ecalj manual

<https://github.com/tkotani/ecalj>

**Reference**

Jan2017 Add minimum description for syml generator at Sec.[4.7.1](#sec:getsyml)  
 Aug2019 slight modification for CMD35

# Introduction

The ecalj  is an first-principles electronic-structure calculations package with some unique features. With ecalj , we can do not only standard calculations (LDA/GGA/LDA+U, relaxation of atomic positions), but also the quasiparticle self-consistent GW calculations(QSGW), linear responses (charge and spin), Wannier functions (and U of them).

This is base on an unique one-body problem solver, the PMT method (=the Linearized APW+MTO method) . Thus we identify the QSGW method implemented in ecalj as the PMT-QSGW method. Introduction to the PMT-QSGW is given in Sec.[1.1](#sec:fecalj). Today “QSGW” is accepted as a standard procedure in the electronic structure calculations .

First, see README.org shown at <https://github.com/tkotani/ecalj#ecalj-> (or ecalj/README.org in the package). Free to download ecalj  package from it, and use it. The QSGW code is version controlled by git.

The ecalj  is related to a FP-LMTO package lmv7 seen at ecalj/Document/Manual/CategoryAndToken.org (or html).  
The lmv7 and ecalj are branched at year 2009. After branched, we added new features: simple install and test; all codes are in f90 (no C compiler); new methods, especially PMT-QSGW; MPI parallelization for QSGW; simple usage with automatic setting of default files by python ver2; a small tool to convert VASP POSCASR to ecalj, and so on. PMT-QSGW shows more stable convergence than the previous version, FP-LMTO-GW [[1]](#footnote-23).

 ——————–  
  package mainly consists of two parts. One is (in ecalj/lm7K/), the other is (in ecalj/fpgw/).  
, based on   
We can perform standard calculations such as LDA,GGA,LDA+U, atomic position relaxation, and so on. In addition, the PMT part has an interface to perform GW (and QSGW) calculation: the one-body part can include a given non-local exchange-correlation potential stored in a file sigm.\*. [[2]](#footnote-24). The QSGW calculation is performed by a script gwsc, which has an iteration loop calling the one-body program (lmf) and many-body part (GW part) alternatively. The many-body part generate the file sigm.\*. See Fig.[[gwscpicture]](#gwscpicture) and around.  
, based on   
As the inputs for the calculation, we have to supply the eigenfunctions and the eigenvalues from the one-body part to the GW part. The eigenfuncions re-expanded by the two types of basis functions, the atomic-like argumentation functions in the muffin-tin(MT) spheres, and the plane-waves in the interstitial region, say, the interstitial plane-wave (IPW) hereafter. IPW is defined as the usual plane waves in the interstitial region, but zero within MTs’. The IPWs + ”atomic like functions within MTs” make “a basis set to expand eigenfuncitons”. See Eq.(17) of .

We need another basis set to expand “product of eigenfuncions”. That is, the mixed product bases (MPB)  
, which consists of the two kinds of bases (caution: do not mixed up with the basis set for eigenfuncitons); (i)the local atom-centered functions confined to MT spheres, so-called the product basis; (ii) IPW. The product-basis are calculated from products of solutions to the Schrödinger equation within the MT sphere. The Coulomb matrix , the dynamically screened Coulomb interaction , and so on, are expanded in the MPB. It can virtually span all the space made of product of eigenfunctions (but, in practical calculations, we need to use a small size of the bases to reduce computational time). We include full energy-dependence of . See Sec.3 of .

Recently T.Kotani includes the Wannier function generator, which was originally developed by T.Miyake and H.Kino on top of previous version of GW part. Thus the Wannier functions (including effective interactions) can be generated in the PMT-QSGW. (U parameters in the full-screening and cRPA).

## Uniqueness of the ecalj package.

We will explain two unique points of ecalj .  
**PMT**:  
Central part in an electronic structure packages is one-body problem solver. It means how to calculate eigenvalues/eigenfunctions for a given one-body potential. Inversely, we have to generate new one-body potential for given eigenfunctions/eigenvalues based on the density functional theory (DFT) in the LDA or GGA (In the followings, 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 states. Then we can calculate atomic forces by perturbation. Based on such an one-body problem solver, we can implement kinds of methods; e.g, dielectric function, magnetic susceptibility, transport and so on. Furthermore, we can implement higher-level approximations such as the QSGW method explained below. An one-body problem solver (in 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 ecalj , we use the PMT method as the one-body problem solver. The PMT method is a new all-electron full potential method. It uses not only the augmented plane waves (APW) but also the muffin-tin orbitals (MTO) together, in addition to the local orbital (lo’s), to represent the eigenfunctions (no other methods use two kinds of augmented waves together). Thus eigenfunctions are expanded in the linear combinations of the APWs, MTOs, and the lo’s. The formulation is clarified in Ref.. Then the electron density and the one-body potential are given in the “3-components representation”. That is, the electron density (one-body potential) is divided into three components,  
   “smooth part onsite muffin-tin (MT) part counter part”.  
Here the counter part is in order to remove smooth part within MTs. This formalism (Soler-Williams formalism ) is also used in the projected augmented wave (PAW) method such as VASP.

We now usually use highly localized MTOs together with APWs of low energy cutoff ( Ry). [[3]](#footnote-26) I think this is promising not only for efficient DFT/QSGW scheme, but also for kinds of applications in future.

:  
In ecalj , we can perform the GW calculation. The usual GW approximation is so-called “one-shot GW” starting from LDA. It usually only calculates differences between the quasiparticle energies (QPEs) and the LDA eigenvalues by a perturbation (only diagonal part of self energy for the LDA eigenfunctions). Its ability is limited; it may fail when its starting point (eigenfunctions and eigenvalues supplied by LDA) is problematic. This is the reason why we originally develop the QSGW method. The QSGW now becomes popular and taken as a possible candidate to go beyond current limitation of such GW and LDA/GGA . In principle, results given by QSGW do not depend on LDA anymore; the LDA are only used to prepare initial condition for self-consistency iteration cycle of the QSGW calculation [[4]](#footnote-27).

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 1.69 eV [[5]](#footnote-28). On the other hand, the QSGW predicts about 1.8 1.9eV, a few tenth of eV larger than experiment (for practical use, we sometimes use “hybrid functional between QSGW and LDA” so as to obtain smaller band gap). Even in the case of NiO and so on, the QSGW gives reasonable results (there is a tendency to give a little larger band gaps than experiments). This is in contrast to the case of the one-shot GW applied to NiO, where we can not have good agreement with experiments because the stating points in LDA is problematic.

The ecalj  have other functions. LDA+U, atomic forces and relaxation (in GGA/LDA), core level spectroscopy and so on. In addition, we can calculate dielectric functions and magnetic responses from QPEs and the quasiparticle eigenfunctions given by LDA/QSGW. But total energy in QSGW is still in research (shown total energies in QSGW calculations are dummy now).

The QSGW calculations are very time-consuming; roughly speaking, it takes 10 or more times expensive than usual one-shot GW (although we can reduce computational time by choosing computational conditions). Thus the size of systems which we can treat is limited to ten atom in a cell or something, say, with a node of 16 cores; computation may require a week or so to have reasonable convergence. (heavy atoms require longer computational efforts, light atoms faster; non-magnetic systems are easier. We still have much room to accelerate the method, but not have done yet so much. Minimum MPI parallelization is implemented). The computational effort is in the most time-consuming part of QSGW.

### What do we expect for QSGW?

Let us recall hybrid functionals such as B3LYP, and LDA+U. In hybrid functional methods, we use Vxc = (1-)\*LDA+\*(Fock exchange like term), where is taken to be usually [[6]](#footnote-30). The can be dependent on materials; for metals should be almost zero. For larger band gap insulator, becomes larger. [[7]](#footnote-31) Despite of success of the functional, its ability is limited. For example, it is known that a hybrid functionals fail to describe metals such as bcc Fe. On the other hand, we have LDA+U method which succeeded to describe materials including localized electrons. However, it contains kinds of ambiguity and U is chosen by hand. The important part of the hybrid functional methods and LDA+U is the non-local potential. It is missing in the DFT. As we discussed above, they give some success but not satisfactory. We somehow need to have a method to determine high-quality non-local potential (a substitution of the exchange-correlation potential in LDA). It is the QSGW method.

Note two important aspects of non-local potential (missing character in the local potential used in DFT). One is the onsite non-locality; it is also taken into account by LDA+U model. However, note that relative shift of O(2p) band with respect to the center of 3d band is not in LDA+U. The other is the off-site non-locality (mainly between nearest neighbors), which may relate to LUMO-HOMO gap. A non-local potential can behave a projector which push down only the HOMO states (valence band) to lower energy. This can be in the hybrid functional but not in LDA+U.

In the QSGW, we determine such a non-local potential with the calculation of the GW method, in a self-consistent manner (we repeat GW calculations until converged). We can expect QSGW much more than hybrid methods/LDA+U. Roughly speaking, because the QSGW automatically determine U of LDA+U, or alpha of the hybrid functionals. More accurately speaking, we determine not only but also (the screened Coulomb interaction) self-consistently. Here corresponds to U and alpha. Thus QSGW gives reasonable results even if it is applied to metals such as Fe. For systems with metallic screening, it gives small non-locality (results are close to those of LDA). For systems with large band gap, QSGW gives large enough non-locality (like 0.25\*(Fock exchange)).

Since we now need to treat complex systems, e.g, metal on insulator, it is very essential to treat kinds of materials on a same footing.  
The main purpose of QSGW is to determine an one-particle effective Hamiltonian [[8]](#footnote-32), which describes the quasiparticle picture (or independent-particle picture) for the system we calculate. In other words, QSGW divides the full many-body Hamiltonian into . The screened Coulomb interaction is determined self-consistently in the QSGW iteration cycle.  
In comparison with LDA, we see differences;

* Band gap. QSGW tends to give slightly larger than experiments. It looks systematic.
* Band width. Usually, sp bands are enlarged (except very low density case such as Na). This is the case for homogeneous electron gas. As for localized bands like 3d electrons, they can be narrowed.
* Relative position of bands. e.g. O(2p) v.s. Ni(3d). More localized bands tends to get more deeper. Exchange splitting between up and down (like LDA+U) get larger. In cases such as NiO, magnetic moment become larger; closer to experimental values.
* Hybridization of 3d bands with others. QSGW tends to make eigenfunctions localized.

However, reality is complexed, and not so simple in cases.

## Rule in this manual

* This font is for executable file(program) or shell scripts.
* echo 3hbasfp0 means doing hbasfp0 with the argument ’3’ supplied as the standard input (read(\*,\*) in fortran).
* This font is for files, directories, contents of files, or variables used in codes.
* ctrl.si,rst.si and so on mean the case of Si. You may need to replace the extension si for your case. (this extension is given by user. Lower case, number, and underscore are allowed.) In the followings, ctrl.\* means a file wish such an extension.
* There are files named foobarU and foobarD, which are for up spin (isp=1) and for down spin(isp=2), respectively; for example, SEXU and SEXD. We sometimes use foobarU to denote foobarU and foobarD together.
* **k** vector in the Brillouin zone is called as **q** or **k**.

## What can we do with the ecalj package?

At Feb.2015, what we can do is as follow. We have limited parallelization. (e.g. k point parallel).

* LDA/GGA LDA+U, calculations, atomic forces and relaxation. Spin-orbit is included only for co-linear spin-density cases.
* Quasi-particle(QP) energy in the 1st-iteration from LDA. (one-shot )
* Make band plot for LDA and the QP energies.
* Spectrum function of the self-energy . Life time (imaginary part) of QPs.
* Dielectric function, and its inverse. (including local-field effect or not).
* QP self-consistent (QSGW)
* magnetic susceptibility.
* Wannier function. (not only one-body part, but also effective interaction and cRPA)

# Install

Install and minimum tests are easy; even in a note PC, e.g., we can use gfortran in Ubuntu. In Thinkpad T480s (2018) install may take five minutes, and all tests may take another fifteen minutes.

For productive runs, we may need multi-cores. Current implementation for parallelization by MPI is limited (not so much especially for the dielectric function part yet). Thus, probably, it may be not so efficient to use too many cores.

Follow the instruction of ecalj/README.org.  
or we can see the same one at <https://github.com/tkotani/ecalj/README.org>. We have a command ecalj/InsallAll.ifort and so on. This command installs ecalj and run a series of install tests automatically.

## Binaries and Scripts

Main binaries and Scripts contained in ecalj are

* ctrlgenM1.py  
  Generate default input file ctrl.\*from the structure file ctrls.\*. The latter file only contains information of crystal structure.
* lmfa  
  Spherical atom calculation as initial condition, and core charge.
* lmchk  
  Check atomic positions, crystal symmetry, and computational conditions. Check ID of MuffinTins (MTs of atoms specified by ctrl file is re-ordered by computer.
* lmf and lmf-MPIK LDA/GGA,LDA+U calculations. (or we can use Vxc in QSGW instead). We mainly use lmf-MPI (${\bf k}$-parallel version) instead of lmf.
* PROCAR mode of lmf: Fat band mode.  
  mpirun -np 4 lmf-MPIK --mkprocar --band:fn=syml mgo gives PROCAR  
  (Try an example  /ecalj/MATERIALS/MgO\_PROCAR/. Run ./job at the directory).
* gwsc  
  QSGW calculation
* job\_band,job\_tdos,job\_pdos  
  band, fermi surface, tdos, pdos plot.
* job\_fermisurface  
  Run job\_fermisurface cu -np 4 10 10 10. This write down all band energies (around Ef) for 10x10x10 in BZ. Then we can view it with xcrysden as xcrysden --bxsf fermiup.bxsf (usage of xcrysden need to be written here). With –allband added for job\_fermisurface, we can write all the bands. So, we need this option if you like to write iso-energy surface at high (or low) energy.
* epsPP\_lmfh  
  Dielectric funciton without local field correction (LFC).
* eps\_lmfh  
  Dielectric funciton with LFC
* epsPP\_lmfh\_chipm  
  Non-interacting transverse spin polarization.
* gw\_lmfh  
  One-shot GW calculation. This also show life-time of QPs (QPU\_lmf). (we need make it parallelized...)
* genMLWF  
  Wannier functions and matrix elements of on it. A implementation of cRPA included.
* dqpu  
  A small python script to compare QPU.\* files (eigenvalues are compared) numerically.

## tests

Install.ifort run tests at ecalj/TestInstall.  
In the following, si:gw\_lmfh means ’make si\_gw\_lmfh’ at ecalj/TestInstall/; this test is performed with the Makefile at the directory.

si:gw\_lmfh/ Results: QPU   
si:gwsc/ : QPU,log.si  
gas:gwsc/ : QPU,log.gas  
nio:gwsc/ : QPU,log.nio  
fe\_epsPP\_lmfh\_chipm: : ChiPM\*   
gas:eps\_lmfh/ : EPS\*  
gas:epsPP\_lmfh/ : EPS\*

(These are just samples; not for practical calculations)

## Directory structure

ecalj  
├── InstallAll.ifort, Install.gfortran ! install and test  
├── Document/  
├── fpgw/ ! full potential GW code  
├── fpgw/Wannier ! MaxLoc Wannier and cRPA code.  
├── lm7K/ ! PMT method   
├── MATERIALS/ !job\_materials.py contains examples.   
├── StructureTool/ ! POSCAR converter, and structure tool  
├── GetSyml/ ! Brillouin zone and symmetry lines generator and visualizer.  
├── TestInstall/ ! Install test; this is invoked from Install.\*  
└── TOOLS/ !Tools for developers

# Theory (note)

Except the technical details, we need to know minimum for these theories.

* DFT in LDA/GGA.
* GW
* QSGW

There are literatures as for GW. A recent one is Ref.. In addition, it is better to know the basics of the PMT method (LAPW+LMTO method) . Here is a small note for .  
  
The Green’s function $G({\bf r},{\bf r}',\omega)$ is the central quantity in the GW calculation. In the one-particle theory (mean-field theory/non-interacting case), the Schrödinger eq. is

$$\begin{aligned}
(i \frac{\partial }{\partial t} - H\_0) \psi({\bf r},t)=0 \label{sch0}\end{aligned}$$

Here is the one-particle Hamiltonian which contains electrostatic potential plus exchange-correlation potential $V\_{\rm xc}$ (here we don’t care how it is given. This should be static but can be non-local as Fock term). When we have (unkown) source term $J({\bf r},t)$ instead of 0 in the right hand side of , we can calcuate $\psi({\bf r},t)$ by multipling inversion of the operator . This is the (non-interacting) Green funciton $G({\bf r},{\bf r}',\omega)$ defined as

$$\begin{aligned}
(i \frac{\partial }{\partial t} - H\_0) G
= \delta(t-t')\delta({\bf r}-{\bf r}'). \label{g0def}\end{aligned}$$

As this shows, is the inverse matrix of the operator , (we pay attention to the boundary condition, especially for time-direction: retarted, advanced or time-ordered). Thus we can write . In other words, the Green function is the integration kernel in order to convert a differential equation to an integral equation. This is the same as the case of Poisson equation (Laplacian) for the electrostatic problem;then $1/|{\bf r}-{\bf r}'|$ is the Green function for the conversion (then we calculate electro static field $\phi({\bf r})$ for give source term $\rho({\bf r})$.

In the case of electro static problem, we have two ways to solve the problem. One is the direct method to solve differential equation, the other is using such integration kernel. They are equivalent, but the latter is easy to handle and suitable for numerical calculations (under the assumption of “superposition low”, that is, linear response). Although $G({\bf r},{\bf r}', t-t')$ contains time variables (to describe wave propagation), essentials are the same.  
  
$G({\bf r},{\bf r}',\omega)$ is the quantity which is well defined even in the many-body perturbation theory. It is defined as the expectation value as $\langle 0|V{\psi}({\bf r}t) V{\psi}^\dagger({\bf r}' t')| 0 \rangle$ (this is not accurate. See literatures..). Here we use the second-quantized field-operator . This definition of is a natural extension of the one-particle case.

However, there are differences in the of one-particle case and many-body case. In the one-particle case, we have no interaction between electrons. Thus an electron added to the ground state (which is by filling electrons up the Fermi energy) is the eigenstate. An electron going on can not be an eigenstate, because of the correlation effect and exchange effect.

Let me explain the correlation effect. In contrast to this, one-particle state (=the quasiparticle(QP)) in the many-body theory can be not the eigenstate; it moves in the sea of ohter electrons and holes, which can be excited to other states, or an moving electron can move with polalized cloud of other electrons and holes. Its motion is affected by oher electrons. It will lose energy gradually; the QP can have life time. This may be identified as the correlation effect.

Let me explain the exchange effect. This is due to the Fermi statistics. The electron, which you focus on, can not be distinguished from other electrons. (Thus mu-on can not feel this effect. for mu-on in solids only contains pure correlation effect). From a point of view, this exchange effect is interpreted as a hopping effect, sudden jump from an electron to another elecron (actually the Hartree-Fock theory gives zero effective mass at the Fermi energy for metallic systems).

We can include both of the effects in the GW calculation at the lowest order. Exactly speaking, these two effects are really mixed up. However, in GW (since GW is at the lowest order), these two are clearly separated.

The rigorous equation for is

$$\begin{aligned}
\left(i \frac{\partial }{\partial t} - H\_0 +
(\Sigma({\bf r},{\bf r}',t-t')-V\_{\rm xc})\right) G
= \delta(t-t')\delta({\bf r}-{\bf r}'). \label{gdef}\end{aligned}$$

Only the difference is adding $\Sigma({\bf r},{\bf r}',t-t')-V\_{\rm xc}$. In other words, all unknown effects are pushed into this term. Many body effect are pushed into (downfolded or projected into) the self-energy (=dynamical one-particle potential) $\Sigma({\bf r},{\bf r}',t-t')$. We don’t treat many-body quantities such as $X({\bf r}\_1 t\_1, {\bf r}\_2 t\_2, {\bf r}\_3 t\_3)$ directly in the calculations.

The method is how to give the $\Sigma({\bf r},{\bf r}',t-t')$ in the lowest order. Because of the long-range property of the Coulomb interaction, we need to have a special technique (not a simple perturbation). $\Sigma({\bf r},{\bf r}',t-t')$ is calculated as the product of , where is (usually) just the non-interacting Green function given in . is the dynamically screened Coulomb interaction calculated in RPA.  
  
Note that completely cancels the effect of $V\_{\rm xc}({\bf r},{\bf r}')$ since it is included in . Thus this division is not meaningful if we can solve a problem completely. However, what we can do is only the perturbation. Quantities for is taken as the basic quantities. On it, we apply the perturbation. Equivalently we can have a division , where is the many-body Hamiltonian.

In priniple, we can start from any $V\_{\rm xc}$. However, as long as we use perturbation theory, we have to minimize the effect of $\Sigma({\bf r},{\bf r}',t-t')-V\_{\rm xc}$. The degree of freedom of the choice of $V\_{\rm xc}$ is used to minimize it. This is the self-consistent pertubation theory idea for QSGW.  
If $\Sigma({\bf r},{\bf r}',t-t')-V\_{\rm xc}$ give small effect, we can use the concept of the QP (independent particle picture) as the basis to evaluate physical quantities.

We have to determine best (or optimum) $V\_{\rm xc}$. How to do it? An idea is that we should determine it so that the size of $\Sigma({\bf r},{\bf r}',t-t')-V\_{\rm xc}$ should be smallest (how to measure the size?). Or, based on the Landau-Silin’s theory, the QPs contained in should be reproduced by . In anyway, this ends up with the self-consitent petrurbation theory. For trial $V\_{\rm xc}$ (and electron density), we have ; from we can calculate $\Sigma({\bf r},{\bf r}',t-t')$. Then we have to determine new $V\_{\rm xc}$ in a method. This is repeated until converged. No unique choice for the method. Thus we testes some possible ways and now we usually use a standard procedure as described in Ref. .

# LDA/GGA calculations and Plots

Calculations are performed by following steps. These steps are detailed in the following sub-sections.

To identify a set of files used for a material we calculate, we use an extension to files. For example, files explained below are with extensions (only lower case allowed) of materials. For example, ctrls.cu and ctrl.cu. In this case cu is the extension. Any extension works. Other possible examples are ctrls.lagao3, ctrl.wgantest1, and so on.

1. Write crystal structure file ctrls.\*, 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.\* (See ecalj/StructureTool/README.txt for the tool. we have vasp2ctrl and ctrl2vasp. Type them without arguments to see help.)
2. Generate ctrl.\*from ctrls.\*by a script ctrlgenM1.py, [[9]](#footnote-42)

* Here ctrl.\* is the main control file which contains all required information to perform calculations. ctrl.\* contains not only the content of ctrls.\*, but also other information needed for calculations. If necessary, we edit the generated ctrl.\* file before next step.

1. Check crystal structure. lmchkis to confirm the crystal structure (space-group symmetry and so on). lmchkis applied not to ctrls.\*but to ctrl.\*. ctrls.\*never used in the following steps.

* : If a crystal structure is only slightly different from a structure with higher symmetry, the lmchkmay give a wrong crystal symmetry. In such a case, you have to “standardize structure” by VESTA or some other tools. This occurs e.g., when we use a structure numerically relaxed by VASP.

1. Run lmfa (calculations of spherical atoms (MT sites) in the cell). It also calculates core eigenfunctions and valence electron charge to set up initial condition. Then we run main calculation of LDA by lmf. It repeats iterations, and end up with converged results in LDA. Main result (electron density satisfying self-consistency) is stored in restart file rst.\* (binary file). It finished within a second.
2. Run LDA/GGA calculations. We can run the LDA/GGA calculation by lmf or lmf-MPIK (-MPIK means kpoint parallel version).
3. Post processing.  
   Plot energy band, DOS, PDOS, by running scripts. We can use scripts job\_band for band plot (need syml.si file (symmetry line for band plot)). We also have job\_pdos, job\_tdos, job\_fermi and so on for DOS, PDOS, fermi surfaces. Since we use gnuplot to plot them, meanings of obtained data is apparently clear.

## Write crystal structure file, ctrls

Let me show some samples of crystal structure files ctrls.\*.

* ~/ecalj/lm7K/TESTsamples/Cu/ctrls.cu  
  ------from here ------------------  
  % 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  
  ------to here ------------------
* ecalj/lm7K/TESTsamples/GaAs/ctrl.gaas  
  ------from here ------------------  
  #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  
  ------to here ------------------
* ecalj/lm7K/TESTsamples/SrTiO3/ctrls.srtio3   
  ------from here ------------------  
  %const da=0 au=0.529177  
  %const d0=1.95/au a0=2\*d0 v=a0^3 a1=v^(1/3)  
  HEADER SrTiO3 cubic   
  STRUC ALAT={a1} DALAT={da}   
   PLAT=1 0 0 0 1 0 0 0 1  
  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  
  ------to here ------------------

Lines starting from ’#’ are neglected as comment lines. Lines starting from ’% const’ define variables and set values (in these cases, da, alat, and bohr, and so on). 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 ctrls.\*. [[10]](#footnote-44)

Note that we have two tags of “categories” "STRUC" and "SITE". (“HEADER” tag is also; but it is just for user’s memo shown in console output). These tags should start from the first column. Thus ctrls.\*is divided into multiple “categories”. In a category, we have “tokens” such as ALAT, DLAT, PLAT. These under STRUC category. ALAT+DALAT specify unit of length in this ctrl file. These are in a.u. (= bohr radius=0.529177Å).

The unit cell is given by PLAT (as noted, ALAT+DALAT 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 number of 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. [[11]](#footnote-45) 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 (in the unit of ALAT+DALAT).

To test ecalj, 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. We need extension ’.vasp’ to recognize it is written in 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 possible way is using cif2cell.

If you have a cif file, run

cif2cell foobar.cif -p vasp --vasp-cartesian --vasp-format=5

And convert POSCAR to ctrls. cif2cell is available from github.

In ctrls.srtio3, we use an expression 1/2 to give POS. We can use mathematical expression instead of values. Mathematical expressions such as “ sqrt(...)” are recognized. (instead of 3\*\*2, use 3^2. Use parenthesis, and no space for an expression). 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. Such a procedure will be explained in other place.xxx)  
This is an example for Antiferro NiO:

#id = NiO  
%const bohr=0.529177 a=7.88  
STRUC ALAT={a} PLAT= 0.5 0.5 1.0 0.5 1.0 0.5 1.0 0.5 0.5  
SITE ATOM=Niup POS= .0 .0 .0  
 ATOM=Nidn POS= 1.0 1.0 1.0  
 ATOM=O POS= .5 .5 .5  
 ATOM=O POS= 1.5 1.5 1.5  
SPEC  
 ATOM=Niup Z=28 MMOM=0 0 1.2 0  
 ATOM=Nidn Z=28 MMOM=0 0 -1.2 0  
 ATOM=O Z=8 MMOM=0 0 0 0

In this case, we define Niup and Nidn sites. These are recognized as Ni atom because of given Z number in SPEC. The subtoken MMOM=Ms,Mp,Md,Mf... re to specify number of magnetic moments () for s,p,d,f channels (difference of up - down electrons within MT sites) as initial condition. In this case, we set n(up)-n(down)=1.2 for Niup site for d channel. Even just one ATOM name is given by yourself, all ATOM in SPEC should be given (in this case SPEC for O should be given).

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).

It is possible to add RELAX= 0 0 1 after SITE\_ATOM\_POS; this means structure relaxation along z-axis (also need to set DYN category given in Document/Manual/CategoryAndToken.org). but its defaults are given (but commented out) automatically in the ctrl file generated by the procedure described in the following section). We detail it in other place xxx.  
After ctrl.\* is generated as shown below, we can run a command lmchk to check weather crystal structure is correctly given or not. It finish in a second. It show symmetry information, and so on used in the calculation.

: Positions of atoms are not necessarily fixed by ctrl.\* when you restart calculation with rst.\* file, because atomic positions are read from rst.\*. We need to pay attention when we use DYN option because lmf run may save relaxed atomic positions into the rst.\*. As “lmf –help” shows, lmf si --rs=1,1,1,0,0 can read atomic position from ctrl file.

## Generate default ctrl from ctrls by ctrlgenM1.py

To run programs of lm7K (lmfa, lmf, lmchk) in ecalj, we need an input file ctrl.\*, which contains not only structures but also other settings. To generate ctrl.\* from ctrls.\*, we have a command "ctrlgenM1.py" (written in python 2.x and call fortran programs(lmfa,lmchk) internally). Two steps required to complete ctrl file: (i) we give reasonable options when we run ctrlgenM1.py. Then (ii) we may edit the ctrl file afterward. In anyway, ctrl file is the starting point of calculations; ctrls is required just in order to generate ctrl.

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 Brillouin 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)   
 --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 nio --nspin=2 --nk1=6

(NOTE: In the spin-polarized case, we need to set initial condition of size of magnetic moment at each atoms. Set it in ctrls.\* as in the previous section, or edit MMOM of ctrl file (MMOM=s p d f ...) to be like MMOM=0 0 1.2.). The ctrlgenM1.py generates ctrl file named as ctrlgenM1.ctrl.cu. To do calculations, copy it to ctrl.cu so that lmf can recognize it.

cp ctrlgenM1.ctrl.cu ctrl.cu

## crystal structure checker: lmchk

Do lmchk to confirm that we can let lmf know correct crystal structure. It also show crystal structure informaiton, equivalent sites, site index and so on.

lmchk --pr60 cu (--pr# gives more informations if # is number)

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 SYMGRP in Document/Manual/CategoryAndToken.org. about how to represent the operations.
* Show atomic positions in ctrl file.
* Tabulate MT radius and distance between atomic sites.

(lmchk –help shows help, but difficult to see. Not need to read it first.)  
lmchk is also shows atom (MT site) id (position and class(equivalent positions). This is needed to interpret PDOS.  
:

## ctrl file

It is not necessary to look into ctrl file first, although some details are explained in the generated ctrl file. Please compare obtained results by lmf with those by other packages or literatures; let me know if you find something strange or your questions.

It is necessary to edit ctrl file to use full ability of lmf. For example, LDA+U, atomic position relaxation, core level spectroscopy, Change setting of default MTO and lo, better mixing procedure for stable convergence; higher accuracy, and so on.

But a few of ctrl file is easy to modify. Search these words and read explanations embedded in ctrl file.  
(1)XCFUN  
(choice of XC—it is not need to repeat ctrlgenM1.py). It is also possible to change number of k points for sampling, to modify crystal structure slightly, and so on; all things needed are in ctrl. It is not needed to repeat ctrlgenM1.py again.  
(2)SO  
To obtain correct dispersion around top of valence at point for GaAs, we need to set SO=1 and NSPIN=2. QSGW calculation (by gwsc) do not allow this option now; Thus we run lmf (or lmf-MPIK) with such settings changed in ctrl file, after QSGW is converged.  
lmf --input shows what can we write in ctrl file. But more than half are not for users, but for developers (or irrelevant now).

## Run 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”. Qcore is treated by frozen core approximation. See Sec.2.5 in Ref.[[kotani2015pmt]](#kotani2015pmt). Qval means electrons for valence s,p,d channels. The valence channels are 4s,4p,4d,4f (if we set EH=s,p,d,f) in this case The int(P)z column is for local orbital. Thus we have 3d treated as local orbital. (ecalj allow add one local orbital per .)

The isp index means spin (1 or 2), since –nspin=1 (when we invoke ctrlgenM1.py) for GaAs, 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

In unix, we ca save console output to llmf by $ lmf cu | tee llmf. As it starts iteration calculations, it shows similar output again and again (it is a little too noisy now). 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 procedure. Although they are different during iterations, it finally get to be the almost the same number. (But they can be slightly different even converged for large systems. But you don’t need to care it so much).  
NOTE: 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 obtained by each iteration.

“ grep gap llmf” shows how the band gap changes (in the usual setting), two same numbers per iteration are shown now.

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 (for convenience, it is simply determined from 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. 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.

## DOS, 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, 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, it might be better to enlarge number of k point NKABC in ctrl file to have smooth curve. Then we do

job\_tdos cu

This shows total DOS as

![DOS(Cu)](data:application/eps;base64,)

DOS(Cu)

The range of DOS and division of total DOS is given by DOSMAX and NPTS in ctrl. Edit tdos.cu.glt for gnulot for your presentation. Please look into job\_tdos file in your bin directory. It is a small script.

For PDOS plot,

job\_pdos cu

It shows figures (number of figures are number of atoms in the cell) in gnuplot (they are written in the same position on X-window; move top one a little). The command job\_pdos is a little time-consuming because we use no symmetry to distinguish all lm channels. (PDOS is not yet implemented for SO=1 case; spin-orbit coupling $\L\dot S$ is added.) We can edit script of gnuplot (pdos.site\*.\*.glt) for your purpose. To plot again, run

gnuplot -persist pdos.site001.cu.glt

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).  
: Usually lmf and so on recognize options such as -v option. For example, ’ lmf gaas -vnspin=2’ or ’lmf gaas -vso=1’. This option changes values of variables defined in % const section. This is recorded in save.\* file, and also shown at the top of console output. However, job\_tdos and so on, do not yet accept these options. Thus we need to modify ctrl file without using -v option. Or you need to write these option to these command by hand (we will fix this problem in future.)

## Band plot

For band plot, we have to set symmetry lines along which we plot eigenvalues. Collections syml.\* are in ecalj/MATERIALS/. (On Jan2017, we prepared automatic generator of syml.\* See Sec.[4.7.1](#sec:getsyml)). Choose and modify one of them and rename it. I will gather other samples soon. ’BZ wikipedia’ or something else will help you to interpret it.

To do band plot, we need syml.cu in your directory.

$ cp ~/ecalj/MATERIALS/Cu/syml.cu .

Then check syml.cu; it is

21 .5 .5 .5 0 0 0 L Gamma  
 21 0 0 0 1 0 0 Gamma X  
 0 (this is the terminator line)

We supply ten data for each lines (integer, three real, three real, two words). First line means, we calculate eigenvalues for **k** points from **k**=(0.5,0.5,0.5) to **k**=(0,0,0). "L Gamma" are names of two end points (.5 .5 .5) and (0 0 0) in this case. These names are used in a gnuplot script for band plot (bndplot.isp\*.glt). 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 2ALAT (or 2(ALAT+DALAT) if DALAT exist.). A line starting from ’#’ is neglected (comment line).

To do band plot, run

$ job\_band cu

. This is for both nspin=1 and nspin=2 (These scripts try to determine the Fermi energy first. You may skip it in cases (but need to change the script)).

![band plot(Cu)](data:application/eps;base64,)

band plot(Cu)

### syml genearator

Jan2017, we added directory ecalj/GetSyml, which contains automatic generator of symmetry-line file syml.\* from ctrl.\*). Simultaneously, we plot a picture of BZ showing the symmetry lines by matplotlib. See ecalj/GetSyml/README. We use open softwares spglib and seekpath (these should be installed separetely). We have to add acknoledgement to them (see README) )when we use this generator.

## Useful samples: ecalj/MATERIAL/

Not only ecalj/lm7K/TestSamples (some of them are by older version), We have a material database in ecalj/MATERIALS/. Move to the directory, and type

$ ./job\_materials.py

Then it shows a help. You see

...  
=== Materials in Materials.ctrls.database are:===  
 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  
 ZnO ZnS ZnSe ZnTe ZrO2 wCdS wZnS  
...

. For these simple materials (now 57 materials), input files can be generated, and run them automatically by a command ./job\_materials.py below. 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 about how to add new material to it; it is not difficult. The command ./job\_materials.py gives ctrls.\* for these materials from descriptions 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).

Try ./job\_materials.py Fe --noexec. (not fe but Fe as it shown above) at ecalj/MATERIALS/. 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 ctrls and ctrl files. (Look into ctrls.fe; we need SPEC section when we add MMOM.)

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. To change the number of cores for lmf-MPIK, set option -np (number of core). See help of ./job\_materials (type this without arguments).  
To make band plot and so on for Fe, follow instructions already explained.

$ ./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 fe -np 4   
 (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).

At the end of job\_pdos, we show a help which pdos data is where (In pdos file, we have 26 numbers a line; first is energy, 2-26 are pdos for s,p,d,f,g; which is which are shown in the 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; I have to do it).

After doing ./job\_materials foobar, you may like move it back to original... In such a case, git works. At ecalj/, do

$ mv MATERIALS MATERIALS.bk  
 $ git checkout MATERIALS

Then you can see MATERIALS/ is moved back to just downloaded one.

## How to add spin-orbit coupling?

In the LDA level calculations, we can use  
SO=0 (no SO), SO=1(LdotS), or SO=2(LzSz) schemes.  
However, SO=1 is not the non-collinear method  
(z-direction is assumed to be spin direction; in cases  
we may need to primitive cell so that the z-direction  
is the spin direction).  
  
Within LDA, we have two possible ways.  
(A) Do LDA and/or QSGW with SO=0 first.  
Then apply the spin-orbit coupling by perturbation.  
or (B) do self-consitent LDA calculations with SO=1 (or =2).  
For semiconductors, we think you observe little differences.  
  
Let me explain the case (A).  
After converged with nspin=1, create new directory and copy  
 ctrl.gas, rst.gas, sigm.gas, QGpsi,  
to it. Then we set  
 nspin=2   
 METAL=3 (usually this is default)  
 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 per iteration is shown in metel mode).  
This is the band gap with SO as a first-order perturbation   
on top of the "QSGW without SO". When you use ctrl fil e 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. Run lmf --help.  
 The switch -v (-vso=1 in this case) replaces so=0 with so=1.   
 This is recorded in save.gas file).  
  
For band plot, you can use the same procedure   
for the case without SO. (Look into the shell script job\_band.  
You have to modify it so that   
 '--rs=1,0 gas -vnit=1 -vso=1 -vnspin=2 -vmetal=3 --quit=band'   
is added to arguments for >lmf --band:syml ...).  
(--quit=band is not necessary if we like to renew eigenfunctions  
self-consistent. Anyway we expect little differences.)  
  
---  
  
\noindent QSGW with SO:\\  
For given sigm file, it is possible to attain self-consisency with SO=1(or=2) with keeping sigm (then we do not set Q=band).   
However, this imply that Vxc is fixed in QSGW,   
it is not necessary better than the above procedure.

## PROCASR (VASP format) generator

PROCASR mode for lmf (not yet in lmf-MPIK) Band weight decomposition. (Size of circles show the size of components. Superposed on band plot). See ecalj/MATERIALS/MgO\_PROCAR/README, May19.2014

## efermi.lmf

This is generated by lmf(lmf-MPI), which is used in the job\_band.

## Effective mass calculation

New band plot and effective mass calculation (curvature):  
(see sample at ecalj/MATERIALS/mass\_fit\_test/ and its README).  
------------  
We now read syml.\* in lm7K/fp/bndfp.F. Thus job\_band is changed  
(nspin=1 or nspin=2 is automatically choosed by job\_band command).  
We do not use plbnd anymore. If necessary, you can modify  
"writeband" subrouitne in lm7K/fp/bndfp.F by yourself.  
  
Follow steps to get effective mass...  
1.   
New syml read label of k point. It is shown by the gnuscript  
file "bandplot.glt". Type "gnuplot -p bandplot.isp1.glt" and so on,  
when you like to remake band plot.  
  
2.   
New syml allow a special input suitable to determine effective mass  
for semiconductor. An example of new syml is (this is a case of GaAs)  
------ syml example start ------------------  
#NOTE: ndiv2, ninit2 nend2 etolv etolc are for mass mode.  
#ndiv qleft(1:3) qright(1:3) llabel rlabel ndiv2 ninit2 nend2 etolv(Ry) etolc(Ry)  
5 0 0 0 .5 .5 .5 Gamma L 257 1 32 0.1 0.01  
5 0 0 0 1. 0 0 Gamma X   
5 0 0 0 .75 .75 0 Gamma K 257 1 32 0.1 0.01  
------- end ----------------------  
New feature is start from the next columns of symmetry points labels.  
In this example, 257 means a line connecting 0 0 0 (Gamma) and .5 .5 .5 (L) is divided  
into 256. Then we only calculate from the 1st point to the 32th point  
among 257 points.  
  
Then we see not only band\*.spin\*, but also  
we have files such as "Band001Syml001Spin1.dat"   
files which contains eigenvalues for selected bands, that is,  
we make a table of detailed plot of energy bands   
whose energy E at Gamma (exactly speaking, at  
left-end point), is evaltop-etolv (Ry) < E < econtop + etolc(Ry).  
  
(As its head line shows, Band\*Syml\*.dat file contains data  
 isyml,iq, ib,isp, QPE-EF, QPE-QPE(start), |q|, mass=2\*2\*(QPE-QPE(start))/|q|\*\*2).  
 here QPE(start) is QPE at the left-end of sym line.)  
  
This sample is only for Gamma-L and Gamma-K.  
  
Let us use this example for GaA and run job\_band.  
(e.g., job\_band gaas -np 2 -vnspin=2 -vso=1;   
 note that -vfoobar=xxx replace value of foobar with xxx in ctrl.gaas)  
Then you can see not only bandplot.isp1.glt.  
Then you can see usual energy bands plot in addition to the dense energy bands  
superposed on it. The dense energy bands are controlled by  
the part "257 1 32 0.1 0.01". In the current version, set all lines  
of etolv and etolc are the same.  
  
There are useful data, not only the eigenvalues in "Band001Syml\*Spin\*.mass"   
files. But, anyway, we have to determine effective mass from the eigenvalues  
on the dense symmetry mesh.  
  
  
3.   
Least square fit by gnuplot.   
  
We have an example is at ~/ecalj/MATERIALS/mass\_fit\_test  
Look into ./job and run it.   
See README in it.  
----  
For your convenience, we have dE/dk in the bnd\*.spin\* files.  
This is useful to determine the Fermi surface.  
See the efermi.lmf to read the Fermi energy.

## density and eigenfunction plot

We can generate density and $\sum\_{n} |\Psi\_{{\bf k}n}({\bf r})|^2$ by --density mode (after 18mar2016) Usages are

>mpirun -np 24 lmf-MPIK si --density

This is for density. We have smrho.xsf, and rho1mt\*(1st components) and rho2mt\*(2nd components).

>mpirun -np 24 lmf-MPIK si --density,iq=1,ib=5

iq=1 means Gamma point. See index shown by the console output of lmf. The index ib is the band index.

>mpirun -np 24 lmf-MPIK si --density,iq=1,ib=5,6,7

In this case we have sum for ib=5,6,7.

Use xcrysden as

>xcrysden --xsf smrho.xsf

Then open menu ’ToolsDetaGrid’. Then plot isosurface and/or planes. For example, you need to set isovalue, and/or turn on ’display color plane’ buttons.

## LDA+U

There is a document of <file://../BACKUP/MarksOriginalDoc/fp.html#ldaplusu>. But I include all required things here. Usage is slighly changed. We have examples at  
~/ecalj/MATERIALS/erasldau,GdNldau. I am going to add some samples.

To run LDA+U calculaiton, we need two extra steps in addition to usual LDA calculaitons:  
Step.(1) Set three tokens in ctrl file. LDA+U type, Parameters U and J.  
Step.(2) Prepare occnum.*ext* (initial condition of diagonal part of the density matrix).  
Let us explain one by one.  
  
For the point (1), we have a new line in SPEC\_ATOM sections. In ecalj/MATERIALS/erasldau, it is

IDU= 0 0 2 2 UH= 0 0 0.1 0.632 JH= 0 0 0 0.055

The SPEC\_ATOM\_IDU token tells lmf that no U is to be added to the s or p channels, but that a U is to be added to the d and f channels. IDU=2 specifies LDA+U functional style 2; this is the "Fully Localized Limit" described in Liechtenstein, PRB 52, R5467 (1995)). IDU=1 specifies the "Around Mean Field" functional (Petukhov, PRB 67, 153106 (2003)). U=0.1 Ry is included on the d orbital, and U=0.632 is included on the f orbital. Additionally J=0.055 is put on the f orbital.

It is stored in dmats.ext. In principle, the density-matrix is read from and written to the file dmats.ext. Thus, in principle, we need a dmats.ext as an initial condition. However, we usually use diagonal-only density matric occunum.ext instead. If no dmats.ext, we read occnum.eras instead. We explain it afterwards. Two density-matrices (for each spin) are written to this file in a (2l+1) by (2l+1) block for each l block for which a U is defined. dmats.ext is an ASCII file which you can read, and it’s quite useful to interpret what’s going on. The diagonal parts are the occupation numbers and are the most important. It is in the spherical harmonics (not the real harmonics). The harmonics is defined in the ecaljdetail.tex.

Note that in the LDA case, complete information is contained in the density, stored in file rst.ext. In the LDA+U, the Density-matrix is stored in file dmats.ext. Thus complete information is contained in the combination of rst.ext and dmats.ext.

The density matrix is printed out in the standard output. It can be represented on the real harmonics. To check it, run

>lmf eras --quit=dmat

. This stops promptly without side effect. Then the symmetrized density matrix represented on the spherical harmonics and real harmonics are shown (we make symmetrization for the given SYMOPS in ctrl file).

Step.(2)  
Usually it is too tedious to supply dmats.ext itself. Instead, you can supply an "occupation numbers" file occnum.ext, which is a starting guess for the density-matrix (its diagonal part). occnum.ext has one line of (2l+1) numbers for the occupation numbers of the first spin, following by a line with the occupation numbers for the second spin. In the case of ecalj/MATERIALS/erasldau, we have occnum.eras.

Er has 11 f electrons, 7 of which go into the majority channel and 4 into the minority channel. There is some choice in which m states to fill and which to keep empty. A key point is that the self-consistent solution you end up with will depend on this choice. The ErAs test uses the following input file for occnum.eras :

0 0 0 0 0  
 0 0 0 0 0  
 1 1 1 1 1 1 1  
 0 1 1 1 1 0 0

The first and second lines are occupation numbers for the majority and minority d channel; the third corresponds to the majority f channel where all states are taken to be filled. The last line corresponds to the minority f channel. In this case, m=-2,-1,0,1 are filled and m=-3,2,3 are empty. As the script notes, different choices of starting occupation numbers lead to different self-consistent solutions. The one with the lowest energy is that which satisfies Hund’s rule (m=0,1,2,3 filled and m=-3,-2,-1 empty).

It is possible to use ’real harmonics (see ecalj/fpgw/exec/jobpdos for definition)’ instead of ’spherical harmonics’. As for occnum.ext, you can use

% real  
 0 0 0 0 0  
 0 0 0 0 0  
 1 1 1 1 1 1 1  
 0 1 1 1 1 0 0

. Here '% real' means on the basis of real harmonics. Check whether this is correctly supplied or not by >lmf --quit=dmat eras.

To run with spin-orbit coupling, set HAM\_SO=1, then run >grep IORBTM -A20 llmf , here llmf is the standard output of lmf. Then you see orbital moments at ’IORBTM’.

NOTE for gwsc:  
When you just like to generate initial condition for gwsc by LDA+U, you have to exit right after sigm generated (add exit right after $echo\_run echo == $ix 'iteration over==' in gwsc). Then gwsc stops right after sigm generated. Then you have to remove or comment out IDU (or set all zero) in ctrl file. Then run lmf together with obtained sigm file. This is because sigm contains LDA+U kind of effect. After this, check energy band (jobband) for check. And run usual gwsc— then you can perform QSGW with the LDA+U’s initial condition.

## QSGW after LDA+U

We modified lm7K/subs/m\_rdctrl.F(feb2019), so that An option which makes “QSGW after LDA+U” easy to run. A line for atom section

IDU=0 0 0 12 UH=0 0 0 .7 JH=0 0 0 0

works in ctrl file. Here 12=10+2 instead of 2 means that we set UH=JH=0 when sigm.\* exits. Thus we can perform QSGW(without U term) calculaiton, after LDA+U calculaiton. This can be convenient for cases when we control the initial condition of QSGW.

## partially occupied core-hole

See <file://../BACKUP/MarksOriginalDoc/fp.html#ldaplusu> around. Search the keyword ldau in fp/lmfp.F sudmtu.F which read occnum.gdn.

Core-hole procedure is one of the important procedure. An example setting for Nd case is

--------------------------  
 ATOM=Nd Z=60 R=3.00  
 C-HOLE=4f C-HQ=-11 P=6.5,6.5,5.5,5.2　　<-- this line is added  
--------------------------

1. C-HOLE=4f means that 4f core is to set “core hole”.
2. P=6.5,6.5,5.5,5.2 is to set principle (fractinal) quantum number explicitly for valence electrons. 　To set this, try lmfa nd|grep conf first. The you can see the electronic configulation of an atom. Key point here is to set “5.2” for f channel. Thus 4f is treated as core and 5f as valence. Fractional parts (0.2 of 5.2) means the radial funciton contains nodes as 5f and closer to the energy right on the change of 4f to 5f. If 5.9, it is closer to the enrgy almost have noded of 6f.
3. C-HQ means that corehole is −１１. Thus we have only 14-11=3 electrons for 4f (spherically occupied).
4. Run lmfa. If it does not show that freeat (warning) atom not neutral, it is the neutral atom. You can also chage the message tagged by Add core hole: Simultaneously you can control initial total valence charge by Q=...

Very recently (aug2019), I fixed a bug of EFERMI(fermi energy for GW calculaiton, given by fpgw/main/heftet.m.F) for the case C-HOLE is supplied.

# How to run QSGW calculation?

In the QSGW, we calculate a *non-local exchange-correlation potential* $V^{\rm xc}({\bf r},{\bf r}')$, by a procedure of GW calculation (very time-consuming part). Then difference $V^{\rm xc}({\bf r},{\bf r}')V-V\_{\rm xc}^{\rm
LDA}({\bf r})\delta({\bf r}-{\bf r}')$ is stored into sigm.\* file. The potential file sigm is a key to perform QSGW calculations as seen in Fig.[[gwscpicture]](#gwscpicture). The sigm contains static non-local potential $\Sigma\_{\rm QSGW}-V\_{\rm xc}^{\rm LDA}$.

Then, we again do one-body calculation by lmf (or lmf-MPIK) where we add this sigm to one-body potential ; when we run lmf or lmf-MPIK(k-parallel mpi version), sigm is read and added to the one-body potential if we have HAM\_RDSIG=12 in ctrl.\*. Thus this means that we replace $V\_{\rm xc}^{\rm LDA}({\bf r})\delta({\bf r}-{\bf r}')$ with $V^{\rm xc}({\bf r},{\bf r}')$.  
This iteration cycle is performed by a script “gwsc” as we explain later on. (In the default setting of ctrl.\* file, lmf try to read sigm.\* file as long as it exists. If not, do lmf or lmf-MPIK calculation.

To start QSGW calculation by gwsc, we need not only ctrl.si, but also another input file GWinput. Its template GWinput.tmp can be generated by the Step.2 as follows.

As a summary, you have to follow steps below in order to perform QSGW calculation.

1. Perform from the Step.1 thru the Step.4 (up to lmfa) in Sec.[4.5](#lm7K-scf). (as same as the case of LDA/GGA).

* You don’t need to perform LDA calculation in advance, since gwsc perform LDA/GGA calculation at its beginning. (It means that we start from the one-body Hamiltonian in LDA/GGA as initial condition.In cases, LDA/GGA give poor initial conditions for QSGW; in such a case, we may need another trick to prepare starting point.).
* [Caution: We have to use the same LMXA ( in the expansion of eigenfunctions in each MT) for all the MT spheres. (This is due to historical reason; we may need to fix this.)] xxx need to explain LMA xxx
* The purpose of this script is to get GWinput.tmp. Other files generated are not used in the following stage.

1. .  
   GWinput is the input file describing the computational conditions for calculation. Usually, the default setting gives reasonable results. To reduce computational time, we may use lcutmx(atom)=2 for oxygen sites (this may be also other small atoms.). grep lcut GWinput -A1 shows lcutmx for each atomic sites) These step 2. and step 3. are just only to get GWinput.

## GWinput

In order to perform QSGW, one another input file GWinput (no extension) 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 --nk1=6 and copy ctrlgenM1.ctrl.si to ctrl.si. NOTE: the option –tratio=1.0@ means we use touching MT; this can be checked by lmchk si; since defaults is almost unity (–tratio=0.97@), this is irrelevant, just to explain options.

We have to write GWinput. The default is given automatically by a command mkGWIN\_lmf2;

$ 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. This means 2x2x2 (8 points in 1st BZ). 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 8x8x8.

The template of GWinput is usually not so bad. But it may give a little expensive setting (or not very good enough in cases).

NOTE: GWinput allows some options. Some newer points are

1. For the atoms without 3d involved, you can use lcutmx=2. Typically we use lcutmx=2 for oxygen.
2. GaussianFilterX0 0.0001 given in GWinput is effective to stabilize the convergence for metals.

## Run gwsc script

Let us perform QSGW calculation. For this purpose, we use a script gwsc. We need to do lmfa in advance. Then do (not need to do lmf);

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; since it calculates off-diagonal elements of self-energy also, we can plot energy band dispersion. In cases (for usual semiconductors), it can give 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 beginning 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. (but QGpsi is quickly reproduced). After you have syml.si (e.g. in ecalj/MATERIALS/), Do

$ job\_band si

![Si, one-shot GW with off-diagonal elements](data:application/eps;base64,)

Si, one-shot GW with off-diagonal elements

[sigwscone]

You can observe large band gap as shown in the Fig.[[sigwscone]](#sigwscone). (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) ...  
  
OK! --> == 5 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 ====

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 anyway \*.0run are confusing files; remove them).

In order to check convergence calculations going well during iteration, do

$ grep gap llmf\*

This shows how band gap changes in llmf.\*run files. In metal cases, we need to compare QPU file, magnetic moment or grep '[xc] save.\*; this shows end of lmf iteration. Energy is not so meaningful but can be indicator to convergence.

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.)

![band plot(Si, QSGW one-shot test) and band(Si) (GGA)](data:application/eps;base64,) ![band plot(Si, QSGW one-shot test) and band(Si) (GGA)](data:application/eps;base64,)

![band(GaAs), QSGW (test case)](data:application/eps;base64,)

band(GaAs), QSGW (test case)

## Spectrum function: How to calculate $\langle {\bf q}n|\Sigma(\omega)|{\bf q}n\rangle$

We have an example at ecalj/MATERIALS/SiSigma/, where you can just type job. It calls a shell script gwsigma, which is just a modification of gwsc for spectrum function plotting. If you have sigm.\*, it will automatically read it as in the case of gwsc.

By the script gwsigma, we calculate the diagonal elements $\rangle \psi({\bf q},n)|\sigma\_{\rm c}(\omega) |\psi({\bf q},n)\langle$. Thus we need to set ${\bf q}$ and band index for which we calcualte. In addition, we need to set resolution of .

1. Set <QPNT> section. This section is to set the q point, and band index for which we calculate the self energy. In addition, energy mesh for plotting is set.

* (A) Set q point set   
    
   If you set   
   ----------  
   \*\*\* all q -->1, otherwise 0; up only -->1, otherwise 0  
   1 0  
   ----------  
   You will have self-energy for all irreducible k points. This may be needed for A(omega).  
   or   
   You have to set all q points as  
   ----------  
   \*\*\* q-points, which shoud be in qbz.,See KPNTin1BZ.  
   3 <--- number of readin q point   
   1 0.0000000000000000 0.0000000000000000 0.0000000000000000 <--1st number is irrelevant  
   2 -0.5000000000000000 0.5000000000000000 0.5000000000000000  
   3 0.0000000000000000 0.0000000000000000 1.0000000000000000  
   ----------  
   To know allowed q points on regular mesh point, run the command "mkGWIN\_lmf2", then  
   supply n1,n2,n3. The templete of GWinput.tmp contains all possible q points. Edit it.  
     
   NOTE:Anyq option can allow you to specify any q points by shifted mesh technique.  
   (if necessary, but only for some special purpose).  
    
  (B) Band index set  
   It is specified by the section  
   ----------  
   \*\*\* no. states and band index for calculation.  
   2  
   4 5  
   ----------  
   means the self-energy for band index 4 and 5. Just two bands.  
   If you like to plot self energy from 1 through 8, use  
   \*\*\* no. states and band index for calculation.  
   8  
   1 2 3 4 5 6 7 8  
   ----------  
   If you need 17 bands for example, it should be 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17  
   in addition to 17 (number of bands at the first line).   
    
    
  (C) energy mesh set  
   At the bottom of <QPNT> section, we have  
   ----------------  
   \*\*\*\*\*  
   0.01 2.0  
   ----------------  
   Two real number should be supplied.  
   These are dwplot and omegamaxin, read in hsfp0.m.F by a line   
   read (ifqpnt,\*,err=2038,end=2038) dwplot,omegamaxin  
   dwplot (=0.01 Ry) is mesh for self energy.  
   omegamaxin=(2.0 Ry) means the range "-2 Ry to 2 Ry" for self-energy plot.  
    
   Note that imaginary part of Sigma is given as the comvolution of ImW(omega) and the pole of Green's function  
   (esmr in GWinput gives energy smearing of the pole). Resolution for Im W (near omega=0) is by dw in GWinput.)  
   I think that the reolution of self-energy is ~ 0.05 eV in the default setting.  
   This is because {\tt dw} \sim {\tt esmr} \sim 0.05 eV.

1. Run gwsigma. This will run   
    echo 4| mpirun -np 24 hsfp0,  
    after dielectric funcition is calculated.  
    Then we have SEComg.UP (DN) files, Look for file handle, ifoutsec,  
    for the file in fpgw/main/hsfp0.m.F to see format for the file.   
    (not hsfp0.sc.m.F but hsfp0.m.F). Search a line  
    open(ifoutsec,file='SEComg'//sss) (around hsfp0.m.F L1052)  
    You can find that we use folloing lines to plot SEComg.\*.  
    ----------------   
    write(ifoutsec,"(4i5,3f10.6,3x,f10.6,2x,f16.8,x,3f16.8)")  
    & iw,itq(i),ip,is, q(1:3,ip), eqx(i,ip,is),  
    & (omega(i,iw)-ef)\*rydberg(), hartree\*zsec(iw,i,ip) !,sumimg ----------------   
    This means we use energy in eV.   
    iw: omega index  
    itq(iq): band index specified by <QPNT>  
    ip: k point index specified by <QPNT>  
    is: spin index  
    q: q vector (cartesian in 2pi/alat)  
    eqx: eigenvalue in eV. (I think relative to the Fermi energy)  
    (omega(i,iw)-ef)\*rydberg(): omega relative to the Fermi energy  
    hartree\*zsec(iw,i,ip): Self energy. real and imaginary part.(complex, two values)  
     
    You can only repeat echo 4| mpirun -np 24 hsfp0   
    when you change setting in <QPNT> section.
2. Example. There is an example MATERIALS/SiSigma/

* plot 'SEComg.UP' u ($9):($10) w l,'' u ($9):($11) w l  
   can give a plot for Re (Sigma\_c(omega)) and Im(Sigma\_c).  
    
   9th: energy in eV (omega(i,iw)-ef)\*rydberg()  
   10th: real part Re hartree\*zsec(iw,i,ip)   
   11th: imag part Im hartree\*zsec(iw,i,ip)

1. To get integrated spectrum function (DOS), we need to superpose all the spectrum function (All q points and all band index). Be careful about the degeneracy (multiplicity) for each q points. You have to build it from SEComg file. To know the multiplicity, search following lines ofkeyword Multiplicity in the console output of qg4gw (lqg4gw).

* Anyway, consider about “is it worth to do?” To confirm your result, use sum rule (sum of spectrum weight). And pay attention to the relation between real and imag parts (Hilbert transformation).

1. In cases to treat magnetic systems which have intrinsic magnetic fluctuations, we may need to be careful about initial condition or mixing procedure to get convergence. In cases, we need to start from LDA+U results as initial condition from which we start QSGW. Let me know about such trouble. [↑](#footnote-ref-23)
2. In the case of using sigm.\*, total energy shown in the console output (also in save.\*) are just the indicator, not the meaningful total energy [↑](#footnote-ref-24)
3. current implementation have not yet efficiently use this locality; this must allow us to speed up one-body problem solver. [↑](#footnote-ref-26)
4. Exactly speaking, we use LDA idea for efficient implementation of QSGW; thus obtained results are slightly dependent on the choice of LDA or GGA [↑](#footnote-ref-27)
5. We undo electron-phonon effect (0.06eV) and spin-orbit effect (0.11eV) from the true the experimental value 1.52 eV. [↑](#footnote-ref-28)
6. exactly speaking, we have range cutoff for Fock exchange term in the HSE functional in addition [↑](#footnote-ref-30)
7. If you use (Hartree-Fock limit), the band gap of Si becomes 20eV or something. [↑](#footnote-ref-31)
8. people often pronounce this “H-naught” [↑](#footnote-ref-32)
9. ctrlgenM1.py exists originally at ecalj/lm7K/ (ctrlgenM1.py was already copied to your BINDIR= defined in ecalj/Install.ifort in the installation). [↑](#footnote-ref-42)
10. For these variables, we can overlay values when we start programs. For example,’lmf -vdalat=0.1 si’ means that alat is recorded in save.si file. [↑](#footnote-ref-44)
11. 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. [↑](#footnote-ref-45)