subtracting the MT contributions up to the finite angular momentum cutoff $l_{\rm Pmax}$, that is, $\bar{P}_{\mathbf{G}}^{\mathbf{k}} \equiv (1-\sum_{aL}\hat{P}_{aL})\exp(i(\mathbf{k}+\mathbf{G})\mathbf{r})$. Here $\hat{P}_{aL}$ denotes the projection operator of aL contrubution. l of L should be $\leq l_{\rm Pmax}$. Larger $l_{\rm Pmax}$ means that $\bar{P}_{\mathbf{G}}^{\mathbf{k}}$ is closer to $P_{\mathbf{G}}^{\mathbf{k}}$. We take $l_{\rm Pmax} = 2\times l_{\rm max}$ where $l_{\rm max}$ denotes the maxmum angular momentum of $A_{au}(\mathbf{r})$ (See `hvccfp0.f`). So $l_{\rm Pmax}=8$, which seems to be large enough, if we take $l_{\rm max}=4$. However, in order to make things consistent, we should not mix up $P_{\mathbf{G}}^{\mathbf{k}}$ with $\bar{P}_{\mathbf{G}}^{\mathbf{k}}$. For example, the matrix elements $\langle P_1|v|P_2\rangle$ can be calculated as

$$\langle P_1|v|P_2\rangle = \sum_{\mathbf{G_1}'\mathbf{G_1}''\mathbf{G_2}'\mathbf{G_2}''} \langle P_1|\bar{P_{1'}}\rangle\langle\bar{P_{1'}}|\bar{P_{1''}}\rangle^{-1}\langle\bar{P_{1''}}|v|\bar{P_{2''}}\rangle\langle\bar{P_{2''}}|\bar{P_{2'}}\rangle^{-1}\langle\bar{P_{2'}}|P_2\rangle, \tag{61}$$

where $1\equiv(\mathbf{k},\mathbf{G_1})$ and so on. The matrix elements $\langle\bar{P}_{2''}|\bar{P}_{2'}\rangle^{-1}\langle\bar{P_{2'}}|P_2\rangle$ are reserved in the variable `ppx` and written into the file `PPOVL` in `rdata4gw`. The base $|M_I^{\mathbf{k}}\rangle$ in Eq.(**??**) corresponds to $|\bar{P}_1\rangle$, and $|\tilde{M}_I^{\mathbf{k}}\rangle$ corresponds to $|\bar{P}_{1'}\rangle\langle\bar{P}_{1'}|\bar{P}_1\rangle^{-1}$.

F Expansion of the eigenfunction $\Psi^{\mathbf{k}n}$ in LAPW

The APW, $\chi^{\mathbf{k}+\mathbf{G}}(\mathbf{r})$, is expressed by a linear combination of local orbitals $A_{au}(\mathbf{r}) \equiv \{\phi_{aL}(r)Y_L(\hat{\mathbf{r}}), \dot{\phi}_{aL}(r)Y_L(\hat{\mathbf{r}})\}$ within each Muffin-Tin. $\phi_{aL}(r)$ and $\dot{\phi}_{aL}(r)$ denote solutions of the radial Schrödinger equations and their energy derivatives, respectively. ($\dot{\phi}$ does not necessarily to be such energy derivatives). $u\equiv(L,I_{\rm P})$ is the composite index where $I_{\rm P}$ takes 0 for ϕ, or 1 for $\dot{\phi}$. a is the index to specify atom in the primitive cell. The APW basis is specified by $s\equiv ajL$, where $L\equiv(l,m)$ is the angular momentum index, and j is the additional index (principle quantum number or so). $A_{au}(\mathbf{r})$ makes normalized-orthogonal basis for each MT a. The APW can be written as

$$\begin{aligned}\chi^{\mathbf{k}+\mathbf{G}}(\mathbf{r}) &= \sum_{au} C_{au}^{\mathbf{k}+\mathbf{G}} A_{au}^{\mathbf{k}}(\mathbf{r}) \quad \text{if } \mathbf{r}\in \text{any MT} \\ &= \exp(i(\mathbf{k}+\mathbf{G})\mathbf{r}) \quad \text{otherwise,}\end{aligned} \tag{62}$$

where we use the Bloch sums,

$$A_{au}^{\mathbf{k}}(\mathbf{r}) \equiv \sum_{\mathbf{T}} A_{au}(\mathbf{r}-\mathbf{R}_{\mathbf{a}}-\mathbf{T})\exp(i\mathbf{k}\mathbf{T}), \tag{63}$$

$\mathbf{R_a}$ is the position of the atom a in the primitive unit cell. The eigenfunction $\Psi^{\mathbf{k}n}$ is expanded as the linear combination of the APW as

$$\Psi^{\mathbf{k}n}(\mathbf{r}) = \sum_{\mathbf{G}} z_n^{\mathbf{k}+\mathbf{G}} \chi^{\mathbf{k}+\mathbf{G}}(\mathbf{r}) \tag{64}$$

$$= \sum_{au} \alpha_{au}^{\mathbf{k}n} A_{au}^{\mathbf{k}}(\mathbf{r}) + \sum_{\mathbf{G}} z_n^{\mathbf{k}+\mathbf{G}} P_{\mathbf{G}}^{\mathbf{k}}(\mathbf{r}), \tag{65}$$

where n is the band index, and the interstitial plane wave (IPW) $P_{\mathbf{G}}^{\mathbf{k}}(\mathbf{r})$ is defined as

$$P_{\mathbf{G}}^{\mathbf{k}}(\mathbf{r}) = 0 \quad \text{if } \mathbf{r} \in \text{any MT}$$

$$= \exp(i(\mathbf{k}+\mathbf{G})\mathbf{r}) \quad \text{otherwise.} \tag{66}$$

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`. The coefficients $\alpha_{au}^{\mathbf{k}n}$ can be calculated as

$$\alpha_{au}^{\mathbf{k}n} = \sum_{\mathbf{G}} C_{au}^{\mathbf{k}+\mathbf{G}} z_n^{\mathbf{k}+\mathbf{G}}. \tag{67}$$

G Notations (Usuda's note from here)

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

$$\mathbf{k}(i_1,i_2,i_3) = 2\pi\left(\frac{i_1}{N_1}\mathbf{b}_1 + \frac{i_2}{N_2}\mathbf{b}_2 + \frac{i_3}{N_3}\mathbf{b}_3\right) \tag{68}$$

for $i_1 = 0,1,2,...N_1 - 1$ and so on. Within the volume $V = \Omega N_c = \Omega N_1 N_2 N_3$, 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

$$\int_V F^{\mathbf{k}}(\mathbf{r}) G^{\mathbf{k}'}(\mathbf{r}) d^3r = \delta_{\mathbf{k}\mathbf{k}'} N_c \int_\Omega F^{\mathbf{k}}(\mathbf{r}) G^{\mathbf{k}'}(\mathbf{r}) d^3r \tag{69}$$

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;

$$\tilde{\Psi}^{\mathbf{k}n}(\mathbf{r}) \equiv \sqrt{N_c}\Psi^{\mathbf{k}n}(\mathbf{r}) \tag{70}$$

$$\int_\Omega |\tilde{\Psi}^{\mathbf{k}n}(\mathbf{r})|^2 d^3r = 1. \tag{71}$$

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

$$\tilde{\Psi}^{\mathbf{k}n}(\mathbf{r}) = \sum_{au} \alpha_{au}^{\mathbf{k}n} A_{au}^{\mathbf{k}}(\mathbf{r}) + \sum_{\mathbf{G}} \beta_{\mathbf{G}}^{\mathbf{k}n} P_{\mathbf{G}}^{\mathbf{k}}(\mathbf{r}), \tag{72}$$

$$A_{au}^{\mathbf{k}}(\mathbf{r}) \equiv \sum_{\mathbf{T}} A_{au}(\mathbf{r} - \mathbf{R}_a - \mathbf{T}) e^{i\mathbf{k}\cdot\mathbf{T}}, \tag{73}$$

$$P_{\mathbf{G}}^{\mathbf{k}}(\mathbf{r}) \equiv 0 \quad \text{if } \mathbf{r} \in \text{any MT}$$

$$\equiv e^{i(\mathbf{k}+\mathbf{G})\cdot\mathbf{r}} \quad \text{otherwise,} \tag{74}$$

where $A^{\mathbf{k}}_{au}(\mathbf{r})$ is the Bloch sum of the atomic function $A_{au}(\mathbf{r})$ in the a-site muffin-tin (MT) sphere. $P^{\mathbf{k}}_{\mathbf{G}}(\mathbf{r})$ 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 specify the argumentaion basis. $A^{\mathbf{k}}_{au}(\mathbf{r})$ is orthnormlized as

$$\int_{|\mathbf{r}|<V_a} A_{au}(\mathbf{r}) A_{au'}(\mathbf{r}) d^3r = \delta_{uu'}, \tag{75}$$

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

$$\frac{1}{N_c}\int_V \{A^{\mathbf{k}}_{au}(\mathbf{r})\}^* A^{\mathbf{k}'}_{a'u'}(\mathbf{r}) d^3r = \delta_{\mathbf{k}\mathbf{k}'}\delta_{aa'}\delta_{uu'}\int_\Omega |A^{\mathbf{k}}_{au}(\mathbf{r})|^2 d^3r = \delta_{\mathbf{k}\mathbf{k}'}\delta_{aa'}\delta_{uu'} \tag{76}$$

$$\frac{1}{N_c}\int_V \{P^{\mathbf{k}}_{\mathbf{G}}(\mathbf{r})\}^* P^{\mathbf{k}'}_{\mathbf{G}'}(\mathbf{r}) d^3r = \delta_{\mathbf{k}\mathbf{k}'}\int_\Omega \{P^{\mathbf{k}}_{\mathbf{G}}(\mathbf{r})\}^* P^{\mathbf{k}'}_{\mathbf{G}'}(\mathbf{r}) d^3r = \delta_{\mathbf{k}\mathbf{k}'}\int_\Omega P^{\mathbf{0}}_{\mathbf{G}'-\mathbf{G}}(\mathbf{r}) d^3r. \tag{77}$$

H Mixed basis

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

$$B_{a\mu}^{\mathbf{k}}(\mathbf{r}) = \sum_{\mathbf{T}} B_{a\mu}(\mathbf{r} - \mathbf{R}_a - \mathbf{T}) e^{i\mathbf{k}\cdot\mathbf{T}}, \tag{78}$$

where $B_{a\mu}(\mathbf{r})$ is the product function, which is real and is zero for $|\mathbf{r}| > V_a$ (See Sec.H.1). We set up $B_{a\mu}(\mathbf{r})$ as ortho-normalized;

$$\int_{|\mathbf{r}|<V_a} B_{a\mu}(\mathbf{r}) B_{a\mu'}(\mathbf{r}) d^3r = \delta_{\mu\mu'}. \tag{79}$$

However the overlap matrix

$$O_{IJ}^{\mathbf{k}} = \int_\Omega \{M_I^{\mathbf{k}}(\mathbf{r})\}^* M_J^{\mathbf{k}}(\mathbf{r}) d^3r, \tag{80}$$

is necessary because $\{P_{\mathbf{G}}^{\mathbf{k}}(\mathbf{r})\}$ are not ortho-normal (in Ω) though they are apparently orthogonal to the space of $\{B_{a\mu}^{\mathbf{k}}(\mathbf{r})\}$. We therefore define the dual-basis function:

$$\tilde{M}_I^{\mathbf{k}}(\mathbf{r}) = \sum_{I'} M_{I'}^{\mathbf{k}}(\mathbf{r}) \{O^{\mathbf{k}}\}_{I'I}^{-1}. \tag{81}$$

We can expand the quantity $F^{\mathbf{k}}(\mathbf{r})$ such as

$$\begin{cases} F^{\mathbf{k}}(\mathbf{r}) = \sum_I M_I^{\mathbf{k}}(\mathbf{r}) F_I(\mathbf{k}) \\ F_I(\mathbf{k}) = \int_\Omega \{\tilde{M}_I^{\mathbf{k}}(\mathbf{r})\}^* F^{\mathbf{k}}(\mathbf{r}) d^3r. \end{cases} \tag{82}$$

H.1 Product function (hbasfp0)

We denote the radial function of atom a as

$$u_{apl\sigma}(r) = r\phi_{apl\sigma}(r), \tag{83}$$

where the index p takes 1 for ϕ and 2 for $\dot{\phi}$ (if you include local orbital with fpgw025 code, p also takes 3.); in addition, p takes indexes for core functions: we combine core and valence functions (sub. `phivc`). Note that the *true* radial function is $\phi_{apl\sigma}(r) = u_{apl\sigma}(r)/r$. Normalization is $1 = \int_0^{S_a} \{u_{apl\sigma}(r)\}^2 dr = \int_0^{S_a} \{\phi_{apl\sigma}(r)\}^2 r^2 dr$, where S_a is the radiaus of the MT-sphere. The function $u_{apl\sigma}(r)$ is stored in `phitot`. [In fpgw025, the orthonomalized radial functions $u_{apl\sigma}(r)$ are stored in `phitoto` as `phitot` of fpgw020, though we also have the un-orthonormalized ones in `phitotr`.]

When producing the product functions, we use spin-averaged function `phiav` (sub. `basnfp`).

$$u_{apl}(r) = \frac{1}{N_{\rm spin}} \sum_\sigma u_{apl\sigma}(r). \tag{84}$$

From them, we make the product functions `rprod`,

$$\tilde{b}_{al\nu}(r) = \frac{1}{r} u_{apl}(r) u_{ap'l'}(r) = r\phi_{apl}(r)\phi_{ap'l'}(r), \tag{85}$$

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 rather given as

$$\tilde{B}_{al\nu}(r) = \frac{1}{r}\tilde{b}_{al\nu}(r), \tag{86}$$

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

Then we calculate the overlap matrix `ovmt`,

$$O_{\nu_1\nu_2} = \int_0^{S_a} \tilde{B}_{al\nu_1}(r)\tilde{B}_{al\nu_2}(r)r^2dr = \int_0^{S_a} \phi_{ap_1l_1}(r)\phi_{ap'_1l'_1}(r)\phi_{ap_2l_2}(r)\phi_{ap'_2l'_2}(r)r^2dr \tag{87}$$

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

After neglecting eigenvectors z_ν with eigenvalues $\epsilon_\nu <$ tolerance $\sim 10^{-4}$, the resulting optimal product functions are the linear combinations of the product functions as

$$b_{al\nu}(r) = \frac{1}{\sqrt{\epsilon_\nu}}\sum_{\nu'} \tilde{b}_{al\nu'}(r)z_{\nu'\nu}, \tag{88}$$

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 as shown 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 also prepare all the required radial integrations, that is `ppbrd`,

$$< \phi\phi B >= \int_0^{S_a} \phi_{ap_1l_1}(r)\phi_{ap_2l_2}(r)B_{al\nu}(r)r^2dr = \int_0^{S_a} \frac{1}{r}u_{ap_1l_1}(r)u_{ap_2l_2}(r)b_{al\nu}(r)dr, \tag{89}$$

which are stored into PPBRD*. The files PPBRD* and used in hx0fp0.m.f hxfp0.m.f through `subrouitne rdpp` (or `rdpp_v2` for fpgw025).

I Expansion of non-local functions

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

$$\begin{cases} v(\mathbf{r},\mathbf{r}') = \dfrac{1}{N_c}\displaystyle\sum_{\mathbf{k}}\sum_{IJ} \tilde{M}_I^{\mathbf{k}}(\mathbf{r})v_{IJ}(\mathbf{k})\{\tilde{M}_J^{\mathbf{k}}(\mathbf{r}')\}^* \\ v_{IJ}(\mathbf{k}) = \dfrac{1}{N_c}\displaystyle\int_V d^3r \int_V d^3r' \{M_I^{\mathbf{k}}(\mathbf{r})\}^* v(\mathbf{r},\mathbf{r}')M_J^{\mathbf{k}}(\mathbf{r}') \end{cases} \tag{90}$$

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

$$\begin{cases} D(\mathbf{r},\mathbf{r}',\omega) = \dfrac{1}{N_c}\displaystyle\sum_{\mathbf{k}}\sum_{IJ} M_I^{\mathbf{k}}(\mathbf{r})D_{IJ}(\mathbf{k},\omega)\{M_J^{\mathbf{k}}(\mathbf{r}')\}^* \\ D_{IJ}(\mathbf{k},\omega) = \dfrac{1}{N_c}\displaystyle\int_V d^3r \int_V d^3r' \{\tilde{M}_I^{\mathbf{k}}(\mathbf{r})\}^* D(\mathbf{r},\mathbf{r}',\omega)\tilde{M}_J^{\mathbf{k}}(\mathbf{r}') \end{cases} \tag{91}$$

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

$$\begin{cases} \epsilon(\mathbf{r},\mathbf{r}',\omega) = \dfrac{1}{N_c}\displaystyle\sum_{\mathbf{k}}\sum_{IJ} \tilde{M}_I^{\mathbf{k}}(\mathbf{r})\epsilon_{IJ}(\mathbf{k},\omega)\{M_J^{\mathbf{k}}(\mathbf{r}')\}^* \\ \epsilon_{IJ}(\mathbf{k},\omega) = \dfrac{1}{N_c}\displaystyle\int_V d^3r \int_V d^3r' \{M_I^{\mathbf{k}}(\mathbf{r})\}^* \epsilon(\mathbf{r},\mathbf{r}',\omega)\tilde{M}_J^{\mathbf{k}}(\mathbf{r}'). \end{cases} \tag{92}$$

J Expansion of a plane wave with the mixed basis

If we substitute a plane wave $e^{i\mathbf{k}\cdot\mathbf{r}}/\sqrt{\Omega}$ for $F^{\mathbf{k}}(\mathbf{r})$ in Eq.(82), we have

$$\begin{cases} \dfrac{1}{\sqrt{\Omega}} e^{i\mathbf{k}\cdot\mathbf{r}} = \displaystyle\sum_J M_J^{\mathbf{k}}(\mathbf{r})\tilde{C}_J^{\mathbf{k}0} \\ \tilde{C}_J^{\mathbf{k}0} = \dfrac{1}{\sqrt{\Omega}}\displaystyle\int_\Omega \{\tilde{M}_J^{\mathbf{k}}(\mathbf{r})\}^* e^{i\mathbf{k}\cdot\mathbf{r}} d^3r. \end{cases} \tag{93}$$

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.(93).

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

$$\sum_J [v_{IJ}(\mathbf{k}) - \epsilon^0(\mathbf{k}) O_{IJ}^{\mathbf{k}}]\tilde{C}_J^{\mathbf{k}0} = 0. \tag{94}$$

Then we check the normalization

$$\sum_{IJ} (\tilde{C}_I^{\mathbf{k}0})^* O_{IJ}^{\mathbf{k}} \tilde{C}_J^{\mathbf{k}0} = 1 \tag{95}$$

and calculate the two quantities

$$\texttt{v(exact)} = \Omega\frac{4\pi e^2}{|\mathbf{k}|^2}, \tag{96}$$

$$\texttt{v(cal)} = \Omega\sum_{IJ}(\tilde{C}_I^{\mathbf{k}0})^* v_{IJ}(\mathbf{k})\tilde{C}_J^{\mathbf{k}0} = \Omega\epsilon^0(\mathbf{k}), \tag{97}$$

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;

$$\begin{aligned} C_J^{\mathbf{k}0} &\equiv \frac{1}{\sqrt{\Omega}}\int_\Omega \{M_J^{\mathbf{k}}(\mathbf{r})\}^* e^{i\mathbf{k}\cdot\mathbf{r}} d^3r \\ &= \sum_I \{O_{IJ}\}^* \frac{1}{\sqrt{\Omega}}\int_\Omega \{\tilde{M}_I^{\mathbf{k}}(\mathbf{r})\}^* e^{i\mathbf{k}\cdot\mathbf{r}} d^3r \\ &= \sum_I O_{JI}\tilde{C}_I^{\mathbf{k}0}. \end{aligned} \tag{98}$$

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

K Dielectric function

K.1 Dielectric function without local-field correction

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

$$\begin{aligned}\epsilon(\mathbf{q},\omega) &= \frac{1}{V}\int_V d^3r \int_V d^3r' e^{-i\mathbf{q}\cdot\mathbf{r}}\epsilon(\mathbf{r},\mathbf{r}',\omega)e^{i\mathbf{q}\cdot\mathbf{r}'} \\ &= 1-\frac{1}{V}\int_V d^3r \int_V d^3r' \int_V d^3r'' e^{-i\mathbf{q}\cdot\mathbf{r}}e^{i\mathbf{q}\cdot\mathbf{r}'}v(\mathbf{r},\mathbf{r}'')D(\mathbf{r}'',\mathbf{r}',\omega) \\ &= 1-v(\mathbf{q})D(\mathbf{q},\omega), \end{aligned} \tag{99}$$

where the relation

$$\int_V v(\mathbf{r},\mathbf{r}'')e^{-i\mathbf{q}\cdot\mathbf{r}}d^3r = v(\mathbf{q})e^{-i\mathbf{q}\cdot\mathbf{r}''} \tag{100}$$

is used and

$$v(\mathbf{q}) = \sum_{IJ}(\tilde{C}_I^{\mathbf{q}0})^* v_{IJ}(\mathbf{q})\tilde{C}_J^{\mathbf{q}0}, \tag{101}$$

$$D(\mathbf{q},\omega) = \sum_{IJ}(C_I^{\mathbf{q}0})^* D_{IJ}(\mathbf{q},\omega)C_J^{\mathbf{q}0}. \tag{102}$$

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 $\mathrm{Re}(\epsilon)$, $\mathrm{Im}(\epsilon)$, $\mathrm{Re}(\epsilon^{-1})$ and $\mathrm{Im}(\epsilon^{-1})$ are stored in `EPS01.nolfc.dat` .

K.2 Dielectric function with local-field correction

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

$$\begin{aligned}\epsilon^{-1}(\mathbf{q},\omega) &= \frac{1}{V}\int_V d^3r \int_V d^3r' e^{-i\mathbf{q}\cdot\mathbf{r}}\epsilon^{-1}(\mathbf{r},\mathbf{r}',\omega)e^{i\mathbf{q}\cdot\mathbf{r}'} \\ &= \sum_{IJ}\left\{\frac{1}{\sqrt{\Omega}}\int_\Omega \tilde{M}_I^{\mathbf{q}}(\mathbf{r})e^{-i\mathbf{q}\cdot\mathbf{r}}d^3r\right\}\epsilon_{IJ}^{-1}(\mathbf{q},\omega)\left\{\frac{1}{\sqrt{\Omega}}\int_\Omega \{M_J^{\mathbf{q}}(\mathbf{r}')\}^* e^{i\mathbf{q}\cdot\mathbf{r}'}d^3r'\right\} \\ &= \sum_{IJ}(\tilde{C}_I^{\mathbf{q}0})^*\epsilon_{IJ}^{-1}(\mathbf{q},\omega)C_J^{\mathbf{q}0}. \end{aligned} \tag{103}$$

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 $\mathrm{Re}(\epsilon)$, $\mathrm{Im}(\epsilon)$, $\mathrm{Re}(\epsilon^{-1})$ and $\mathrm{Im}(\epsilon^{-1})$ are stored in `EPS01.dat` .