ecalj — Get statrted (aug2013)

Takao Kotani

September 2, 2013

Contents

1 Introduction

"ecalj" is a package for first-principle electronic structure calculation. It has unique features. Especially, we can perform the quasiparticle self-consistent GW (QSGW) calculation based on the PMT method (=Linearized APW+MTO method) as seen in Sec.1.1.

The ecalj web site is at

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

We can download ecalj package from this site. Free to download and use it, but we need you to clarify your acknowledgment to the package in your publications. We also have a web site at http://pmt.sakura.ne.jp/, however, most of all are yet in Japanese and not yet well organized.

The ecalj is related to another FP-LMTO package lmv7 seen 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) until now. All codes are now in f90, and we have simplified complications before the branching. We have a MPI parallization for QSGW (altohugh it is limited yet).

Read this manual first. Then I show where we should read in the lmv7 manual if necessary. Our code is quite simplified (and trying to simplified) in its usage for users.

If you have something, let takaokotani(at)gmail.com know it. With your ideas, we like to have collaborations and newer development on it. Or I may clean up this manual, or help something on it.

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. Then we can calculate one-body potential for given eigenfunctions and eigenvalues based on the density functional theory (DFT) in the LDA or GGA (hereafter 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 by perturbation.

Based on such a solver of one-body problem, we can implement kinds of methods; e.g, linear responses, transport and so on. In addition, we can implement higher-level approximations such as the QSGW method as we explain in the followings

Such one-body problem solvers (in the case of linear methods) are characterized by (i) linear combinations of what basis set to represent 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 muffin-tin orbitals

(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 eigenfunctions are represented by linear combinations of the APWs, MTOs, and the lo's. Then the electron density and the one-body potential are represented by "smooth part + onsite muffin-tin (MT) part - counter parts". The counter part is to remove smooth part within MTs. This formalism (Soler-Williams formalism) is also used in the projected augmented wave (PAW) 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 ability is limited; it can fail when its starting point (eigenfunctions and eigenvalues supplied by LDA) is problematic. Thus T.Kotani with collaborators have developed the QSGW method. The QSGW now becomes popular, performed by other researchers. In principle, results by QSGW do not depend 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 (shown total energies in QSGW calculations are dummy now).

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

Recent development 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 requires 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 accelerate the method, but not yet so much. Minimum MPI parallelization is implemented). The computational effort is $\propto N^4$ in dominant part.

1.2 What can we expect for QSGW?

We expect that QSGW will give similar results with hybrid functional methods, which use Vxc = alpha*LDA+(1-alpha)*(Fock exchange), where alpha is taken to be about 0.25 usually. However, the QSGW automatically determine W (the screened Coulomb interaction) self-consistently; very roughly speaking, such alpha is automatically determined.

Two important aspects of non-local potential (DFT gives local potential). One is the onsite non-locality; it is also taken into account by LDA+U by a model. However, note that relative shift of O(2p) band with respect to 3d band is not in LDA+U. The other is the off-site non-locality (mainly between nearest neigbors), which gives LUMO-HOMO gap.

The main purpose of QSGW is to determine "best H_0", on which we describe one-particle picture to describe theories. In addition, W is determined consistently.

In comparision with LDA, we see differences;

- Band gap.
- Band width. As for sp bands, it is enlarged (except very low density case such as Na). As for localized bands like 3d electrons, they can be narrowed.
- Relative position of bands. eg. O(2p) v.s. Ni(3d), exchange splitting between up and down (like LDA+U)
- hybridyzation of 3d bands with others. QSGW tends to make eigenfunctions localized.
- magnetization.

In principle, we can calculate total energy (in RPA for QSGW), but not yet. We can calculate the screened Coulomb interaction $W = v/\epsilon$. In addition, spin susceptibility χ_0^{+-} (In near future, we need to combine it with W).

1.3 What is in this booklet?

Here we show minimum on the ecalj package.

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

- (NOT YET) Relaxiation of atomic positions in LDA/GGA.

To perform calculations in ecalj becomes easier recently. After we prepare *a crystal structure file* named as `ctrls.*`, we run a a script (`ctrlgenM1.py`) to generate `ctrl.*` It contains (not so bad) default setting to do following calculations.

To help writing `ctrls.*`, ecalj contains samples and a converter between VASP-POSCAR format and ecalj-ctrls format.

However, we need to know a little more; 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.

The QSGW code is version controlled by git, and hosted by github which anyone can access. Thus your results is reproducible by others (need to specify version number and input files.)

2 Install

Look into ecalj/README. (it is shown 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 somewhere else where you specified at the bottom of Makefile.)

3 LDA calculations

Calculations are performed by following steps.

1. 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 (symmetry).
2. 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 edit the generated ctrl file for your purpose. You can also write `ctrl.*` by your hand from scratch.
3. Do lmfa (just calculate atoms (MT sites) placed in the cell). It also calculate core eigenfunctions and valence electron charge for initial condition. And start main calculation lmf successively. Then we get self-consistent result of LDA.
4. Plot energy band, DOS, PDOS, by running scripts.

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

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

"dot + `(id)`" is extension which we supply (only lower letter allowed). For example, `ctrls.cu`, `ctrls.lagao3`. Any name works.

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 starting from '`% const`' define variables and set values (in these cases, `da`, `alat`, and `bohr`). Then the variable `alat` is referred to as `{alat}`; in the cu case, `{alat}` means 6.798. Lines not start from "#" nor "%" are main content in the ctrls file. [1]

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 of GaAs, 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. [2] In cases, we specify such subtokens as SITE_ATOM_POS.

In the SITE category, 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 coordinate, but in the unit of ALAT+DALAT.

[1] for these variables, we can give values when we start program. E.g, 'lmf -vdalat=0.1 si'; this alat is recorded in save.si file.

[2] This may looks slightly uncomfortable since the end of range of ATOM is not clearly shown; it end just at the next ATOM token or new category.

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 example 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 parenthesis, but no space for separation). (After ctrl generated, these is a command lmchk to check weather crystal structure is correctly given or not. And show symmetry information, and so on.) We can use default atomic symbols (to check default atom name (MT name) type `ctrlgenM1.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 now. 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 needed if we prepare ctrl file directory).

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 afterward 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 specified are shown at the beginning of the console 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 recognizes 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 information.

- 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 relaxation) 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
...
```

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

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

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

- `IO_TIM` is for debugging. It shows 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 anisotropy 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 multiple SPEC_ATOM=foobar).

  Then we place these MTs at SITE sections in the cell. 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. Setting `POS=` under SITE_ATOM=foobar means that we place MT named as foobar defined in SPEC_ATOM=foobar. In addition, we can set SITE_ATOM_RELAX, if you like to find relaxed structure (we simultaneously set DYN category) in LDA. As for relaxation, 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 info 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 refereed 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 recognized 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 computational confusion.

  PZ is to set local orbital (if not, no local orbitals). EH and RSMH are to specify first set of MTOs.(We can check how local orbitals are set by lmfa explained in the next section). EH2 and RSMH2 are to specify 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 comma(,) 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 (continuous principle quantum number) are large ~ 0.9 for core like orbital, and smaller for extended orbital ~ 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 (or its envelope function as a smooth Hankel function). 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 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 condition 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 manage 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 minimum 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 ctrls.* and ctrl.* These are in lm7K/TESTsample/* and ecalj/CMDsampls. In addition, ecalj/MATERIALS contain many samples (need a command); see a later subsection.

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 yet explained some useful feaures; STRUC category to deform crystal; DYN category for dynamics; LDA+U treatment; Adding background charge; Core-Hole treatments. 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 keeps console output (standard output) to a file (llmfa in this case).

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 charge 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 column 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 summary we have 4s,4p,4d,3d,4f as valence. This means we use corresponding 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 console 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` means it stop at 8th iteration, although 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 Little 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-Faulkner energy.

"grep diffe lllmf" shows how the changed of total energy (and charges) during iteration. diffe 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 (we will shrink it). As long as it converged well, you don't need to look into it in detail. Eigenvalues 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). "`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. These are in Ry; Zero level is not so meaningful (detemined with the potential at MT boundaries).

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 initial atom file by lmfa). We can do parallel 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, band, and pdos from the result of lmf self-consistent calculations. We have scripts

```
job_tdos, job_band_nspin1, job_band_nspin2, job_pdos
```

. Look into these scripts, and then you see how to plot them.

For total DOS plot, it is better to check ctrl file; `BZ_TETRA=1`(this is default; 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

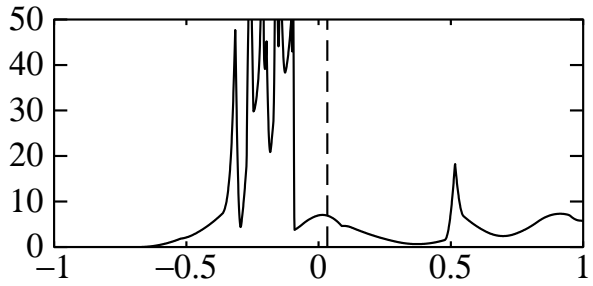

Figure 1: DOS(Cu)

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/MATERIALS/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 calculation 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 (need to do it by changing the script).)

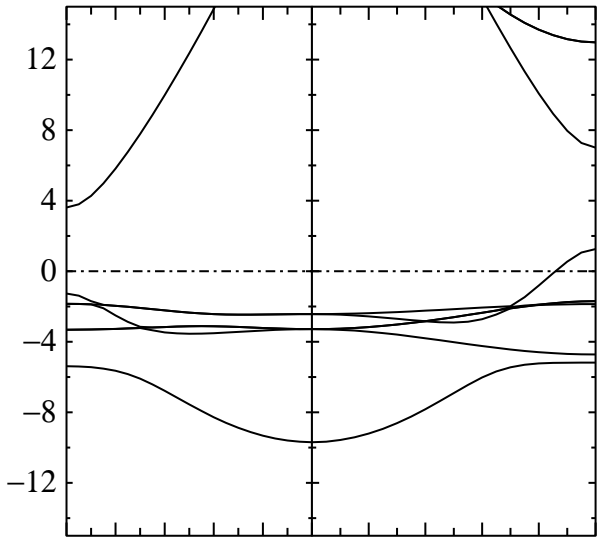

Figure 2: band plot(Cu)

For PDOS plot,

```
job_pdos cu
```

It shows many figures in gnuplot. It is time-consuming because we use no symmetry to distinguish all lm channels. (PDOS is not yet implemented for SO=1 case; spin-orbit coupling Ł$\dot{S}$ is added.) We have to re-organize script of gnuplot (pdos.site*.*.glt) for your purpose. In principle, 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 separately. (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
```

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 a compact manner (in addition, options passed to ctrlgenM1.py and options to lmf-MPIK are included). See `ecalj/MATERIALS/README`. 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 difficult).

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 template `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 advantageous 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. Each line is independent; "keyword" followed by some of numbers (or on or off for logical switch of keyword). Except such lines, there are tag sandwiched sections such as `<PRODUCT\_BASIS> ... </PRODUCT_BASIS>`, no comment lines in the middle allowed. They are read by read(*,*) (thus spaces cause no problem).

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

- `emax_sigm` is the maximum limit of the self-energy (measured from the Fermi energy). I think 2.5 $\sim$ 5Ry is reasonable choice. But in cases, small `emax_sigm` can give poor dispersion curve (slightly unnatural behavior) because of sudden cutoff by emax_sigm. However, we like to use smaller value to reduce computational time. That is, larger is better, but expensive. Generally speaking, accuracy less than 0.1eV (for bandgap) is allowance of current method. 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 dangerous 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 is quadratically larger

(become rough). 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.

- 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 specify 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.
- `<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).

  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 defaults, 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 algorithms. 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...)

4.2 do QSGW calculation

Let us perform QSGW calculation. For 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 calculates 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 view.

This is an example of one iteration of QSGW cycle. (now a little different but essentially 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 begging 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 minims $V_{\rm xc}^{\rm LDA}$. What we did is the one-shot GW from LDA result; but note that we calculate not only diagonal elements 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
```

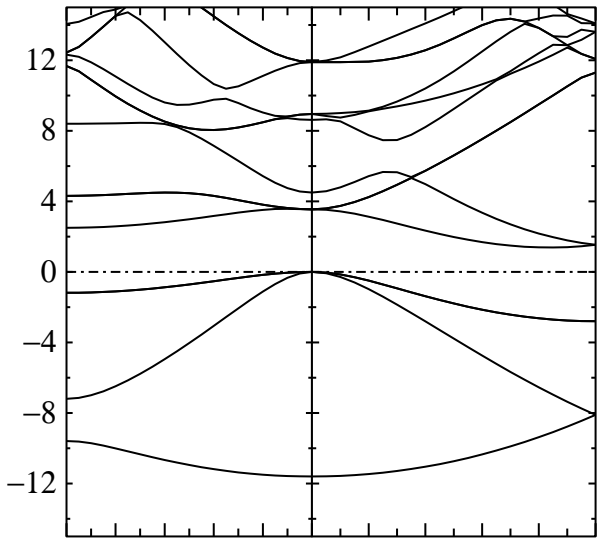

Figure 3: Si, one-shot GW with off-diagonal elements

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.)

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
```

```
 ...(skeip here) ...

         mpirun -np 2 /home/takao/bin/hsfp0_sc  < _IN_ > lsc
OK
         /home/takao/bin/hqpe_sc  < _IN_ > lqpe
OK
5 iteration over
   mpirun -np 2 /home/takao/bin/lmf-MPIK  si > llmf_gwscend.5
```

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. Information 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 beginning.

In order to check convergence 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 around 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.)

5 Appendix

5.1 appendix 1; How to add spin-orbit coupling

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

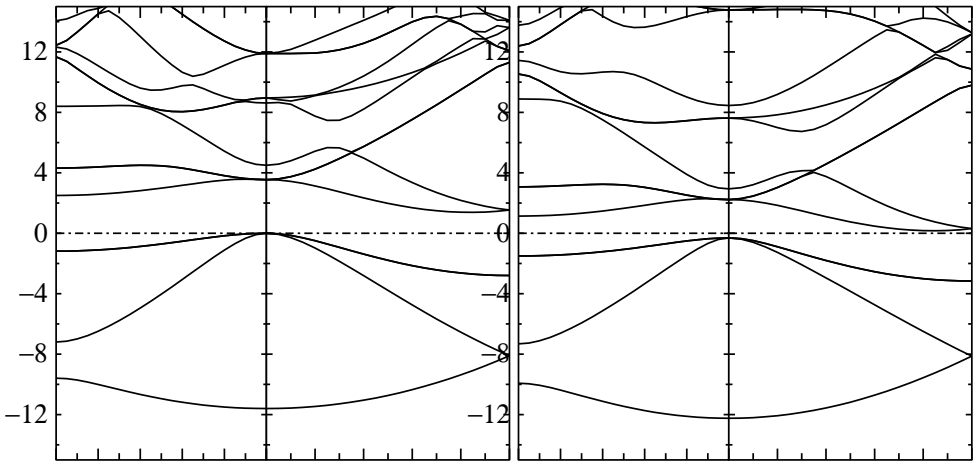

Figure 4: band plot(Si, QSGW one-shot test) and band(Si) (GGA)

```
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
```

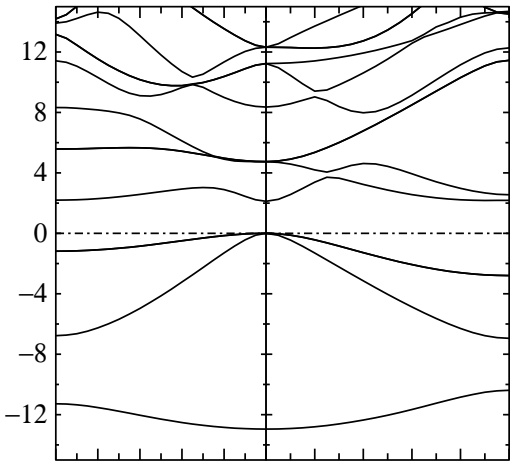

Figure 5: band(GaAs), QSGW (test case)

```
-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.
```

5.2 appendix 2: How to set local orbitals

```
As we stated, do "lmfa |grep conf" to check used MTO basis.

We have to set SPEC_ATOM_PZ=?,?,?
(they ordered as PZ=s,p,d,f,... ) to set local orbitals.

lmv7 (originally due to ASA in Stuttgart) uses a special terminology
"continious principle quantum number for each l", which is just
relatated to the logalismic derivative of radial funcitons at MT
boundary. It is defined as
 P= principleQuantumNumber + 0.5-1/pi*atan(r* 1/phi dphi/dr),
where phi is the radial function for each l.   For example,
 P= n.5 for l=0 of free electron (flat potential) because phi=r^0,
 P= n.25 for l=1 because phi=r^1;
 P= n.147584 for l=2 because phi=r^2; P=, n.102416 n.077979 for l=3,4.
(Integer part can be changed). See Logarithmic Derivative Parameters in
http://titus.phy.qub.ac.uk/packages/LMTO/lmto.html#section2

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.

  Examples of choice:
  Ga p: in this case, choice 0 or choice 2 is recommended.
      We usually use lo for semi-core, or virtually unoccupied level.

     (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,4p,5p.
     (1)3p lo ---> 4p val (when 3p is treated as valence)
        3d semi core, 4d valence
        Set PZ=0,3.9
        (P is not requied to set. *.9 for core like state. It is just an initial condition.)
     (2)5p lo ---> 4p val (PZ>P)
        Set PZ=0,5.5
        5.5 is just simply given by a guess (no method have yet
implemented for
        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.

     (3xxx)4p lo ---> 5p val (we don't use this usually. this is for test purpose)
        4p lo, 5p valence
        Set PZ=0,4.5 P=0,5.5 (In this case, set P= simultaneously).
        (NOTE: zero for s channel is to use defalut numbers for s)

  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 ---> 4d val  (when 3d is treated as valence)
         Set PZ=0,0,3.9  (P is not required to set)
     (2) 5d lo ---> 4d val  (PZ>P)
         Set PZ=0,0,5.5
     (3xxx) 4d lo ---> 5d val  (this is for test purpose)
         Set PZ=0,0,5.5 P=0,0,4.5
         (NOTE: zero  for s,p are to use defalut numbers )

   If you like to read from atm.ga file instead of rst file(if exist).
   You have to do lmf --rs=1,1,0,0,1, for example. See lmf --help
   Becase rst file keeps the setting of MTO, thus change in ctrl is not
   reflected without the above option to lmf.
============================================================
```

6 MEMO random

These are memo randoms. I have to explain them.
xxx under construction xxxxxxxxxxxxxxx

```
== not meaningful total energy ===
 Total energy shown in QSGW mode in current version is not meaningul.
 (just treat as an indicator to convergence).

== Do we use vwn or gga for QSGW? ===
  In prinicple, QSGW results should not depends on vwn or gga
  (XCFUN=1 or 103 in ctrl). But there is minor dependence, because
   1. frozen core density.
   2. core eigenfunction.
   3. radial basis funcitons
   4. Slight numerical reason
      (This is probably because Sigma-interpolation procedure
       But not exactly figured out yet
       -->affect about 0.02eV as for band gap for GaAs. ).
  In anyway, use vwn (HAM_XCFUN=1) as standard.
  And such technical things affects, 0.05 eV level of error for band gap.

== one show QSGW ==
  one-shot QSGW may be useful in cases.
  As it contains off-diagonal part, we can resolve band tanglement
  problem in Ge (no band gap).


== Restart calculation in lda ==
 lmf(lmf-MPIK) read rst.* in defaluts.
 rst contains electron density.
 If rst is already converged, it stops after two iteration.
 rst contains atomic positions.
 So, in order to read atomic positions change in ctrl,
 Use options shown in lmf --help.

== Restart calculation in qsgw ==
  To remove mixsigm* (mixing for sigm), maybe required.

== iteration check ===
  First, watch console output of gwsc (do redirect to output file)
  Need to check OK! signs arrayed on 1st columns.

  gwsc iteration is time cosuming,
  So we need to check calculations are normally going on or not.
```

```
  Memory inefficiency.
  Set 'KeepEigen off' an 'KeepPPOVL' off.
  In fact, out code is still inefficient for memory usage.

  grep gap llmf ---> minimum gap at mesh point.
  see save.* ,or grep '[xc] ' save.*
  the end of iteration of lmf is shown as x or c.
  (if failed, QPU file.
  dqpu QPU.4run QPU.3run
  As for usual semi-conductor, accuracy abou t0.1 eV is limit of current implementation.
  Set vwn (xcfun=1) looks better (stable) for GW.

  $grep rms lqpe*
  shows
           ...  rmsdel=2.44D-04
           ...  rmsdel=4.91D-03
           ...   rmsdel=2.44D-04
           ...   rmsdel=3.37D-04
  If rsmdel is getteing to be smaller, it is on convergence path.
  (but in magnetic cases, it may give be too good even not yet going to
be converged..., beccause magnetic energy is so small)

  grep diffe llmf  ---> difference of energies of each iteration.

  ehf (harris energy)
  ehk (Hohenberg kohn energy)

== emax cutoff for APWs. ==
  We can not use so many APWs in current version,
  because of overcompleteness (this is because null vector within MTs),
  In anyway, use pwemax=3 as standard (test it with 4 or 5).
  To avoid failure of calculation, we may use smaller MT radius for
  alkali, and alkali-earth elements.

== Check 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. Furthermore, 3d+4d as valence is better.
  Thus we need to set PZ=0,3.9,4.5
  I also got aware that emax_sigm should be large enough (4$\sim$5 Ry)
  to have smooth band dispersion. n1n2n3 can be 10x10x10.

== 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.

== QOP check
   In cases, it is better to use QOPchoice=2 instead of default QOPchoice=1.
   (For slabs, QOPchoice=2 may be better; need check more. In anyway,
    it is problematic to use unbalanced k points for anisotropic cell).

== 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.


=== LDA+U ===
not yet...
sample

=== MAE by rotating crystal ===
(we have a sample at

=== spin wave ===
J calculation.


xxxxxxxxxx
At line 663 of file slatsm/fopna.F (unit = 13, file = '')
Fortran runtime error: File 'vessm.gas' already exists
--> mpirun -np 12 lmf ? (you have to use lmf-MPIK..)

xxxxxxxxxxxxxxxxxxxxxxxxx
If not stable convergence in gwsc, try to set
mixbeta 0.5
(and/or mixpriorit 3 or something)
at the begining of sigma.


=======
cleargw (directory):
This command clean up up intermediate files under (directory).
This recursively into deeper level. Be careful, or edit it.
I use it as '>cleargw .'


------------------------------------------------------
 QSGW: convergence check sheet.
------------------------------------------------------
1. Basis to expand eigenfuncitons.
   As for APW, try pwemax=3, 4, 5.
   In principle, bigger is better.
   Local orbitals for semicore were requied (for Fe, and so on).
```

```
2. Re-expand eigenfuncions (QpGcut_psi for IPW part)
   In principle, bigger is better.
3. Mixed product basis.  QpGcut_cou for IPW part.
   <ProductBasis> section for PB part.
4. number of k points, QOPchoice(irrelevant but speed up).

5. omg,dw  (bins to accumulate imaginary part).

6. emax_sigm (use 2Ry to 6Ry. And see stability. In priniciple, bigger is better.)

7. Do GW with XCFUN=1 or 103 (VWN or GGA)?
   In priniple, bigger emax_sigm reduce dependence on them.
   But, not easy. We may take the difference as allowance of error.

-----------------------------------
Magnetic moment within MTs are shown as
-----------------------------------
 charges:       old           new
 smooth      17.240314     17.240740   ...
 mmom         0.000024     -0.000010
 site    1    6.207135      6.206590
 mmom         1.062276      1.062991  <--- here
 site    2    6.207115      6.206834
 mmom        -1.062323     -1.062958  <--- here
 site    3    1.172718      1.172918
 mmom         0.000011     -0.000011
 site    4    1.172718      1.172918
 mmom         0.000011     -0.000011
In this case, MTsite1 has 1.062991 and MTsite2 has -1.062958.
>grep 'lin mix' -A30 llmf
can take out this message (if console output is in llmf).


------------------------------
ORBITAL MOMENT in pertubation:
------------------------------
Try
>lmf nio --rs=1,0 -vso=1 --quit=band >llmf
After converged, try
>grep IORBTM -A20 llmf
Then llmf shows shows orbital moment in first order perturbation.
(Here --rs=1,0 read rst.* file but not change it. See >lmf --help.
--quit=band means quit just after band calculation.)
```