Contents

1 Introduction

"ecalj package" is for first-principle electronic structure calculations with unique features. Especially, as we explain in the next subsection, a main feature is in the quasiparticle self-consistent GW (QSGW) calculation based on the PMT method (=Linearized APW+MTO method).

1.1 Features of the ecalj package.

Generally speaking, central part in any electronic structure packages is the one-body problem solver. That is, how to calculate eigenvalues and eigenfunctions for given one-body potential. In turn, we can calculate one-body potential for given eigenfuncitons and eigenvalues based on the density functional theory (DFT) in the LDA or GGA (hearafter LDA means both of LDA and GGA). Then we can make the electron density self-consistent by iterations until converged, and obtain total energy of ground state. We can also calculate atomic forces simultaneously. Based on such a solver, we can implement higher-level approximations such as the QSGW method, where we replace the exchange-correlation potential in DFT with that given by a GW method. In addition, kinds of physical properties, linear responses and so on, can be calculated.

Such one-body problem solovers (in the case of linear methods) are characterized by (i) linear combinations of what basis set give eigenfunctions; (ii) how to represent electron density and one-body potential. In the ecalj, we uses the PMT method, which is an all-electron full potential method where we use not only the augmented plane waves (APW), but also the muffintion oribtails (MTO), in addition to the local orbital (lo). Within out knowledge no other methods allows us to use two kinds of augmented waves simultaneously. Thus eigenfuncions are represented by linear combinaitons of the APWs, MTOs, and the lo's. Then the electron density and the one-body potential are represented by "smooth part + onisite muffin-tin (MT) part - counter parts to remove smooth part within MTs. This part (Soler-Williams formalism) is essentially the same as that of the projected augmented wave method.

With ecalj, we can perform the GW calculation. The usual GW approximation is to obtain quasiparticle energies (QPE) by one-shot calculation starting from LDA; this is so-called "one-shot GW". Its abilitiy is limited; it can fail when its starting point (eigenfunctions and eigenvalues supplied by LDA) is problematic. Thus T.Kotani with collabolators have developed the QSGW method. The QSGW now becomes popular, performed by other reserchers. In principle, results by QSGW do not depent on LDA anymore; the LDA are only used to prepare initial condition for self-consistency cycle of the QSGW calculation (however, exactly speaking, we use LDA to assist efficient implementation of QSGW, e.g. to prepare required radial functions).

Usually the QPEs obtained by QSGW reproduce experiments better than LDA. For example, the band gap by GGA for GaAs is about 0.5 eV in contrast to the experimental value of 1.6eV (If we virtually remove electron-phonon effect from the experimental value, it becomes about 1.7eV). On the other hand, the

QSGW predict about 1.9eV, slightly larger than experiment (for practical use, we sometimes add "scaling correction" on it to have better agreements with experiments). Even in the case of NiO and so on, QSGW gives not so bad results (it tends to give a little larger band gaps than experiments).

From QPEs and eigenfunctions obtained in QSGW, we can calculate dielectric functions and so on. But total energy in QSGW is still in research.

The ecalj package can also do other functions, LDA+U, atomic force and relaxiation (in GGA/LDA), spin fluctuation, optical response and so on.

Recent developement of dual-based MTO prescription allows us to use very localized MTOs (with damping factors $\exp(-r/(1\text{a.u.}))$), together with APWs of low energy cutoff ($\sim$ 3 Ry). I think this is promising not only for efficient DFT/QSGW scheme, but also for kinds of applications in future. The MTOs can be used instead of the Wannier functions, but not so much research on it yet.

The QSGW calculation requies so much computational time: roughly speaking, it takes 10 or more times expensive than usual one-shot GW (but you can reduce computational time by choosing computational conditions). Thus the size of systems we can treat is about ten atom in a cell; computation requires about a week or so to have reasonable convergence. (heavy atoms require longer computational efforts, light atoms faster; we still have room to acceralate the method, but not yet so much. Minimum MPI parallization is implemented). The computational effort is $\propto N^4$.

The ecalj web site is at

`https://github.com/tkotani/ecalj`

We can download ecalj package from this site. Free to download, modify, and use it, but we need you to clarify acknowledgement to the package in your publications; without it, we can not continue further developments. We also have a web site at http://pmt.sakura.ne.jp/, however, most of all are yet in japanese and not yet well organaized. The ecalj is related to lmv7 package at `http://titus.phy.qub.ac.uk/packages/LMTO/fp.html`. The lmv7 and ecalj are branched off at year 2009. After branched, major contributions are due to Takao Kotani and Hiori Kino (NIMS). All codes are now in f90, and we have simpified complications in the lmf7 code.

A manual on lmv7 may be a help to learn ecalj. But it may make you confused, becasue we have removed confusing parts from it, and simplified. Read this first. Then I show where we should read in the lmv7 manual.

If you have something, let takaokotani@gmail.com know it. With your ideas, we like to have collabolations and newer development on it. Or I may help something on it.

1.2 What is in this booklet?

Here we can show minimum on the ecalj package.

- How to perform self-consistent calculations by the density functional theory (DF) in the LDA
- How to perform the QSGW calculations.
- How to plot energy bands (BAND), total density of states (DOS), and the partial density of states (PDOS).
- How to plot dielectric functions, and non-interacting spin susceptibility χ_0^{+-}. (not documented yet...)

To perform calculations becomes easier recently. After we prepare a crystal structure file named as `ctrls.*`, we run a a script (`ctrlgenM1.py`) to generate `ctrl.*` Its contains (not so bad) default setting to do calculations. To help writing `ctrls.*`, ecalj contains samples and a converter between VASP-POSCAR format and ecalj-ctrls format. (Thus we can use tools for VASP).

However, we may need to know a littlle more to obtain results for publications; some knowledge to judge whether obtained results are reasonable or not. And check convergence by some different conditions. Especially, results by QSGW need to be examined carefully. But, an advantage is that the QSGW code is version controlled and hosted by github, your results can be reproduced by others.

2 Install

Look into ecalj/README. (or you can see it at https://github.com/tkotani/ecalj). The installation is not do difficult (especially for gfortran, and ifort). After finished, we have all required binaries and shell scripts in your /bin. (or someswhere else where you specified at the bottom of Makefile.)

3 LDA calculations

Calculations are performed by following steps.

- Write ctrls file,"crystal structure file", which contains crystal structure. It can be by hand, or convert it from POSCAR (in vasp). There is a tool to convert between POSCAR and ctrls. (ecalj/StructureTool/README). There is a checker, lmchk, to confirm the crystal structure is read well or not.
- Generate ctrl from ctrls by a script `ctrlgenM1.py`. `ctrl` is the main control file which contains all required information for calculations; as a part it contains ctrls. Then we we edit the generated ctrl file for your purpose.
- Do lmfa (just calculate atoms (MT sites) placed in the cell). It also calculate core eigenfunctions and valence charge for initial condition. And

start main calculation lmf successively. Then we get self-consistent result of LDA.

- Plot energy band, DOS, PDOS, by runnning scripts.

These `ctrls` and `ctrl` are with extensions as

```
ctrl.(id)
ctrls.(id)
```

. `(id)` is extention which we can add (only lower letter allowed). For example, `ctrls.cu`, `ctrls.lagao3`.

3.1 Write crystal structure file, ctrls

We have a sample of Cu in `~/ecalj/lm7K/TESTsamples/Cu/ctrls.cu`; it is

```
% const da=0 alat=6.798
STRUC  ALAT={alat} DALAT={da}
       PLAT=  0.0 0.5 0.5  0.5 0.0 0.5   0.5 0.5 0.0
SITE    ATOM=Cu POS=0 0 0
```

Another sample at `~/ecalj/lm7K/TESTsamples/GaAs/ctrl.gaas` is

```
#id  = GaAs
%const bohr=0.529177 a=5.65325/bohr
STRUC
     ALAT={a}
     PLAT=0 0.5 0.5  0.5 0 0.5  0.5 0.5 0
SITE
     ATOM=Ga POS=0.0 0.0 0.0
     ATOM=As POS=0.25 0.25 0.25
```

Lines starting from '#' are neglected as comment lines. Lines statring from `% const` define variables and set values (in these cases, `da` and `alat bohr`). Then the variable are used as `{alat}`; in the cu case, `{alat}` means 6.798. Lines not start from "#" nor "%" are main content in the ctrls file.

Note that we have two tags of "categories" "STRUC" and "SITE". These tags should start from the first column. Thus ctrls is divided by multiple "categories". In a category, we have "tokens" such as ALAT, DLAT, PLAT. These under STRUC category. ALAT+DALAT specify a unit to mesure length in this ctrl file. These are in a.u. (= bohr radius=0.529177Å). We use ALAT+DALAT as unit to specify primitive vectors to specify unit cell. The unit cell is given by PLAT (ALAT as unit). In the above example, three primitive cell vectors specified by nine numbers after PLAT=; they give three primitive vectors; PLAT1=(0,0,0.5), PLAT2=(0.5, 0.0, 0.5), and PLAT3=(0.5, 0.5, 0). DALAT is convenient to change lattice constant; but it is fixed to be zero here; thus no effect in this example.

Note that SITE category can have multiple ATOM tokens. The number of ATOM token under SITE should be the same as atoms in the primitive cell. In the case of GaAs; SITE contain multiple ATOM tokens. POS just next to ATOM is taken as subtokens under ATOM token. (this may looks sligtly uncompfortable since the end of range of ATOM token is not so clear; but it is not a probelm). In cases, we specify such subtokens as SITE_ATOM_POS.

In the SITE caterogy, we place atoms (MT names) in the primitive cell. In these cases we use defaults atomic symbol (MT names) for `ATOM`. POS is in the Cartesian coodinate, but in the unit of ALAT+DALAT.

For your test, you may make a test directory and copy a ctrls.* to your directory. If you have VESTA and ecalj/StructureTool installed, you can see its structure by

```
$ viewvesta ctrls.cu
```

(here $ means command prompt).

> NOTE: As written in ecalj/README, you have to install VESTA and viewvesta. Then set VESTA= at the top of ecalj/Structure/viewvesta, and make softlink to it. The command `viewvesta(~/ecalj/StructureTool/viewvesta.py)` generate `POSCAR_cu.vasp` first, then send it to VESTA. `viewvesta` also accept `POSCAR_cu.vasp` directly. Except names starting from `ctrl` and `ctrls`, `viewvesta` sends the name to VESTA directly. VESTA requires extension '.vasp' to identify VASP format. We have samples in `~/ecalj/StructureTool/sample`. A tool `vasp2ctrl` converts POSCAR_.vasp to ctrls.. "–help" show a small help.
> • ecalj/StructureTool/ is not tested well. Not believe it so much... We will fix it on your request.

Another exmaple with atoms is `~/ecalj/lm7K/TESTsamples/SrTiO3/ctrls.srtio3`. This contains `SITE` category as

```
SITE
  ATOM=Sr POS=1/2 1/2 1/2
  ATOM=Ti POS= 0   0   0
  ATOM=O  POS=1/2  0   0
  ATOM=O  POS= 0  1/2  0
  ATOM=O  POS= 0   0  1/2
```

. Note that an expression 1/2 can be used for POS. As it show, we can use mathematical expression instead of values. Mathematical expressions such as "$+ - */$ sqrt(...)" are recognized. (instead of $3 * *2$, use 3^2. You can use parenthese, but no space for separation). (After ctrl generated, these is a command lmchk to check wether atomic structure is correctly given or not.) We can use default atomic symbols (to check default atom name (MT name) type `ctrlgen2.py --showatomlist`). Instead of such default symbols, we can use your own symbol as

```
SITE
  ATOM=M1 POS=1/2 1/2 1/2
  ATOM=M2 POS= 0   0   0
  ATOM=O  POS=1/2  0   0
  ATOM=O  POS= 0  1/2  0
  ATOM=O  POS= 0   0  1/2
SPEC
  ATOM=M1 Z=38
  ATOM=M2 Z=22
  ATOM=O  Z=8
```

. Then we have to add extra category SPEC where we set Z number. (You can use Z=37.5 for virtual crystal approximation, however, you can not do it in ctrls. Edit it in ctrl file.)

We can see other samples in `~/ecalj/lm7K/TESTsamples/*/ctrls.*`. (we also have a sample generator. See later section.) Note that ctrls file is jut in order to generate default ctrl file in the followings. Not from ctrls but from ctrl, we can start calculations. Thus ctrls is not necessary if we prepare ctrl file directry.

3.2 Generate default ctrl from ctrls by ctrlgenM1.py

To run programs of lm7K (lmfa and lmf) in PMT, we need an input file ctrl which contains many other settings. To generate ctrl from ctrls, we have a command "ctrlgenM1.py" (written in python 2.x and call fortran code internally). Two steps required to complete ctrl file: (i) we give reasonable options to invoke ctrlgenM1.py. This generate a ctrl file.; (ii)we need to edit the ctrl file afterwards for your calculation.

At first, try `ctrlgenM1.py` without arguments. It shows help. To generate `ctrl` from `ctrl`, type

```
$ ctrlgenM1.py cu --nk1=8
```

Here cu specify ctrls.cu. The option –nk1=8 means the number of division of the Brillowin zone for integration. It means 8x8x8 division. If we like to use 8x8x4, we have to supply three arguments –nk1=8 –nk2=8 –nk3=4. The above command gives following console output.

```
$ ctrlgenM1.py cu --nk1=8
 === INPUT arguments (--help gives default values) ===
  --help  Not exist
  --showatomlist  Not exist
  --nspin=1
  --nk=8
  --xcfun=vwn   !(bh,vwn,pbe) 1
  --systype=bulk !(bulk,molecule)
```

```
  --insulator  Not exist !(do not set for --systype=molecule)

...

OK! A template of ctrl file, ctrlgen2.ctrl.cu, is generated.
```

As we see above, options which you speified are shown at the beginig of the cosole output (in this case –nk1=8). Others such as –nspin=1 are default settings. If we like to perform spin-polarized calculations, we add other option '–nspin=2' as

```
ctrlgenM1.py cu --nspin=2 --nk1=8
```

(NOTE:In the spin-polarized case, we need to set initial condition of size of magnetic moment at each atoms: we have to edit SPEC_ATOM_MMOM of ctrl file (`MMOM=s p d f ...`) to be like `MMOM=0 0 2`. We will explain this later on). The `ctrlgenM1.py` generates ctrl file named as `ctrlgenM1.ctrl.cu`. To do calculations, copy it to ctrl.cu so that lmf recognize it.

```
cp ctrlgenM1.ctrl.cu ctrl.cu
```

3.3 crystal structure checker: lmchk

Do lmchk to confirm correct crystal structure is really given or not.

```
lmchk --pr60 cu
```

Then it reads `ctrl.cu`. `--pr60` is an option of verbose. Bigger number gives more informations.

- Lattice info, Space group symmetry operations (in lmf format), and their generators (these operations can be generated from a few of them.) See `http://titus.phy.qub.ac.uk/packages/LMTO/tokens.html#SYMGRPcat` about how to represent the operations.
- Show atomic positions. If rst.* (this is generated after DFT calculation and relaxiation) exists, it may show a position in rst.* file (relaxed position). NEED to CHECK.
- Tabulate MT radius and distance between atomic sites.

3.4 ctrl file

The ctrl file generated by the `ctrlgenM1.py` contains explanations. Read it first. This is a head part of `ctrl.cu` generated by ctrlgenM1.py:

```
### This is generated by ctrlgenM1.py from ctrls
### For tokens, See http://titus.phy.qub.ac.uk/packages/LMTO/tokens.html.
### Do lmf --input to see all effective category and token ###
### It will be not so difficult to edit ctrlge.py for your purpose ###
```

```
VERS    LM=7 FP=7        # version check. Fixed.
IO      SHOW=T VERBOS=35 TIM=2,2
             # SHOW=T shows readin data (and default setting at the begining of
console output)
             # It is useful to check ctrl is read in correctly or not
               (equivalent with --show option).
             # larger VERBOSE gives more detailed console output.
SYMGRP find  # 'find' evaluate space-group symmetry automatically.
             # Usually 'find is OK', but lmf may use lower symmetry
...
```

`#` means comment lines. We can also use lines start from `% const ...` to define variables and set constant.

We see "categories" such as `VERS` and `IO`, and so on. The begining of categories are starting from the first column. Under categories, we have "tokens" such as `VERBOSE`. Thus we specify full name of token `VEROSE` under category `IO` as `IO_VERBOSE`.

The ctrl file generated by ctrlgenM1.py contains explanations. Thus read them first. Here we give some comlementary explanations to it.

- `IO_TIM` is for debugging. Itshows which subroutines are called and so on. Bigger number shows deeper subroutines.
- `SYMGRP` is a category without token under it; we set generators of space group (See explanation in previous paragraph). When we set `find`, it automatically calculate symmetry of crystal lattice. If we like to enforce symmetry, set some of generators which are shown by lmchk.
- We see `ctrls` is embedded in the `ctrl` by `ctrlgenM1.py`.

  ```
  ... (skip) ...
  % const  da=0 alat=6.798
  STRUC   ALAT={alat} DALAT={da}
          PLAT=  0.0 0.5 0.5  0.5 0.0 0.5   0.5 0.5 0.0
          NL=4 NBAS= 1  NSPEC=1
  SITE    ATOM=Cu POS=0 0 0
  ... (skip) ...
  ```

  NL, NBAS(number of SITE) and NSPEC(number of SPEC) are automatically added by ctrlgenM1.py. It is possible to deform unit cell by adding optional tokens (it is possible to rotate PLAT for magnetic anisotoropy calculation). See `http://titus.phy.qub.ac.uk/packages/LMTO/tokens.html#STRUCcat`. For new calculations, it is better to find some examples first.

- **SITE** category: As for MT sites, we have two categories. (1)SPEC(species) and (2)SITE(specify centers of atoms(species) in primitive cell) As for SPEC, we specify MTs(radius, Z, MTOs on it) appeared in the cell. These are defined subtokens under SPEC_ATOM=foobar (we have multimple

SPEC_ATOM=foobar). Then we place these SPEC_ATOM=foobar at SITE sections.

At SITE, we specify atomic sites (What SPEC_ATOM is placed to positions by POS) in a primitive cell. We set `POS=` by direct form (cartesian) but with the unit of `ALAT+DLAT`. Total number of SITE (number of tokens SITE_ATOM) is the number of atoms in the primitive cell. SITE_ATOM=foobar means we place foobar defined in SPEC_ATOM=foobar. We set subtoken SITE_ATOM_POS under SITE_ATOM, to specify atomic positions. In addition, we can set SITE_ATOM_RELAX, if you like to find relaxed structure (we simultaneously set DYN category) in LDA. As for relaxiation, see `LaGaO_relax/ctrl.lagao` example, and read `http://titus.phy.qub.ac.uk/packages/LMTO/tokens.html#DYNcat`.

The SITE_ATOM=foobar (with same foobar with different POS) are not necessarily equivalent with respect to the space group operation of a system. Thus `SITE_ATOM=foobar` are divided into "classes" which are connected by the operation. The lmf automatically judge "classes" (see also infor by lmchk). Thus not need to specify it, but it may be better to check it. A sample is `lmchk lagao` at `~/ecalj/lm7K/TESTsamples/LaGaO_relax`

- **SPEC** category: In ctrls, we have not yet specified contents of SPEC; we have just given default symbols or only Z= when we use non-default names (shown by ctrlgenM1.py –showatomlist). The command `ctrlgenM1.py` adds default SPEC sections.

  We have some `SPEC_ATOM`, under which we give subtokens such as `SPEC_ATOM_R`(MT radius), `SPEC_ATOM_Z`(nucleus charge), cutoff parameters of angular momentum, and so on. These `SPEC_ATOM` is reffered to in SITE.

  An example of SPEC category is

```
SPEC
    ATOM=Fe Z=26 R=1.70
      KMXA={kmxa}  LMX=3 LMXA=4 NMCORE=1
      PZ=0,3.9,4.5
      EH=-1 -1 -1 -1  RSMH=0.85 0.85 0.85 0.85
      EH2=-2 -2 -2   RSMH2=0.85 0.85 0.85
      MMOM=0 0 2 0

    ATOM=... (then the similar block of ATOM= are repeated.)
       ...
```

  Under the token `ATOM=Fe`, we have subtokens `SPEC_ATOM_Z`,`SPEC_ATOM_R`, and so on.

  Subtokens Z= is the nucleus charge and R= MT radius. Note that Fe is just a name to distinguish MT sphere in the cell. If you set SPEC_ATOM_Z=27, it is recognazed as Co (since Z=27). `LMX=3` is the maximum l of MTOs.

Thus maximum l of MTO is l=3. The maximum of l to expand electron density and potential within MT is LMXA (in contrast to usual LAPW), we can use quite small LMXA such as LMXA=4. NMCORE=1 means we calculate core density without non magnetically-polarization. This can reduce comuputational confusion.

PZ is to set local orbital (if not, no local orbitals). EH and RSMH are to spefify first set of MTOs.(We can check how local orbitals are set by lmfa explained in the next section). EH2 and RSMH2 are to spefify second set of MTOs.

After PZ=, we have three numbers. These are numbers for s,p,d,f,g,... channels. Zero means not exist. You can use space or , as delimiter. Here not only the integer part of principle quantum number, but also the fractional part should be supplied (If PZ=0,3,4, it does not work.) Now PZ=3.9 for p and PZ=4.5 for d. This means we use local orbital for 3p, and local orbital for 4d (fractional parts (continious principle quantum number) are large $\sim$ 0.9 for core like orbital, and smaller for extended orbital $\sim$ 0.3 or something. See Logarithmic Derivative Parameters at `http://titus.phy.qub.ac.uk/packages/LMTO/lmto.html`). This is a little confusing, thus we will explain this in appendix. See Sec.**??**.

EH(damping factor), and RSMH (where the smooth Hankel function bent) determines MTOs. We now set four numbers for them. Thus we set MTOs s,p,d,f with EH=-1 and RSMH=0.85. Our current test shows that RSMH is one half of R (that is, 0.85=1.70/2, but minimum RSMH is 0.5) and not need to be dependent on s,p,d,f. (If LMX=2, s,p,d are allowed and no f MTOs.) EH is -1; not need to change except test purpose. In a similar manner, EH2 and RSMH2 for second set of MTOs are given. Just three numbers means these for s,p,d.

MMOM=s,p,d,f... gives initial condition of magnetic moment in μ_B (number of up-down electron).

In cases such as As, the local orbital given by default ctrl is responsible of rather deep core, and it is not need to be treated as valence electrons. In such a case, we don't need local orbital.

In the case of AntiFerro-II NiO, it contains two NiO in a primitive cell. Thus it is reasonable to have place two SPEC_ATOM as Ni1 and Ni2, although subtokens under ATOM=Ni1 and ATOM=Ni2 (e.g. `SPEC_ATOM_EH` for them) are the same except initial contition of magnetic moment of MMOM=s,p,d,f... See example of NiO.

The minimum help of call Category_token_subtoken are listed with minimum explanation with

```
$ lmf --input
```

It gives a long output. But many of them are experimental and not need to touch them. A part of it is

```
Token            Input   cast  (size,min)
-----------------------------------------
   ... ...

    STRUC_ALAT        reqd   r8       1,  1
      Scaling of lattice vectors, in a.u.
   ... ..
```

This is an minimu explanation of it. "reqd" means "required" (no default). r8 means it read with real number, 1,1 means that ALAT=xxx should contain one number minimum (max is also one) (See also STRUC_PLAT, and so on).

There are kinds of examples in ecalj packages. Please look into their of ctrls.* and ctrl.* These are in lm7K/TESTsample/* and ecalj/CMDsampls. In addition, ecalj/MATERIALS contain many samples; see a later subsection.

In anyway, we think it is very important to add examples which shows all the functions of ecalj package (not only materials but also to show its functions). It is not yet, but we will do in on your request.

As for what is shown in `$ lmf --input`, most of important tokens are already described in the ctrl file generated by ctrlgenM1.py. So, we don't need to care many options shown by it. But we have not explained "STRUC category to defrom crystal, DYN category for dynamics, LDA+U treatment, Adding background charge, Core-Hole treatments" here. We will prepare examples for it if requested.

`http://titus.phy.qub.ac.uk/packages/LMTO/tokens.html#STRUCcat`
`http://titus.phy.qub.ac.uk/packages/LMTO/tokens.html#DYNcat`

3.5 Do LDA/GGA calculations, and get convergence

Here we show how to get converged results from a `ctrl` file.

At first, we need initial guess of charge density. It can be given by a super position of atomic charge density. To obtain the charge density, we solve atoms first. It is by

```
$ lmfa gaas | tee llmfa
```

It takes just a few seconds. Here tee is a command of linux. It divide console output (standard output) to a file, and to console.

Then try

```
 $ grep conf llmfa
```

. Then you see a key point that

```
conf:SPEC_ATOM= Ga : --- Table for atomic configuration ---
conf:  isp  l  int(P) int(P)z    Qval     Qcore   CoreConf
conf:    1  0       4  0         2.000    6.000 => 1,2,3,
conf:    1  1       4  0         1.000   12.000 => 2,3,
conf:    1  2       4  3        10.000    0.000 =>
```

```
conf:    1  3       4  0         0.000    0.000 =>
conf:    1  4       5  0         0.000    0.000 =>
conf:-----------------------------------------------------
conf:SPEC_ATOM= As : --- Table for atomic configuration ---
conf:  isp  l  int(P) int(P)z    Qval     Qcore   CoreConf
conf:    1  0       4  0         2.000    6.000 => 1,2,3,
conf:    1  1       4  0         3.000   12.000 => 2,3,
conf:    1  2       4  3        10.000    0.000 =>
conf:    1  3       4  0         0.000    0.000 =>
conf:    1  4       5  0         0.000    0.000 =>
conf:-----------------------------------------------------
```

This is an initial electron distribution, and how we divide core and valence. In this case core charege Qcore are (6 electron for s channel=1s,2s,3s and 12 electron for 2s and 3p). Core is not treated separately from valence electrons (frozen core approximation; we superpose rigid core density to make all-electron density). Qval means electrons for each s,p,d channels. The valence channels are 4s,4p,4d,4f (when we set EH=s,p,d,f). The int(P)z columun is for local orbital. Thus we have 3d treated as local orbital. (our code can add one local orbital per l)

isp means spin (1 or 2), since –nspin=1 for Ga and As, no isp=2 exist. In summay we have 4s,4p,4d,3d,4f as valence. This means we use correspoing number of MTOs and local orbitals.

After lmfa, let us start main calculation.

```
$ lmf cu
```

It might be better to do `$ lmf cu | tee llmf` in order to keep cosole output to llmf. As it is a iteration calculations, it shows similar output again and again. Then you end up with self-consistent result as

```
    ......
   it  8  of 30    ehf=   -3304.895853   ehk=   -3304.895853
 From last iter    ehf=   -3304.895856   ehk=   -3304.895855
 diffe(q)=  0.000003 (0.000007)    tol= 0.000010 (0.000010)   more=F
c ehf=-3304.8958531 ehk=-3304.8958529
 Exit 0 LMF
 CPU time:    7.024s     Mon Aug 19 02:03:19 2013   on
```

`it  8  of 30` meand it stop at 8th iteration, althogh we set maximum number of iteration 30. Note that this number is given by `ITER_NIT=30` in `ctrl.cu`). `ehf` and `ehk` are the ground state energy in Ry. They are calculated in a littel different manner. Although they are different during iterations, it finally get to be the almost the same number.(but it can be different like 10 micro Ry per atom even converged. But you don't need to care it so much). ehk:Hohenberg-Kohn energy, ehf: Harris-foulkner energy.

grep diffe lllmf shows how the changed of total energy (and charges) during iteration. diffe(q) mean changes of energy with previous iteration、q is for

electron density difference as well. See also save.* file, which only show ehk and ehk.

Thus we do have ground state energy. Although output of lmf is long, most of all are to monitor convergence. As long as it converged well, you don't need to look into it in detail. Eigenvalus are shown as

```
bndfp:  kpt 1 of 4, k=  0.00000  0.00000  0.00000   ndimh = 122
-1.2755 -1.2008 -1.2008 -0.2052 -0.2052 -0.2052 -0.0766 -0.0766 -0.0766
-0.0174 -0.0174 -0.0174  0.1094  0.1095  0.1095  0.2864  0.2864  0.4170
 0.4170  0.4736  0.6445  0.6445  0.6445
```

This is at k= 0.00000 0.00000 0.00000 . (because of historical reason, two same bndfp: are shown in each iteration; two band path method). At, "lmf cu—grep -A6 BZWTS" shows the fermi energy (for insulator, we see band gap). Deep levels which gives little dependence on k are core like levels.

rst.* contains is the main output which contains electron density. mix.* is a mixing file (which keeps iteration history). When you restart lmf again, it read rst.cu and mix.cu. If you start from lmfa result, please remove them. (or lmf –help show option of –rs=(five numbers), how to read atm file which is the intial atom file by lmfa). We can do pararell calculation with lmf-MPIK, we can invoke it with mpirun -np 8 lmf-MPIK cu. It should give the same answer.

3.6 DOS, Band, PDOS plot

We already have script to plot dos and band from the result of lmf self-consistent calculations. At `~/ecalj/lm7K/TESTsmaples/*`, we have

```
job_tdos, job_band_nspin1, job_band_nspin2
```

. Read this script, and then you see how to plot enery bands.

For total DOS plot, it is better to check ctrl file; `BZ_TETRA=1`(this is defalult; thus make sure that `BZ_TETRA` do not exist or `BZ_TETRA=1`). In addition, we have to enlarge number of k point `NKABC` large enough. Then we do

```
job_tdos cu
```

This shows total DOS as

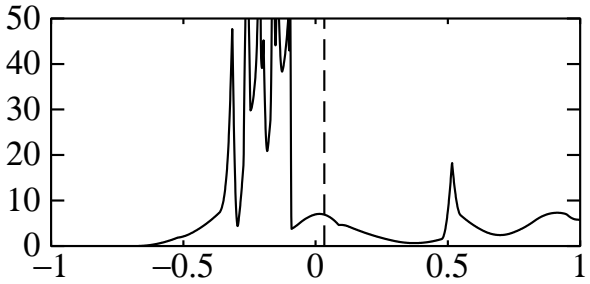

Figure 1: DOS(Cu)

For band plot, we have to set symmetry lines along which we plot eigenvalues. It is given in `syml`. `syml` also must have extension as `syml.cu`. Collections are in ecalj/MATERIALS/. Choose and modify one of them and rename it.

Here let us a sample in syml.cu.

```
$ cp ~/ecalj/lm7K/TESTsamples/Cu/syml.cu .
```

Thus we have `syml.cu` to your directory.

Look into `syml.cu`; it is

```
21  .5 .5 .5     0  0 0          L to Gamma
21   0  0  0     1  0 0         Gamma to X
0    0 0 0  0 0 0
```

First line means, we calculate eigenvalues for **k** points from **k**=(0.5,0.5,0.5) to **k**=(0,0,0). "L to Gamma" is just a comment since program only read seven numbers for each line. Second line means, we calculate eigenvalues for k points from **k**=(0,0,0) to **k**=(1,0,0). 3rd line means calcultion just stop here. Units of **k** are in 2π`ALAT`.

A line starting from '#' is neglected (comment line).

Do

```
job_band_nspin1 cu
```

is for spin=1. `job_band_nspin2` is for spin=2 when it exist. (These scripts try to determine the Fermi energy first. You may skip it in cases.)

For PDOS plot,

```
job_pdos cu
```

It shows many figures in gnuplot. It is a little expensive because we use no symmetry to distinguish all lm channel in a simple manner. (PDOS is not yet implemented for SO=1 case; spin-orbit coupling $\mathbf{L}\dot{S}$ is added.) We have to re-organize script of gnuplot (pdos.site*.*.glt) for your purpose. In priniciple, meanings of all data files are shown (see at the bottom of console output about lm ordering in a line), thus not so difficult to rewrite *.glt. For example, to plot eg and t2g separaely. (NOTE: site id is shown by lmchk).

WARNING: Usually lmf and so on recognize options such as -vnspin=2 or something, where nspin is defined as `% const nspin=1` in ctrl file. The option overlaid `% const nspin=1`; replace it with =2; (this is shown as save.* file, and top of console output). However, job_tdos and so on, do not yet accept these options. Thus you may need to modify these command (we will fix it in future.)

3.7 Samples: ecalj/MATERIAL/

We have a material database in `ecalj/MATERIALS/`. At the directory, type

```
$ ./job_materials.py
```

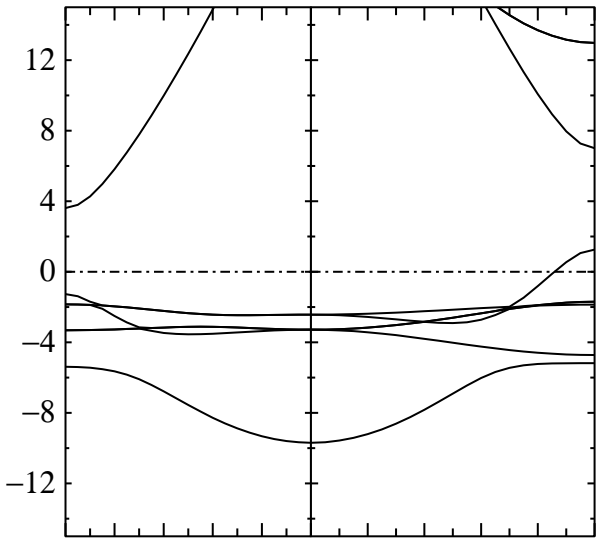

Figure 2: band plot(Cu)

Then it shows a help. You see

```
...
=== Materials in Materials.ctrls.database ===
  2hSiC  3cSiC  4hSiC  AlAs  AlN  AlNzb  AlP  AlSb  Bi2Te3  C
  CdO  CdS  CdSe  CdTe  Ce  Cu  Fe  GaAs  GaAs_so  GaN
  GaNzb  GaP  GaSb  Ge  HfO2  HgO  HgS  HgSe  HgTe  InAs
  InN  InNzb  InP  InSb  LaGaO3  Li  MgO  MgS  MgSe  MgTe
  Ni  NiO  PbS  PbTe  Si  SiO2c  Sn  SrTiO3  SrVO3  YMn2
...
```

. For these simple materials, input files will be generated. The ctrls are stored in `ecalj/MATERIALS/Materials.ctrls.database` (in addition, options passed to ctrlgenM1.py and options to lmf-MPIK are included). The command `./job_materials.py` gives ctrls.* for these materials based on the description in the `Materials.ctrls.database`. And then it generates ctrl file by calling ctrlgenM1.py internally, and run lmfa lmf-MPIK successively (when no –noexec).

For example, try `./job_materials.py Fe --noexec`. (not fe but Fe as it shown above). Then it makes a directory Fe/ and set ctrl.fe (also ctrls.fe) in the directory. Without '–noexec', it does calculation for Fe successively. As for NiO and Fe, we see that `./job_materials.py` gives SPEC_ATOM_MMOM in generated ctrl file.

Try `job_materials.py GaAs Si`.
Then directories GaAs/ and Si/ are generated. See `save.*` files containing total energies iteration by iteration. Starting from `ctrl.*` in these directory, the command perform DFT calculations (Console output is stored in `llmf`, `save.*` gives total energies. `rst.*` contains self-consistent density, from which we can calculate energy bands and so on).

"`./job_materials --all --noexec`" generates ctrls and ctrl files of these materials. "`./job_materials --all`" do self-consistent LDA calculations for materials (it takes an hour or more. Change the number of cores for MPIK in the script(search lmfjob) if you like).

To make band plot and so on for Fe,

```
  $ ./job_materials.py Fe  (and need to type return)
  (If you like start over, remove Fe/ under it first).
  $ cd Fe
  $ ./job_materials.py fe
    (but it might be better to do --noexec, and observe Fe/ctrls.fe and
Fe/ctrl.fe first. grep conf llmf shows the initial electron distribution).
  $ cat save.fe  (this shows total energies of each iteration. 'c ' at
the first column gives converged result. 'h ' is from atm file.)
        If it does not ends with 'c ...' line, something strange
occurs. see llmf (console out put of lmf is saved to llmf).
  $ cp ../syml.fe .
  $ job_band_nspin2 fe
        (As I said, this shell script do not yet accept
    options to lmf. Look into the script).
        (This calculate fermi energy first for safe; it takes
    some time)
  $ job_tdos fe
  $ job_pdos fe (as I said, this supress space-group symmetry, thus time consuming).
```

Then it shows a help. See joblmf file also (it contains options to invoke lmf. This is shown in save.*. In principle, options in joblmf should be passed to band plot and so on. But not yet implemented (it is not so diffficult).

4 QSGW calculation

The QSGW calculation requires to calculate "non-local exchange-correlation potential $\Sigma_{\rm xc}$" (it may be better to say static version of self-energy), which is calculated by GW calculation. Then $\Sigma_{\rm xc} - V_{\rm xc}^{\rm LDA}$ is stored into `sigm` file. Then, we calculate one-body calculation by lmf (or lmf-MPIK) where we add sigm to one-body potential. This iteration cycle is performed by a script "gwsc" as we explain later on.

4.1 GWinput

Let us start from ctrls.si as

In order to perform QSGW, one another input file `GWinput` is necessary in addition to `ctrl`. Thus all input files for QSGW is just two files, ctrl.* and GWinput. A templete `GWinput` can be generated by a script `mkGWIN_lmf2`. You may have to modify it in cases for your purpose.

Let us start from ctrls.si;

```
#id  = Si
%const bohr=0.529177 a= 5.43095/bohr
STRUC
     ALAT={a}
     PLAT=0 0.5 0.5  0.5 0 0.5  0.5 0.5 0
SITE
     ATOM=Si POS=0.0 0.0 0.0
     ATOM=Si POS=0.25 0.25 0.25
```

. Do "ctrlgenM1.py si –tratio=1.0"and rename ctrlgenM1.ctrl.si to ctrl.si. (–tratio=1.0 means touching MT (check it by lmchk); it is probably slightly advantagenous in our version of GW calculation). Let us invoke `mkGWIN_lmf2` as follows.

```
$ lmfa si (lmfa is needed to do in advance).
$ mkGWIN_lmf2 si
......
== Type three integers n1 n2 n3 for Brillowin Zone meshing for GW! ==
 n1=
```

Then it pause and ask numbers. You have to type three numbers as 2+ return + 2+return+2 return.

```
== Type three integers n1 n2 n3 for Brillowin Zone meshing for GW! ==
 n1= 2
 n2= 2
 n3= 2
2 2 2
```

```
...(skip)...
OK! GWinput.tmp is generated!
```

Generated file is `GWinput.tmp`; you have to copy it to `GWinput`.

```
$ cp GWinput.tmp GWinput
```

These '2 2 2' you typed is reflected in a section 'n1n2n3 2 2 2 ' in `GWinput`. You can edit it, and change it to e.g. 'n1n2n3 4 4 4' if you like to calculate self-energy on dense BZ mesh. (QPNT.chk contains irreducible k point for given n1 n2 n3; KPTKPTin1BZ.gwinit.chk contain all k points in Brillowin Zone). Generally speaking, you don't need to repeat mkGWIN_lmf2 as long as you don't change MTO sections in ctrl file (number of EH,EH2,PZ).

Look into `GWinput`. Because of historical reason, input system is different from ctrl. The main input files is GWinput. [it is a unified file of old GWIN0, GWIN_V2, and QPNT]. This controls the setting of GWcalculation. 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-04 ! =tolopt
 lcutmx(atom)
  3 3
  atom   l
...
</PRODUCT_BASIS>
```

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

In the QSGW calculation, we have to set some cutoff parameters for self-energy calculations.

The readin routine for GWinput is `getkeyvalue` defined in `gwsrc/keyvalue.f` 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 (this is not used for gwsc)
4.`<QforEPS>`,`<QforEPSL>` section
We will explain each by each in the followings.

- `KeepEigen, KeepPPOVL`: defaults are off. 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 or large systems, use them off. Then you can save memory usage. We need to examine it a little more...

- +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).

- `emax_sigm` is the maximum limit of the self-energy (measured from the Fermi energy). I think 2.5 $\sim$ 5Ry is reasonable choice. Generally speaking, larger is better but expensive. In addition, we can not expect accuracy for high energy bands, thus large `emax_sigm` may give wrong results; we may just change `emax_sigm` and check stability. As `emax_sigm` gives a sudden energy cutoff, energy band dispersion may give unnatural behevior (then let me know). Generally speaking, accuracy less than $\sim$0.1eV (as for bandgap) is allowance of current implementation. Probably, it may be not impossible to have better accuracy, but it may ask us to repeat many calculations with changing conditions to confirm stability.

- `0.100000D-02 ! =tolopt` controls a number of Product basis to expand the Coulomb interaction. (The product basis is to expand the Coulomb interaction is different from the basis to expand eigenfunction.) In our experience, `0.100000D-02` (=0.001) is not so bad. If you like to reduce computational time use 0.01 or so, but a little dangeous in cases. With 0.0001, we can check stability on it.

- `QGcut_psi` is a little (usually 0.5 or so) larger than `QpGcut_cou`. It becomes accurate if we use large `QpGcut_cou`. But it enlarge size of IPW(interstitial plane wave) part of Mixed product basis. For test, try 2.7, 3.2, 3.7 for `QGcut_cou` (and add 0.5 or 1 for `QGcut_psi`). Larger one is expensive.

- `dw` and `omg_c` specify real space bins which we accumulate imaginary part weight of polarization functions. dw is bin width (in Ry) at omega=0, then bin width is twiced at `omg_c`. The bin width becomes quadratically coarser at high energy. If bins are too wide, dielectric function can be less accurate, but results are not necessarily so much affected. For metal,

our code can capture Drude weight numerically. We do not need to be so sensitive to the choice of them usually.

- `alpha_offG`
  `alpha_offG` corresponds to α to define auxially function. `alpha_offG`=1d0 is usually good in the sense that it seems to be almost a limit at $\alpha \to 0$. So we can usually fix it as `alpha_offG`=1d0, and check the convergence as for `n1n2n3`.

- `delta` (this may be not used now): keep it as it is.

- `{niw}` : 1 integer.
  Number of integration points along the imaginary axis 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.

- `GaussSmear` and `esmr` : 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.

- `Q0Pchoice`: This controls how to choose offset Gamma points in a new version (2012 version). It is usually better to used Q0Pchoice=1. (For slabs, Q0Pchoice=2 may be better; need check more. In anyway, it is problematic to use unbalanced k points for anisotropic cell).

- `deltaw` (only for one-shot GW) 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.

- n1n2n3: BZ division for k point integration. We usually take '4 4 4' to '8 8 8' for GaAs. For metal such as Fe, '10 10 10' or more is better.

- lcutmx(atom) is the l cutoff of product basis for atoms in the primitive cell (do lmchk for atom id). In the case of Oxgen, we can usually use lcutmx=2 (need check by the diffence when you use lcutmx=2 or lcutmx=4). Then the computational time is reduced well.

- Other part of product basis section in <PRODUCT_BASIS>... </PRODUCT_BASIS> is usually not need to be touched. But if you like to calculate big systems with smaller CPU time, or do very accurate calculations, we may need to touch it. It is described elsewhere.

- <QPNT> tag is to spesify one-shot GW. At which point, do we calculate QPE, not for QSGW. If you set k point in it not on regular mesh point, you have to set 'Any Q on'; but it is expensive. Since QSGW have ability to plot energy band within full BZ, it should be better to do it.

- **set QPNT for eps mode**: See Section 'calculation of susceptibility': <QforEPS> and <QforEPSL> are to specify at which k point do we calculate susceptibility. It is for epsPP_lmfh, eps_lmfh (dielectric functions) and epsPP_chipm (spin susceptibility). For eps modes (scripts `eps_*`, which are for linear responses, you have to specify q point in the following ways.

  To specify it directry (cartesian),

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

  You can specify Q points in addition 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).

- AnyQ (default is off. not for gwsc)
  If this is on, you can spefify any Q point which is not on the mesh point in the one-shot gw gw_lmfh. For the purpose, we need to prepare eigenfunctions at extra **k** points. But it is automatic. In order to make computaiton efficients, 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).

We need a setting in ctrl file to read sigm file (HAM_SIGP). It is simplified now, and not need to care it so much. As we set RDSIG=12 in defauls, lmf read sigm file and add it to one-body potential as long as sigm.* exist.

> **NOTE for old users**: We now set `SIGP[MODE=3 EMAX=9999.]` in ctrl file to read self-energy in lmf (or lmf-MPIK). This is because we use very localized MTOs (similar with the Maxloc Wannier). Our test shows reasonable results and this simplify algolisms. In my previous version, we asked you to use `SIGP[MODE=3 EMAX=2.0]` where EMAX is a little (0.5Ry) less than `emax_sigm`. If something strange occurs, try this setting).

In principle, QSGW result should not depended on the choice of XCFUN. However, it can affect slightly. In our tests, it seems slightly better to use vwn (XCFUN=1) for QSGW calculations. (BUT need to check more...)

4.1.1 <PRODUCT_BASIS> section

— I think this contains old description; not need to read this —

This section is to define product basis to expand W and so wihin MT. 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, whi
 lcutmx(atom) = maximum l-cutoff for the product basis.  =4 is required for atoms with valenc
  4  3
  atom   l  nnvv  nnc ! nnvv: num. of radial functions (valence) on the augmentation-waves, 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.
> $\Sigma_{\rm x}$ 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 × 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.

4.1.2 ANFcond:not maintained well now...

—- skip to read this —

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).

4.2 Do QSGW calculation

Let us perform QSGW calculation. Fot this purpose, we use a script `gwsc`.

```
gwsc (number of iteration+1) -np (number of nodes) (id of ctrl)
```

If (number of iteration+1)=0, it gives one-shot calculation from LDA. But it is different from the usual one-shot in literatures; since it caltulates off-diagonal elements of self-energy also, we can plot energy band dispersion plot. In cases (such as usual semiconductors), it gives rather reasonable results in comparison with experiments from practical point of veiw.

This is an exaple of one iteration of QSGW cycle. (now a little different but essentiall similar)

```
takao@TT4:~/ecalj/test1$ gwsc 0 -np 2 si
gwsc 0 -np 2 si
### START gwsc: ITER= 0, MPI size=  2, TARGET= si
--- No sigm nor sigm.$TARGET files for starting ---
 ---- goto sc calculation with given sigma-vxc --- ix=,0
No sigm ---> LDA caculation for eigenfunctions
        Start  mpirun -np 2 /home/takao/ecalj/TestInstall/bin/lmf-MPIK  si > llmf_lda
OK! --> Start  echo 0| /home/takao/ecalj/TestInstall/bin/lmfgw si > llmfgw00
OK! --> Start  echo 1|/home/takao/ecalj/TestInstall/bin/qg4gw > lqg4gw
OK! --> Start  echo 1|mpirun -np 2 /home/takao/ecalj/TestInstall/bin/lmfgw-MPIK  si> llmfgw01
OK! --> Start  /home/takao/ecalj/TestInstall/bin/lmf2gw >llmf2gw
OK! --> Start  echo 0|/home/takao/ecalj/TestInstall/bin/rdata4gw_v2  > lrdata4gw_v2
OK! --> Start  echo 1| /home/takao/ecalj/TestInstall/bin/heftet > leftet
OK! --> Start  echo 1| /home/takao/ecalj/TestInstall/bin/hchknw > lchknw
OK! --> Start  echo 3| /home/takao/ecalj/TestInstall/bin/hbasfp0 > lbasC
OK! --> Start  echo 3| mpirun -np 2 /home/takao/ecalj/TestInstall/bin/hvccfp0 > lvccC
OK! --> Start  echo 3| mpirun -np 2 /home/takao/ecalj/TestInstall/bin/hsfp0_sc > lsxC
OK! --> Start  echo 0|/home/takao/ecalj/TestInstall/bin/hbasfp0  > lbas
OK! --> Start  echo 0| mpirun -np 2 /home/takao/ecalj/TestInstall/bin/hvccfp0  > lvcc
OK! --> Start  echo 1|  mpirun -np 2 /home/takao/ecalj/TestInstall/bin/hsfp0_sc > lsx
OK! --> Start  echo 11|  mpirun -np 2 /home/takao/ecalj/TestInstall/bin/hx0fp0_sc > lx0
OK! --> Start  echo 2|  mpirun -np 2 /home/takao/ecalj/TestInstall/bin/hsfp0_sc  > lsc
OK! --> Start  echo 0|  /home/takao/ecalj/TestInstall/bin/hqpe_sc  > lqpe
OK! --> == 0 iteration over ==
OK! --> Start  mpirun -np 2 /home/takao/ecalj/TestInstall/bin/lmf-MPIK  si > llmf_gwscend.0
OK! ==== All calclation finished for gwsc 0 -np 2 si ====
```

Here `echo (integer)` is readin in at the beging of the code. To see it, please look into gwsc script (gwsc is at ecalj/fpgw/exec/ and copied to your bin/

by make install2). In anyway, this console output shows calculations finished normally.

Now we get rst.si and sigm.si file which contains (static version of) self-energy minums $V_{\rm xc}^{\rm LDA}$. What we did is the one-shot GW from LDA result; but note that we calculate not only diagonalelements but also off-diagonal elements.

We can write energy dispersion (band plot) in the same manner in LDA. To do it, we need rst.si, sigm.si, ctrl.si, QGpsi, and ESEAVR. (but QGpsi and ESEAVR are quickly reproduced). After you have syml.si (e.g. in ecalj/MATERIALS/), Do

```
$ job_band_nspin1 si
```

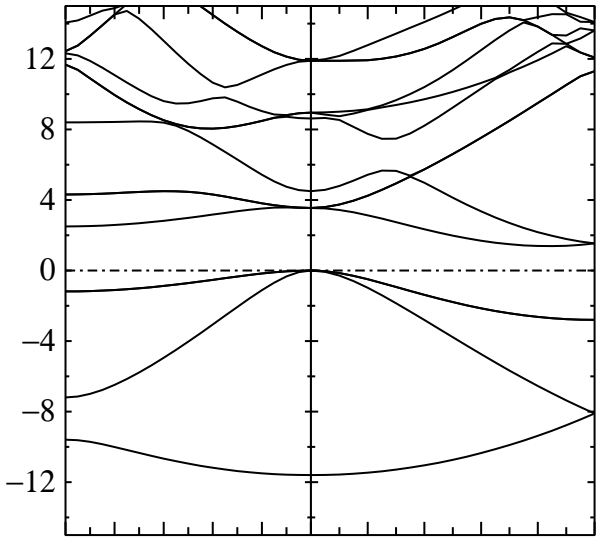

Figure 3: Si, one-shot GW with off-diagonal elements

You can observe large band gap as shown in the Fig.4.2. (To see it again, `gnuplot bnds.gnu.si -p`. All plots are in gnuplot, thus it is easy to replot it as you like).

We have QPU file (and also QPD for spin=2), which contains content of the diagonal part of self-energy. It will be explained elsewhere.

You can make total DOS and PDOS plot by

```
$ job_tdos si
$ job_pdos si
```

CAUTION:pdos plot is not allowed for so=1. (even tdos–¿ ask to t.kotani.) In job_pdos, we use no space group operation and read `simg_fbz.*` (simga file in all q point in BZ) file.

To get final QSGW results, we have to repeat iteration until eigenvalues are converged. Note that total energy shown by console output llmf (and also shown in save file) is not so meaningful in the QSGW; we just take it as an indicator to check convergence. Let us repeat 5 iteration more. "-np 2" means one core to use.

```
$ gwsc 5 -np 2 si
### START gwsc: ITER= 5, MPI size=  2, TARGET= si
 --- sigm is used. sigm.$TARGET is softlink to it  ---
 ---- goto sc calculation with given sigma-vxc --- ix=,0
 we have sigm already, skip iter=0
 ---- goto sc calculation with given sigma-vxc --- ix=,1

 ...(skip here) ...

OK! --> Start  echo 11|  mpirun -np 2 /home/takao/ecalj/TestInstall/bin/hx0fp0_sc > lx0
OK! --> Start  echo 2|  mpirun -np 2 /home/takao/ecalj/TestInstall/bin/hsfp0_sc  > lsc
OK! --> Start  echo 0|  /home/takao/ecalj/TestInstall/bin/hqpe_sc  > lqpe
OK! --> == 5 iteration over ==
OK! --> Start  mpirun -np 2 /home/takao/ecalj/TestInstall/bin/lmf-MPIK  si > llmf_gwscend.5
OK! ==== All calclation finished for gwsc 5 -np 2 si ====
```

Note that we do skip 0th iteration (it is for one-shot from LDA) since we start from rst.si and sigm.si given by one-shot LDA. Thus we do just five iterations. Infomation of eigenvalues are in `QPU.(number)run` files. (for magnetic systems with nspin=2), wee have `QPD.(number)run` also). Check it by ls;

```
$ ls QPU.*run
QPU0.run  QPU.1run  QPU.2run  QPU.3run  QPU.4run  QPU.5run
```

(These are overwritten when we again repeat gwsc; be careful.) Note that `QPU0.run` was old one when you did 1-shot GW from LDA at the begining.

In order to check convergece calculations going well, do

```
$ grep gap llmf*
```

This shows how band gap changes in llmf.*run files.

Let us check convergence of the QSGW calculations. For this purpose, it is convenient to take a difference of QPU(QPD) files by a script `dqpu`. These files are human readable. To compare `QPU4.run` and `QPU5.run`, do

```
$ dqpu QPU.3run QPU.4run
```

Then we see a list of numbers (these are the differences of values in QPU files). Then it shows at the bottom as

```
Error! Difference>2e-2 between:   QPU.4run   and   QPU.5run
:  sum(abs(QPU-QPD))= 0.05736
```

but you don't need to care it so much. You rather need to check the difference of values. I can say most of all difference (especially aroung the fermi energy are) are almost 0.00eV or 0.01eV, we can judge QPEs are converged. If not converged well, you may need to repeat `gwsc` again. (when the size of two QPU files are different, dqpu stops.)

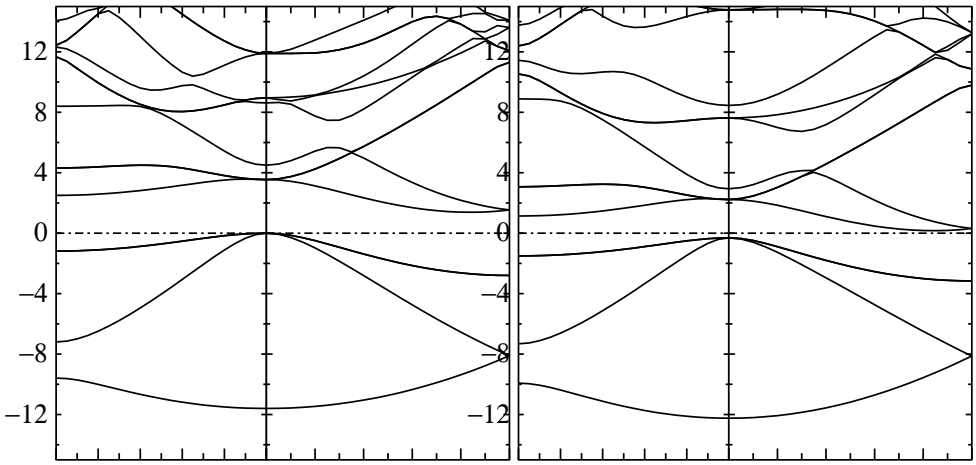

Figure 4: band plot(Si, QSGW one-shot test) and band(Si) (GGA)

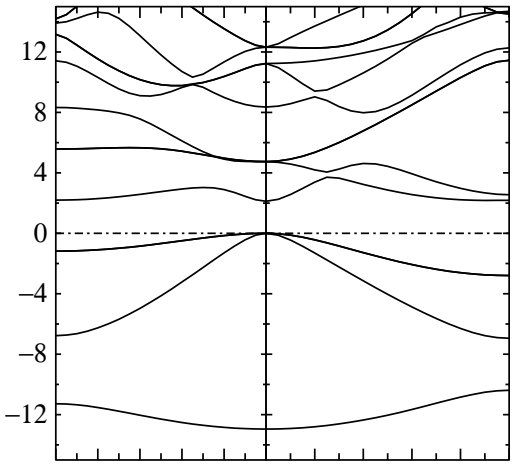

Figure 5: band(GaAs), QSGW (test case)

4.3 Output files of QSGW

4.4 QPU

This is only the diagonal part of self-energy [1] in human format. It is like this

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

```
 ================================================================
  quasiparticle energies MAJORITY
 ================================================================
E_shift=  0.0000000000000000D+00  0.0000000000000000D+00  0.0000000000000000D+00 eV

          q               state  SEx   SExcore SEc    vxc   ---   dSEnoZ  eQP(starting by lmf)  eHF  Z=1  FWHM=2Z*
  0.00000  0.00000  0.00000  1  -35.37 -14.40  10.45 -43.40   0.00   4.08 -15.46   0.00   0.00 -21.83 1.00   0.0000
  0.00000  0.00000  0.00000  2  -35.37 -14.40  10.45 -43.40   0.00   4.08 -15.46   0.00   0.00 -21.83 1.00   0.0000
  0.00000  0.00000  0.00000  3  -35.37 -14.40  10.45 -43.40   0.00   4.08 -15.46   0.00   0.00 -21.83 1.00   0.0000
  0.00000  0.00000  0.00000  4  -35.42 -14.53  10.42 -43.69   0.00   4.15 -15.40   0.00   0.00 -21.67 1.00   0.0000
  0.00000  0.00000  0.00000  5  -35.42 -14.53  10.42 -43.69   0.00   4.15 -15.40   0.00   0.00 -21.67 1.00   0.0000
  0.00000  0.00000  0.00000  6  -17.75  -2.76   6.72 -14.41   0.00   0.62 -12.99   0.00   0.00 -19.10 1.00   0.0000
  0.00000  0.00000  0.00000  7  -14.52  -2.81   2.67 -15.77   0.00   1.11  -0.12   0.00   0.00  -1.68 1.00  -0.0000
  0.00000  0.00000  0.00000  8  -14.52  -2.81   2.67 -15.77   0.00   1.11  -0.12   0.00   0.00  -1.68 1.00  -0.0000
  0.00000  0.00000  0.00000  9  -14.52  -2.81   2.67 -15.77   0.00   1.11  -0.12   0.00   0.00  -1.68 1.00  -0.0000
  0.00000  0.00000  0.00000 10   -6.18  -4.31  -4.47 -16.76   0.00   1.79   0.10   0.00   0.00   6.37 1.00   0.0000
  0.00000  0.00000  0.00000 11   -4.67  -1.77  -4.49 -12.61   0.00   1.69   3.29   0.00   0.00   9.47 1.00  -0.0000
  0.00000  0.00000  0.00000 12   -4.67  -1.77  -4.49 -12.61   0.00   1.69   3.29   0.00   0.00   9.47 1.00  -0.0000
  0.00000  0.00000  0.00000 13   -4.67  -1.77  -4.49 -12.61   0.00   1.69   3.29   0.00   0.00   9.47 1.00  -0.0000
  ...
```

From the 6h row, we have the eigenvalue datas. All of the unit of energy is in eV. Zero of eQP is at the Fermi energy (or middle of VBM and CBM).

`q` : $\mathbf{k}$ vector

`state`: Band index n, which is from the lowest eigenvalue (not include cores).

`SEx`: $= = \langle\Psi_{\mathbf{k}n}|\Sigma_{\mathrm{x}}^{\mathrm{core2+valence}}(\mathbf{r},\mathbf{r}')|\Psi_{\mathbf{k}n}\rangle$

`SExcore`: $= \langle\Psi_{\mathbf{k}n}|\Sigma_{\mathrm{x}}^{\mathrm{core1}}(\mathbf{r},\mathbf{r}')|\Psi_{\mathbf{k}n}\rangle$

`SEc`: $= \langle\Psi_{\mathbf{k}n}|\Sigma_{\mathrm{c}}^{\mathrm{core2+valence}}(\mathbf{r},\mathbf{r}',\epsilon_n(\mathbf{k}))|\Psi_{\mathbf{k}n}\rangle$

`vxc`: LDA exchange correlation energy. $\langle\Psi_{\mathbf{k}n}|V_{\mathrm{xc}}^{\mathrm{LDA}}([n_{\mathrm{total}}],\mathbf{r})|\Psi_{\mathbf{k}n}\rangle$

`dSEnoZ`: $\langle\Psi_{\mathbf{k}n}|\Sigma_{\mathrm{x}}^{\mathrm{core1}}(\mathbf{r},\mathbf{r}')+\Sigma_{\mathrm{xc}}^{\mathrm{core2+valence}}(\mathbf{r},\mathbf{r}',\epsilon_n(\mathbf{k}))|\Psi_{\mathbf{k}n}\rangle-\langle\Psi_{\mathbf{k}n}|V_{\mathrm{xc}}^{\mathrm{LDA}}([n_{\mathrm{total}}],\mathbf{r})|\Psi_{\mathbf{k}n}\rangle$
= `SEx + SExcore + SEc - vxc`

`eQP`: QP energy. $\epsilon_n(\mathbf{k})$+`dSE` (this is given first by lmf)

`eHF`: HF energy of 1st itteration. $\epsilon_n(\mathbf{k})$+`SEx + SExcore -vxc`

`ReS(elda)`: $\mathrm{Re}\langle\Psi_{\mathbf{k}n}|\Sigma_{\mathrm{x}}^{\mathrm{core1}}(\mathbf{r},\mathbf{r}')+\Sigma_{\mathrm{xc}}^{\mathrm{core2+valence}}(\mathbf{r},\mathbf{r}',\epsilon_n(\mathbf{k}))|\Psi_{\mathbf{k}n}\rangle$

4.5 XCU

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

4.6 SEXU

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

4.7 SEXcoreU

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

4.8 SECU

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

4.9 TOTE.UP (TOTE.DN)

This is a central output. It contains 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.

4.10 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). **They have extentions of LDA (even in its file) but it is wrong; it is just what is calculated by lmf; thus sigm maybe added**

4.11 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.

4.12 DOSACC2.lda

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

4.13 Core_ibas*_l*.chk

Used core eigenfunctions.

4.14 VXCFP.chk

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

5 calculation of susceptibility

We can calculate dielectric functions by epsPP_lmfh (no local field corrction), or eps_lmfh (with local field correction) for given q point. First thing is that it require many number of k point (for example, 'n1n2n3 15 15 15' or more for Si). After set it with setting QPNT as explained in GWinput section, start `epsPP_lmfh si` or `eps_lmfh si`. Then you will have 'EPS*.nlfc.dat'(no local field collection) and/or 'EPS.dat'. Simple samples are in TestInstall/gas_eps_lmfh

and /gas_epsPP_lmfh. (but too small number to compare with high level calculations). It is time-consuming to use large product basis for `eps_lmfh` (`epsPP_lmfh` do not use product basis), we can use small product basis; but we lose not so much accuracy.

As for epsPP_chipm mode, we can calculate non-interacting spin susceptibility as for "MagAtom (id)" which is given in GWinput.

xxxxxxxxxxxxxxxxxxxxx
under construction
xxxxxxxxxxxxxxxxxxxxx

A How to add spin-orbit coupling

```
Do LDA and/or QSGW with SO=0 first.
Then apply the spin-orbit coupling by perturbation.

After converged with nspin=1 (or 2), create new directory and copy
  ctrl.gas, rst.gas, sigm.gas, QGpsi, ESEAVR
to it. Then we set
  nspin=2
  METAL=3
  SO=1  (this is ldots calculation off-diagonal elements included).
  Q=band (we do not change potential.)
in ctrl.gas.
Then run
>lmf gas >& llmf_SO
You can see "band gap with SO" by
> grep gap llmf_SO.
Then you can see two same lines.
 VBmax = 0.101949  CBmin = 0.236351  gap = 0.134402 Ry = 1.82786 eV
 VBmax = 0.101949  CBmin = 0.236351  gap = 0.134402 Ry = 1.82786 eV
(two lines are because of two-band path mechanism,
which asks less memory usage than a path mehod)
This is the band gap with SO as a first-order perturbation starting on
top of the "QSGW without SO". When you use ctrl file generated by
ctrlgenM1.py. You can do the above procedure with
>lmf --rs=1,0 gas -vnit=1 -vso=1 -vnspin=2 -vmetal=3 --quit=band)
(--rs=1,0 read rst.gas but not write rst.gas. Ses lmf --help
 -vso=1 replace the setting of % const so=0 with so=1).

For band plot, you can use the same procedure
for the case without SO. (Look into the job_band_nspin1 script.
You have to modify it so that --rs=1,0 gas -vnit=1 -vso=1 -vnspin=2
-vmetal=3 --quit=band is added as arguments for >lmf --band:syml ...).
```

```
---
For given sigm file, it is possible to do full self-consisent SO calculations
(then we do not set Q=band). However, note that Vxc is fixed in QSGW,
it is not necessary better than the above procedure.
```

B How to set local orbitals.

In cases, it is better to add local orbitals and check convergence, especially in QSGW.

```
  Do "lmfa |grep conf" to check used MTO basis.
  (it just gives atm.gaas file. neglect ves... files).
  Then we see integer numbers at int(P) column.
  It show up to high l, but used number of l for MTO is just controlled
  by number of EH,RSMH in ctrl file.

  We usually set int(P)z (principle quantum number of local orbital)
  to int(P)-1 or int(P)+1.

  For example if we set SPEC_ATOM_PZ=0,4.9,..., it means we set local orbital
  of 4p (numbers after PZ= are ordered as PZ=s,p,d,f,... ).
  Zero means no local orbital for s channel.

  We need to set not =4 but =4.9 for PZ. It is the continuous principle
  quantum number (see next paragraph).

  Let let us explain "Continious principle quantum number".
  It is defined as
     P=principleQuantumNumber + 0.5-atan(1/phi dphi/dr)
  Its fractioanl part 0.5-atan(1/phi dphi/dr) is closer to unity for
  core like orbital, but closer to zero for extended orbitals.
  There are the free electron value for P
  (in the case of constant potential, phi=r^(l+1)).
  They are shown in 'pfree' in console output. These numbers are
   1.500000
   2.250000
   3.147584   0.147584=0.5-atan(2+1)
   4.102416
   5.077979

 Examples of choice:
  Ga p: in this case, choice 0 or choice 3 is recommended.
     (0) no lo (4p as valence is default treatment without lo.)
        3p core, 4p valence, no lo: default.
        Then we have choice that lo is set to be for 3p or 5p.
```

```
   (1) 3p lo
      Set PZ=0,3.9
   (2) 5p lo ---> 4p val (PZ>P)
      Set PZ=0,5.5
      5.5 is just simply given by a guess. If 5.2 or something, it may fails
      because of poorness in linear-dependency. We may need to observe
      results should not change so much on the value of PZ.

Ga d: (in this case, choice 0 or choice 1 is recommended).
   (0) no lo (3d core, 4d valecne, no lo: default.)
        Then we have choice that lo is set to be for 3d,4d,5d.
   (1) 3d lo
       Set PZ=0,0,3.9
   (2) 5d lo
       Set PZ=0,0,5.5

  In default, lmf try to read rst file first. To read atm file,
  you have to do lmf --rs=1,1,0,0,1, for example. See lmf --help
  Becase rst file keeps the setting of MTO, change in ctrl is not
  reflected without the above option to lmf.
```

C How gwsc script works.

We now explain what is done in gwsc script.

The self-consistency loop consists of two steps (1)lmf part, and (2) GW part. In the (1)lmf part, we do self-consistent calculations for given exchange-correlation (or LDA/GGA at starting point), then we need to generate eigenfunctions and eigenvalues required for the (2)GW part.

C.1 lmf part

We first perform self-consistent LDA/GGA calculation with adding sigm to one-body poteneital if it exist. [2]

- **echo 0 |lmfgw**: Get some small information files (crystal structure info and so on) used in the next **qg4gw**.
- **echo 1 |qg4gw** : Get **k** points used in the GWcalculations and the correponding **G** vectors. See the output lqg4gw.
- **echo 1 |lmfgw** : Calculate eigenfunctions, eigenvlaues, and $\langle\psi|V_{\rm xc}^{\rm LDA}|\psi\rangle$ for these **k** .
- **lmf2gw**: store these datas into DATA4GW_V2 and CphiGeig , whose I/O is controlled by a key subroutine **gwinput_v2.f**.

[2] **echo 1|qg4gw** and so on means that we invoke **qg4gw** with the argument 1 from the standard I/O (not from console).

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

C.2 GW part

This part is in order to generate Sigma-$V_{xc}^{\rm LDA}$ from following files given by (1) lmf part.
• DATA4GW_V2 : Crystal structures and so.
• CphiGeig : Eigenvalues and Eigenfunctions this is divided into Cphi and Geig in rdata4gw.
• 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.

- **rdata4gw_V2**: Read DATA4GW_V2,CphiGeig, and generate files ECORE,PHIVC,CPHI,GEIG,VXCFP,... (see `fpgw/main/rdata4gw_v2.m.F`)
- **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 probably 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_sc** : gives the exchange part of the self-energy.
- **echo 11**|**hx0fp0_sc** : gives the correlated part of the screened Coulomb interaction $W - v$.
- **echo 2**|**hsfp0_sc** : gives the correlated part of the self-energy.
- **echo 0**|**hqpe_sc** : gather datas and write down final results into files sigm, QPU and so on.

NOTE: As for core exhcnage part, we can do it easily without using product basis. We will have to fix it since it becomes somehow time-consuming for large systems.

D Cautions

We will detail it. You don't need to see this. Ask me if you have questions.

```
== emax cutoff
  It is not necessary good to enlarge emax
== Used MTO
 Near beginig of console output, what MTO you use is shown as: (GaAs case).
 sugcut:  make orbital-dependent reciprocal vector cutoffs for tol= 1.00E-06
 spec      l    rsm    eh     gmax    last term    cutoff
  Ga       0*   1.13  -1.00   6.579    1.19E-06    1459
  Ga       1*   1.13  -1.00   7.028    1.26E-06    1807
  Ga       2*   1.13  -1.00   7.475    1.09E-06    2109
  Ga       3    1.13  -1.00   7.920    1.06E-06    2637
  Ga       0*   1.13  -2.00   6.579    1.19E-06    1459
  Ga       1*   1.13  -2.00   7.028    1.26E-06    1807
  Ga       2    1.13  -2.00   7.475    1.09E-06    2109
  As       0*   1.18  -1.00   6.300    2.13E-06    1243
  As       1*   1.18  -1.00   6.720    1.26E-06    1471
  As       2*   1.18  -1.00   7.140    1.37E-06    1837
  As       3    1.18  -1.00   7.558    1.05E-06    2229
  As       0*   1.18  -2.00   6.300    2.13E-06    1243
  As       1*   1.18  -2.00   6.720    1.26E-06    1471
  As       2    1.18  -2.00   7.140    1.37E-06    1837

== gwsc cause error stop.
 Have you ever changed MTO setting? Consistent with GWinput?

== QSGW for Fe.
  It is better to use 3p as core. Adding 4d as local orbital seems to
  make calculation a little unstable (magnetic moment is a little
  dependent on emax in GWinput).
  n1n2n3 10 10 10 gives reasonabel result.

== RSRNGE: enlarge RSRNGE ===
  Use RSRNGE=10 or so (in cases, RARNGE=20 or more is required),
  for large number of k points. Try and enlarge it if it fails with a
  message "Exit -1 rdsigm: Bloch sum deviates more than allowed tolerance (tol=5e-6)".
  We will have to make it automatic in future.

== EPS mode,
  Check Im part of chi0 is smoothly damping at high energy (typically
  1Ry or larger enengy range). If there is some large Im part remains,
  something strange (usually due to orthogonality problem of
  eigenfunctions when you set low q).
```

```
   Related source codes are in ecalj/lm7K/ .
   A command ecalj/lm7K/ctrlgenM1.py can generate 'standard input file (ctrl file)'
   just from a given crystal structure file called as ctrls file.
   Binaries are lmf and lmf-MPIK (MPI k-parelell verion).

   Recently, I renewed some part of algolism of GW/QSGW calculations
   (some ideas are taken from from PRB.81,125102(2010)
    and Copmuter Physics Comm. 176(2007)1-13).
   ---> this is better than old versions; speed, memory (file size),
   and accuracy for anisortopic systems.
   For comparison, you can use old version in .git (gitk --all and check it out).

=== When calculation in LDA level fails ===
when calculation fails in LDA level.
  (1) smaller MT
  (2) fewer PW. smaller pwemax.
  (3) core as semicore.
```