

ecalj MainDocument

This is MainDocument of ecaljdoc. All files are linked from this file.

- Here we give [GetStarted](#), together with install and overview of QSGW.
- We have [UsageDetails](#) in another file.
- [Qiita Japanese](#) may be a help, but most of all are shown here.
- When link did not work, go to <https://github.com/tkotani/ecalj> to download.

Licence

[AGPLv3](#). For publications, we hope to make a citation as;

[1] ecalj package available from <https://github.com/tkotani/ecalj/>.

Install

To install ecalj, look into [install](#), as well as [install for ISSP](#)

Features of ecalj package

1. All electron full-potential PMT method

The PMT method means; a mixed basis method of two kinds of augmented waves, that is, APW+MTO.

In other words, the PMT method= the linearized (APW+MTO) method, which is unique except the [Questaal](#) having the same origin with ecalj. We found that MTOs and APWs are very complementary, corresponding to the localized and the extended natures of eigenfunctions. That is, very localized MTOs (damping factor $\exp(-\kappa r)$ where $\kappa \sim 1$ a.u.; this implies only reaching to nearest atoms) together with APWs (cutoff is ≈ 3 Ry) works well to get reasonable convergences. We can perform atomic-position relaxation at GGA/LDA level. Because of including APWs, we can describe the scattering states very well.

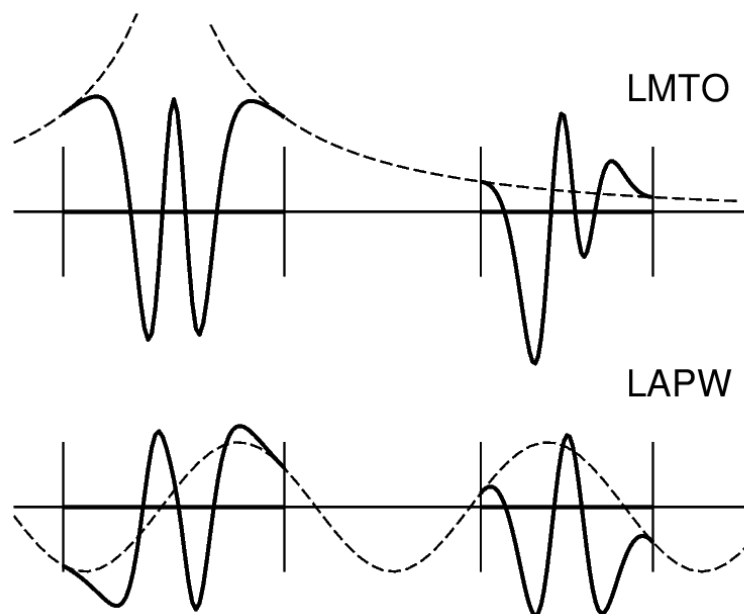


Figure 2.1: Qualitative sketch of the LMTO and LAPW basis functions. Both start from a smooth envelope function (shown dashed). The envelope is defined as an atom-centered Hankel function when making an LMTO and a plane wave in the case of an LAPW. Inside the atomic spheres (shown by thicker lines), the envelope functions are replaced by numerical solutions of the Schrödinger equation which match smoothly at the sphere boundaries.

(This fig is taken from [nfp-manual](#) by M. Methfessel and M. van Schilfgaarde)

The current PMT formulation is given in

[1][KotaniKinoAkai2015, PMT formalism](#)

[2][KotaniKino2013, PMT applied to diatomic molecules.](#)

Since we have automatic settings for basis-set parameters, we don't need to be bothered with the parameter settings. Just crystal structures (POSCAR) are needed for calculations.

- Original PMT was started at <https://journals.aps.org/prb/abstract/10.1103/PhysRevB.81.125117>
- PMT uses smooth Hankel functions described in [the smooth Hankel paper](#) by E. Bott, M. Methfessel, W. Krabs, and P. C. Schmidt, which was used in [B][nfp paper](#) by M. Methfessel and M. van Schilfgaarde, and R.A. Casali. Our PMT is on top them.

In addition to PMT basis, we use local orbitals together.

2. PMT-QSGW method

The PMT-QSGW means

the Quasiparticle self-consistent GW method (QSGW) based on the PMT method.

After converged, we can easily make band plots without the Wannier interpolation. This is because

an interpolation scheme of self-energy is internally built in.

We can handle even magnetic metals. Since we have implemented ecalj on GPU, we can handle ~40 atoms with four GPUs.

[3][Kotani2014, Formulation of PMT-QSGW method](#)

[4][D.Deguchi PMT-QSGW applied to a variety of insulators/semiconductors](#)

[5][M.Obata GPU implementation](#), where we treat Type II GaSb/InAs (40 atoms) with four GPUs.

3. Dielectric functions and magnetic susceptibilities

We can calculate GW-related quantities such as dielectric functions, spectrum function of the Green's functions, Magnetic fluctuation, and so on. Since our QSGW can generate eigenfunctions and eigenvalues at any k points.

4. The Model Hamiltonian with Wannier functions

We can generate the effective model (Maxloc Wannier and effective interaction between Wannier functions).

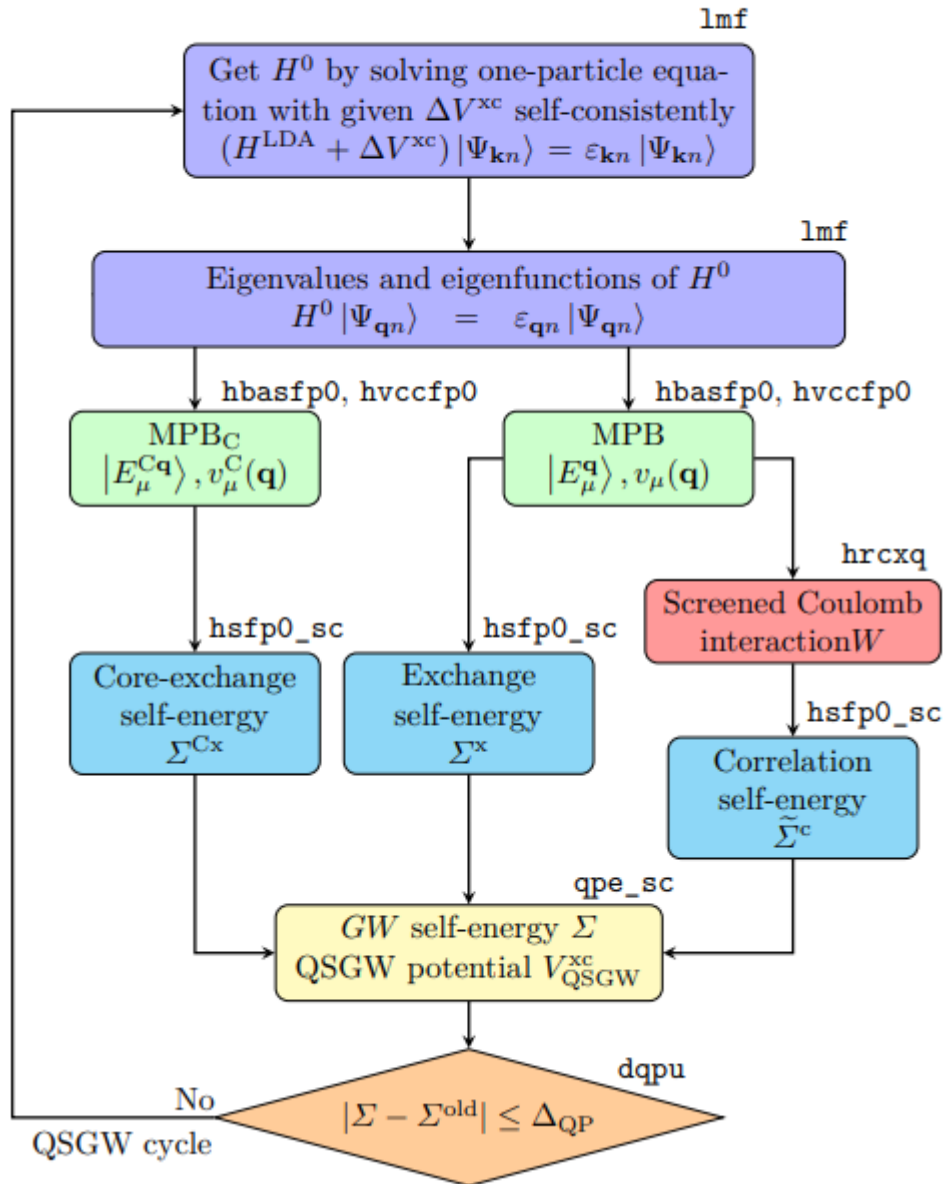
This is originally from codes by Dr.Miyake, Dr.Sakuma, and Dr.Kino. The cRPA given by Juelich group is implemented. We are now replacing this with a new version MLO (MuffinTinOrbail-based Localized Orbitals), where we can easily throw away (downfolding) the APW degree of freedom.

We try to construct the effective Hamiltonian in

<https://journals.aps.org/prb/abstract/10.1103/PhysRevB.108.035141> via a mapping procedure.

Overview of QSGW

- Band calculations (LDA level) are performed with the program `lmf`. The initial setting file is `ctrl.foobar` (`foobar` is user-defined). Before running `lmf`, it is necessary to run `lmfa`, which is a spherically symmetric atom calculation to determine the initial conditions for the electron density (`lmfa` finishes instantaneously).
- A file `sigm.foobar` is the key for QSGW calculations. The file `sigm.foobar` contains the non-local potential $\Delta V_{\rm xc} = V_{\rm xc}^{\rm QSGW} - V_{\rm xc}^{\rm LDA}$. By adding this potential term to the usual LDA calculation performed by `lmf`, we can perform QSGW calculations. See figure below.
- Thus the problem is how to generate $V_{\rm xc}^{\rm QSGW}(\mathbf{r}, \mathbf{r}')$. This is calculated from the self-energy $\Sigma(\mathbf{r}, \mathbf{r}', \omega)$, which is calculated in the GW approximation. Roughly speaking, we obtain $V_{\rm xc}^{\rm QSGW}(\mathbf{r}, \mathbf{r})$ with removing the omega-dependence in $\Sigma(\mathbf{r}, \mathbf{r}', \omega)$.
- Therefore, the calculation of $V_{\rm xc}^{\rm QSGW}$ is the major part of the QSGW cycle, and is calculated in a double-structure loop. That is, there is an inner loop of `lmf`, and an outer loop to calculates $V_{\rm xc}^{\rm QSGW}$ using the eigenfunctions given by `lmf`. This outer loop can be executed with a python script called `gws` (which runs fortran programs). The computational time for QSGW is much longer than that of LDA calculation.
As a rule of thumb, it takes about 10 hours for 20 atoms
(depending on the number of electrons. For systems more than 10 atoms per cell or so, we recommend to use GPUs).
- Here is the QSGW cycle shown in Figure 1 in <https://arxiv.org/abs/2506.03477>. MPB meand the mixed product basis to expand products of eigenfunctions.



- We have [GPU acceleration for QSGW](#), which also describe basics of QSGW. QSGW algorithm fits to GPU computations very well. With four GPUs, we can compute systems with 40 atoms per cell. (As for lmf part, GPUs are not efficiently used yet.). Our PMT allows us to handle large vacuum region for slab model.
- We can perform QSGW virtually without parameter settings by hands. Thus I think ecalj is one of the easiest to perform GW/QSGW. See band database in QSGW at <https://github.com/tkotani/DOSnpSupplement/blob/main/bandpng.md> (this is a supplement of <https://arxiv.org/abs/2507.19189>). This is away from complete database, but showing the ability of ecalj.

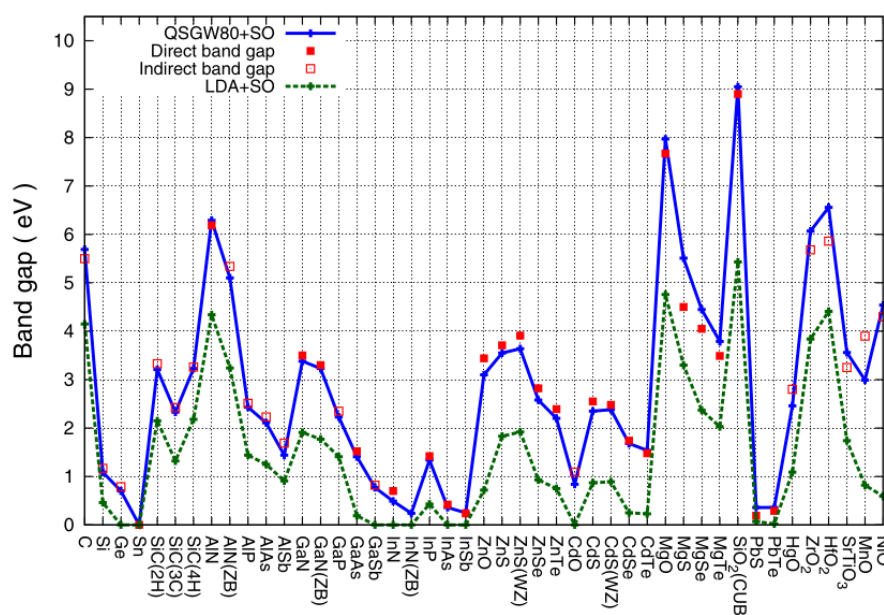


Fig. 1. (Color online) Band-gap energies calculated by using QSGW80+SO (blue solid line) and LDA+SO (green dotted line), together with experimental values (solid squares: direct band gap, open squares: indirect band gap). Respective values are shown in Table II.

This is taken from [4][D.Deguchi](#)

In comparison with LDA, we see differences in QSGW;

- Band gap. QSGW tends to give slightly larger than experiments. It looks systematic as in the Figure above.
- Band width. Usually, sp bands are enlarged (except very low density case such as Na). This is consistent with the case for homogeneous electron gas. Localized bands like 3d electrons get 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 get larger (like LDA+U). 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.

PROF

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

GetStarted

Here we explain DFT/QSGW calculations with ecalj. Then we explain how to make band plots. For simplicity, we treat paramagnetic cases (nsp=1), no 4f, no SOC. We explain things step by step.

Further details are explained at [UsageDetailed](#)

Step 0. Get POSCAR

We first need POSCAR (crystal structure in VASP format).

You can find samples of POSCAR in ecalj/ecalj_auto/INPUT/testSGA/POSCARALL as

```
cd ecalj
mkdir TEST
cd TEST
mkdir test1
mkdir test2
cat ecalj_auto/INPUT/testSGA/joblist.bk
cp ../ecalj_auto/INPUT/testSGA/POSCARALL/POSCAR.mp-2534 test1
cp ../ecalj_auto/INPUT/testSGA/POSCARALL/POSCAR.mp-8062 test2
```

For example, POSCAR of mp-2534 GaAs is given as:

```
Ga1 As1
1.0
3.52125300000000002 0.00000000000000000 2.0329969999999999
1.17375100000000000 3.31986900000000002 2.0329969999999999
0.00000000000000000 0.00000000000000000 4.0659929999999997
Ga As
1 1
direct
0.00000000000000000 0.00000000000000000 0.00000000000000000 Ga
0.25000000000000000 0.25000000000000000 0.25000000000000000 As
```

This is another POSCAR for ba2pdo2cl2 (QSGW results are shown below):

```
POSCAR_ba2pdo2cl2
1.0
-2.06443 2.06443 8.40383
2.06443 -2.06443 8.40383
2.06443 2.06443 -8.40383
Ba Pd O Cl
2 1 2 2
Cartesian
0.0 0.0 6.5153213224
0.0 0.0 10.2923386776
0.0 0.0 0.0
0.0 2.06443 0.0
2.06443 0.0 0.0
0.0 0.0 3.1625293056
0.0 0.0 13.6451306944
```

If you have cif and like to convert it to **POSCAR**, do

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

Step 1. convert POSCAR to ctrl

Then we convert POSCAR to ctrl by vasp2ctrl.

ctrls is the structure file used in ecalj.

```
vasp2ctrl POSCAR.mp-2534
mv ctrls.POSCAR.mp-2534.vasp2ctrl ctrls.POSCAR.mp-2534
cat ctrls.mp-2534
```

ctrls.mp-2534 contains crystal structure equivalent to POSCAR:

```
cat ctrls.mp-2534
STRUC
  ALAT=1.889726877743552
  PLAT=      3.52125300000      0.00000000000      2.03299700000
           1.17375100000      3.31986900000      2.03299700000
           0.00000000000      0.00000000000      4.06599300000
SITE
  ATOM=Ga POS=      0.00000000000      0.00000000000
0.00000000000
  ATOM=As POS=      1.17375100000      0.82996725000
2.03299675000
```

- MEMO:
 - ctrl2vasp ctrl.mp-2534 can convert back to VASP file. Check this by VESTA. We can use viewvesta (convert and invoke VESTA).
 - many unused files are generated (forget them).
 - you can use any name for sites such as [Niup or something](#), in such a case you have to set SPEC section in addition. [Non integer number of Z is allowed](#).. Learn afterward.
 - In old ctrls, you may see NL,NBAS,NSPEC, which are not necessary now.

Step 2. Get ctrl from ctrls

PROF

ctrl is the basic input file for ecalj. We generate template of ctrl with ctrlgenM1.py from ctrls.

ctrl has user-defined extension as `ctrl.foobar`.

Minimum explanations, which we expect to read by users, are embedded in the generated ctrl file.

When we run lmf, we can add command line option such as -vnspin=2. Then `%const foobarx=1` defined in the ctrl file is overridden (referred with {foobarx}). save.* file shows values of foobarx you used.

It is possible to enforce antiferro symmetry (except so=1 case).

We only need ctrl file in the following calculations (while some tmp* tmp2* and so on are generated).

```
ctrlgenM1.py mp-2534
```

If no problem, you see

```
...  
=== End of ctrlgenM1.py. OK! A template of ctrl file, ctrlgenM1.ctrl.mp-  
2534, is generated.
```

Here `ctrlgenM1.py` internally calls `lmf` and `lmchk`, which generate irrelevant files which are automatically deleted.

'SiteInfo.lmchk and PlatQlat.chk' are explained later on (these are easily reproduced by ctrl).

Then copy as

```
cp ctrlgenM1.ctrl.mp-2534 ctrl.mp-2534
```

and edit `ctrl.foobar` if necessary.

How to edit? Explanations are embedded in ctrl.foobar (please let me know wrong descriptions). Possible points to rewrite in ctrl.foobar:

1. Number of k points (nk1,nk2,nk3).
2. nsp=2 if magnetic
3. SpinOrbitCoupling: so=0 (none), so=1 (LdotS), 2 (LzSz). nsp=2 is required for so=1,2. so=1 does not yet support QSGW. SOC axis can also be freely selected, but currently (0,0,1) default and (1,1,0) are supported (m_augmbl.f90). If you want to set SO=1 in QSGW, currently, run QSGW calculation with so=0 or so=2 to obtain ssig file, then set so=1
4. xcfun (choice of LDA exchange correlation term). Only =1:BH, =2:VWN, =103:PBE-GGA.
5. LDA+U settings (not explained yet).
6. ssig=1.0 (If you choose QSGW80, use ssig=0.8) is for QSGW calculations.
 $V^{\text{xc QSGW}} - V^{\text{xc LDA}}$ is stored in a file `sigm.foobar`. We add $\text{ssig} \times (V^{\text{xc QSGW}} - V^{\text{xc LDA}})$ to the potential in the lmf calculation as long as `sigm.foobar` file is available in the same directory.

- `lmchk --pr60 foobar` allows you to check the recognized symmetries by `lmf`. Turning off `--pr60` or reducing 60 will reduce the verbosity of output.

At this point, you can visually check the following check files.

- SiteInfo.chk
MT radius Atomic positions
- PlatQlat.chk
Primitive lattice vector (plat) Primitive reciprocal lattice vector (qlat)

[Here we explain details of ctrl file.](#)

Hereafter, we only use `ctrl.foobar` (`ctrls.foobar` is used hereafter.). We can delete other files.

Install VESTA

It is convenient to see crystal structures with VESTA.

(I installed VESTA-gtk3.tar.bz2 (ver. 3.5.8, built on Aug 11 2022, 23.8MB) on ubuntu 24)

At ecalj/StructureTool/, we have 'viewvesta' command.

Try

```
viewvesta ctrl.si
```

to see the structure in VESTA. To show ctrl.si, we use a converted at /StructureTool, [vasp2ctrl](#) and [ctrl2vasp](#).

(We have ~/ecalj/GetSymb/README.org. but Users do not need to read this.)

Step 3. LDA calculation

1. Run [lmfa](#) at first. It is for spherical atomic electron densities, contained in the crystals. [lmfa](#) ends instantaneously.

```
lmfa ctrl.mp-2534
```

gives spherical atom calculation for initialization. [lmfa](#) calculates spherically symmetric atoms and generates the files required for [lmf](#) below.

Check [conf](#) section in the console output as

```
lmfa ctrl.mp-2534 |grep conf
```

. This shows atomic configuration (there are no side effects even if [lmfa](#) is repeated). The initial condition of electron density for [lmf](#) is given as the superposition of spherically symmetric atomic densities given by [lmfa](#). In addition, [lmfa](#) calculations are performed with the logarithmic derivative of the radial wave function at the MT sphere edge fixed (READP True in default [ctrlgenM1.py](#) setting). The derivatives are contained in [atmpnu.*](#) files. So, [atmpnu.*](#) are needed for [lmf](#).

2. After [lmfa](#), we run LDA calculation as:

```
mpirun -np 8 lmf mp-2534 |tee llmf
```

- mp-2534 (GaAs) gives 5.75 eV for GaAs, while the experimental value is 5.65 eV.
- llmf contains information of iterations, check eigenvalue and fermi energies, band gap.
- rst.mp-2534 is generated. Self-consistent charge included.

- You can change lattice constant as $ALAT=1.8897268777743552*5.65/5.75$ in ctrl file. simple math operators such as $*$ $+$ $-$ $/$ $**$ can be possible in ctrl.
- Note: ctrlp is intermediate file generated by python from ctrl. Fortran calls a python code internally.(ctrl2ctrlp.py is responsible for the math)
- check save.mp-2534. Show history of lmfa and lmf. one line per iteration. Show your console options. c,x,i,h
LDA energy shown two values need to be the same (but slight difference).
Repeat lmf stops with two iteration.
- SiteInfo.lmchk : Site infor
- PlatQlat.chk : Lattice info
- estaticpot.dat : electrostatic potential of smooth part.

NOTE:

We have deguchi paper <https://sci-hub.tw/https://doi.org/10.7567/JJAP.55.051201>

All calculation is by the default setting in QSGW on the PMT method.

No empty spheres. EH=-1,EH=-2, MT radius is -3% untouched.
RSMH=RSMH2=R/2

Step 4. Create k-path and BZ for band plot

After the calculation converges, it might be necessary to make a band plot with `job_band` command explain later on. The normality of the calculation of bands can be confirmed by the band plot (for magnetic systems, check the total magnetic moment and the magnetic moment for each site).

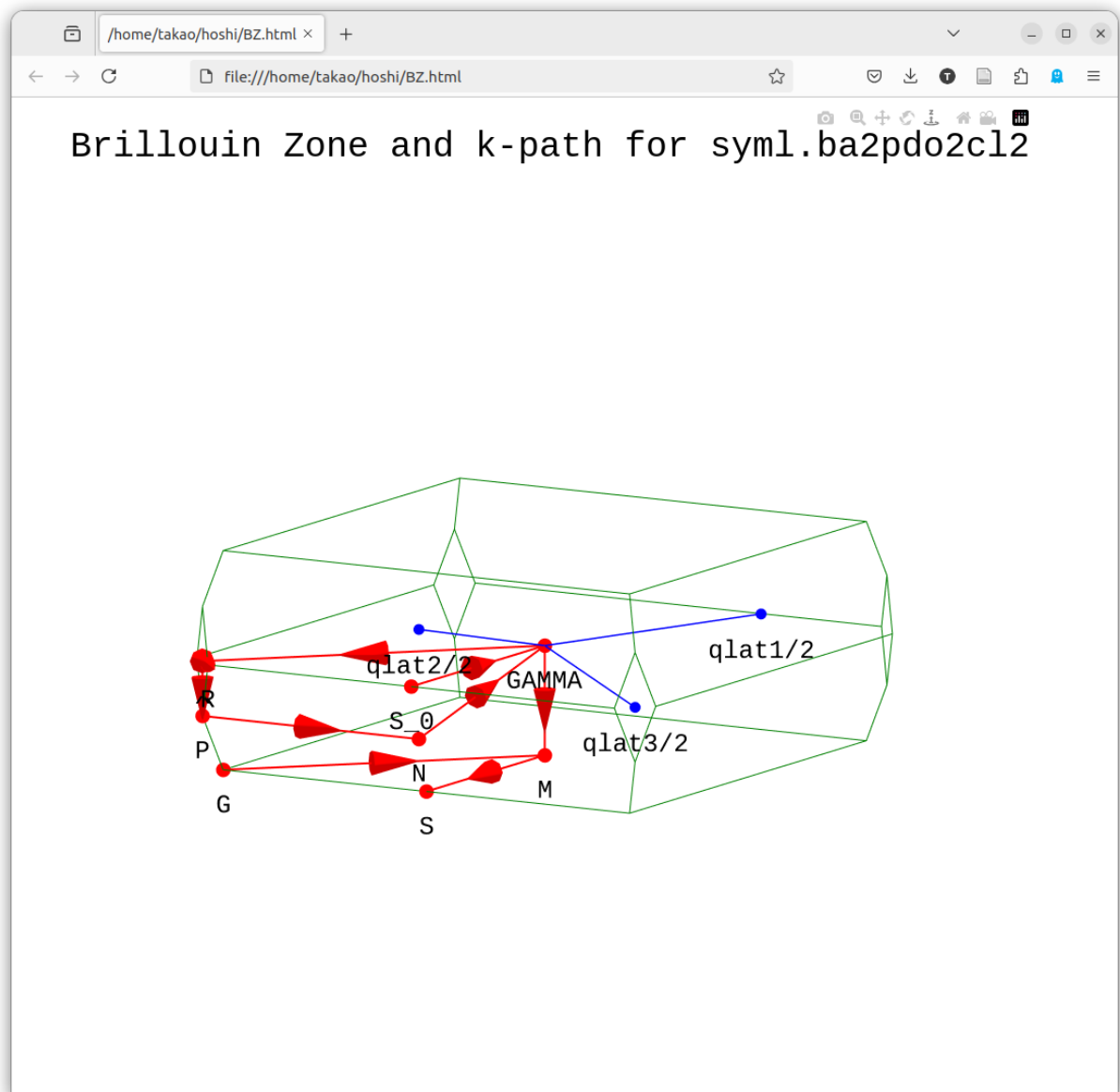
Before `job_band`, run `getsym1 gaas`. Install any missing packages with pip. It is on spglib by Togo and seekpath. After finished, view BZ.html. It shows the k-path in the BZ. We show an example for ba2pdo2cl2 in the figure below.

It is an interactive figure written with plotly, so you can read the coordinate values.

```
getsym1 foobar
```

(foobar is that in ctrl.foobar)

- Samples of BZ.html by getsym1 are seen at <https://ecalj.sakura.ne.jp/BZgetsym1/>



Step 5. band plot

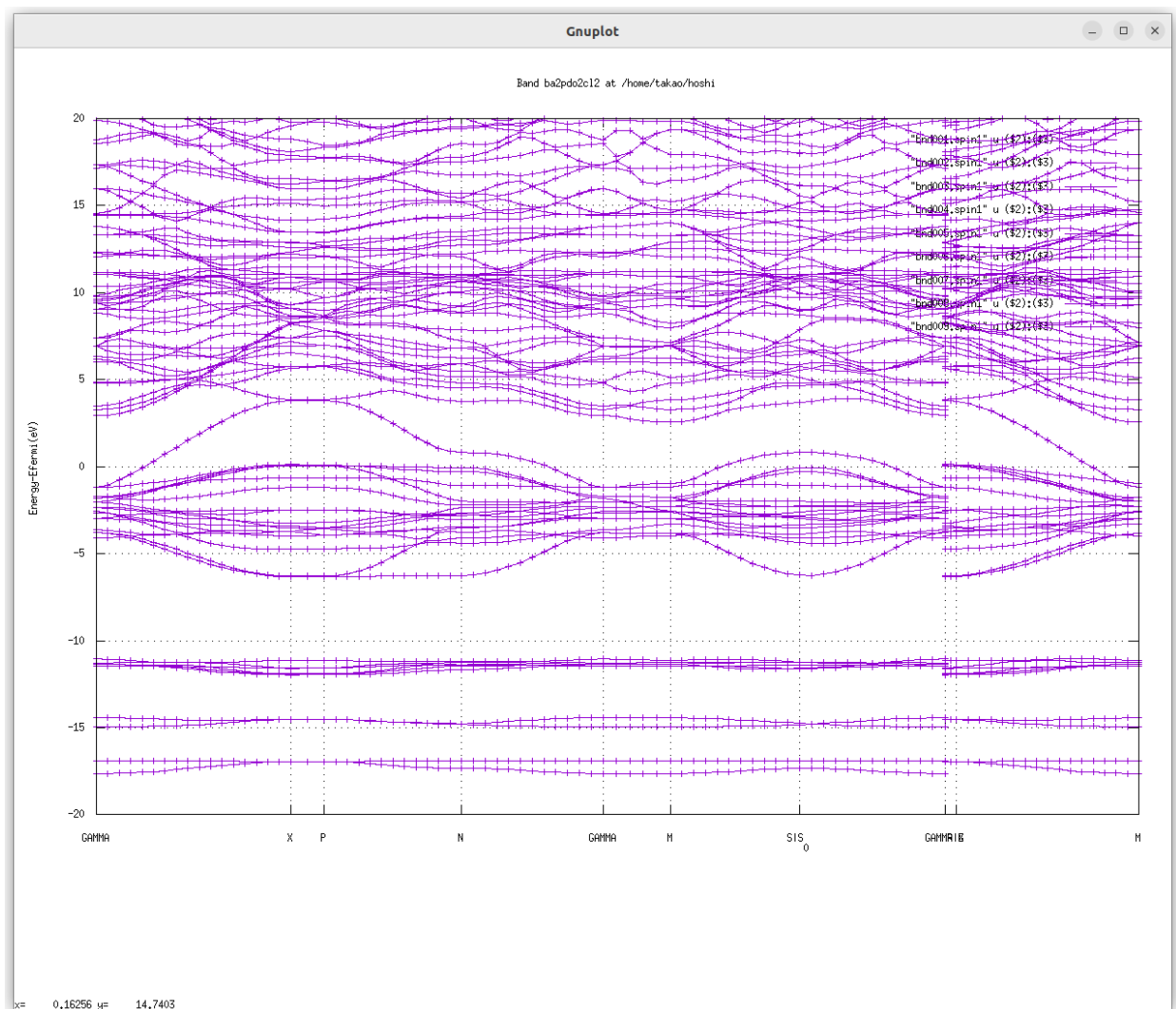
PROF

(this is a case for `ba2pdo2cl2`)

```
>job_band ctrl.ba2pdo2cl2 -np 8
```

A gnuplot script can be created. Edit it if necessary. If you edit `sym1.ba2pdo2cl2` before `job_band`, you can adjust the symmetry line and mesh size.

- The following picture is the LDA bands for the default calculation of `ba2pdo2cl2` (the names of the symmetric points can be confirmed with `BZ.html`. In addition, look into `sym1.foobar`). 0 eV is the Fermi energy. Since this is metallic, we see no band gap.



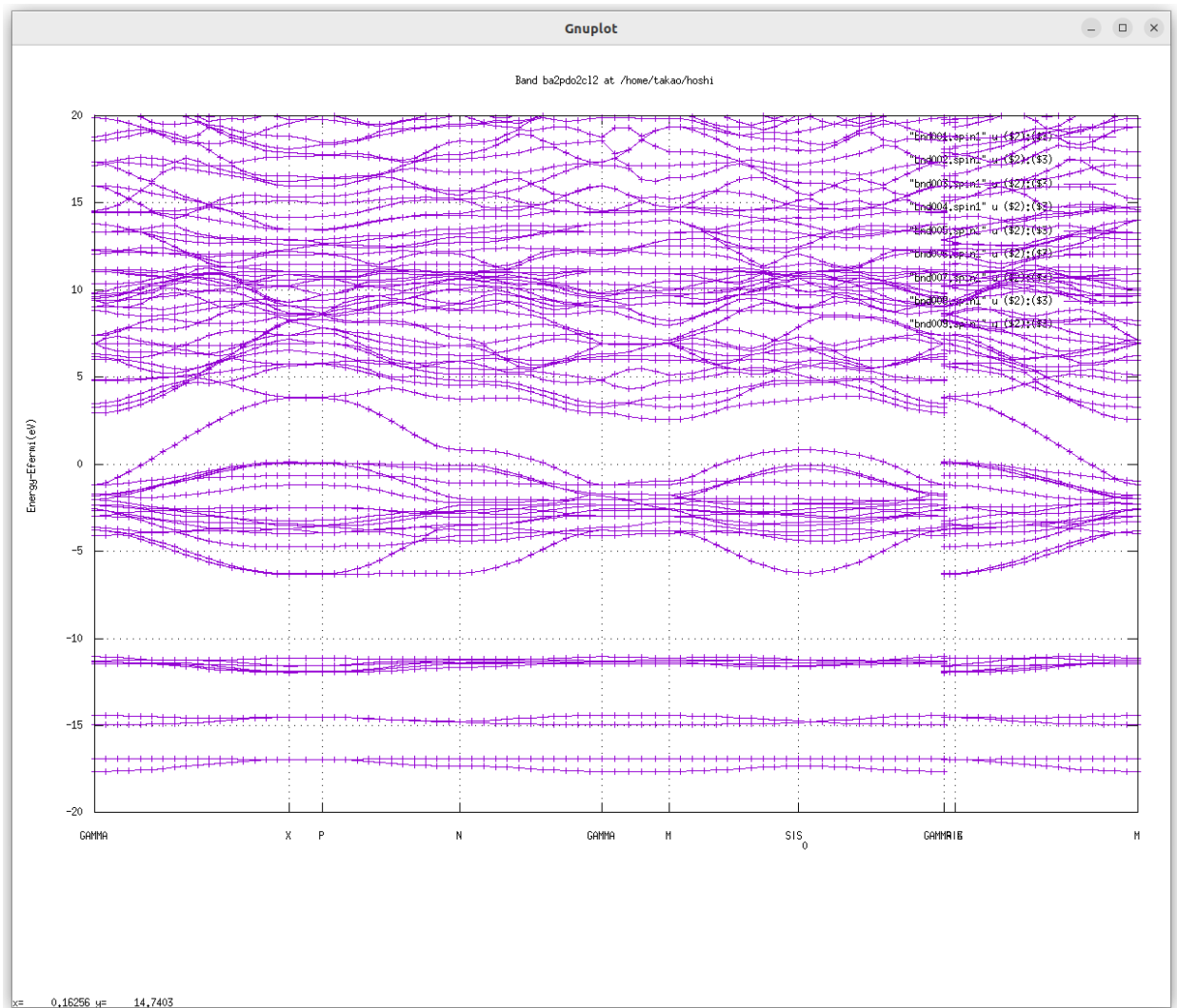
- The defaults are fine except for the k-mesh setting. For example, it is better to increase the k mesh for Fe. In general, for semiconductors, 4 4 4 for Si is a reasonable level, 6 6 6 is a level that can be used for a paper, and 8 8 8 is a level for checking accuracy. For metals such as Fe, 8 8 8 is a reasonable level.

PROF

Here we are talking about band energies.

- In ecalj, the k mesh for **lmf** (ctrl) and the k mesh for GW (n1n2n3 specified in GWinput) can be different. The former has affected little on computational time, but the latter has a large effect (thus we want to reduce **n1n2n3** in GWinput).
- In ecalj's band plot mode, theoretically degenerated bands because of symmetry at the BZ edge are not degenerated. This is because there are limited numbers of APW basis functions, so run the

band plot with pwemax=4, etc. (Temporary solution: We want to automate it).



job_tdos, job_fermisurface, job_pdos

job_pdos calculates PDOS, **job_tdos** calculates total DOS, and **job_fermisurface** draws the Fermi surface with Xcrysden.

job_fermisurface can be used to draw the shape of the CBM bottom as ellipsoid of Si.

PROF

XXX

Step 6. QSGW calculation

We now run QSGW calculations. QSGW is computationally very expensive. So we recommend you to run smaller systems at first.

For QSGW calculations, we need one additional input file **GWinput**, whose template is generated by mkGWinput.

```
mkGWinput ctrl.mp-2534
```

Then copy and edit GWinput.tmp to GWinput.

In **GWinput**, $n_1 n_2 n_3$ should be smaller than $n_{k1} n_{k2} n_{k3}$ in ctrl usually, in order to reduce computational time (1/2 or 2/3 of ctrl, for example)
 If $n_1 n_2 n_3 = 6 \times 6 \times 6$ for Si, it is reasonable. Except k points, not need to modify so much (ask us). **GWinput** is explained [here](#). Input system is different from ctrl.

flow of QSGW calculation with the script gwsc

We run the QSGW calculations with gwsc. For semiconductors, several QSGW iterations are fine, close enough to final results.

QSGW is to obtain band structures (or one-body Hamiltonian), the total energy is not yet.

QPU file contains diagonal components of GW calculations.

Note that our **Mixed Produce basis** is a key technology for the GW calculation.

```
gwsc -np NP [--phispsym] [--gpu] [--mp] nloop extension
```

(--phispsym is for magnetic materials to keep the same basis for up and down)

Then console outputs of **gwsc** is something like

```
### START gwsc: ITERADD= 1, MPI size= 4, 4 TARGET= si
===== Ititial band structure =====
--> No sigm. LDA caculation for eigenfunctions
0:00:00.226245 mpirun -np 1 /home/takao/bin/lmfa si >llmfa
0:00:00.807062 mpirun -np 4 /home/takao/bin/lmf si >llmf_lda
===== QSGW iteration start iter 1 ===
0:00:03.071054 mpirun -np 1 /home/takao/bin/lmf si --jobgw=0
>llmfgw00
0:00:03.904403 mpirun -np 1 /home/takao/bin/qg4gw --job=1 > lqg4gw
0:00:04.431022 mpirun -np 4 /home/takao/bin/lmf si --jobgw=1
>llmfgw01
0:00:05.918216 mpirun -np 1 /home/takao/bin/heftet --job=1 > leftet
0:00:06.444439 mpirun -np 1 /home/takao/bin/hbasfp0 --job=3 >lbasC
0:00:07.064558 mpirun -np 4 /home/takao/bin/hvccfp0 --job=3 > lvccC
0:00:07.812283 mpirun -np 4 /home/takao/bin/hsfp0_sc --job=3 >lsxC
0:00:08.545956 mpirun -np 1 /home/takao/bin/hbasfp0 --job=0 > lbas
0:00:09.156775 mpirun -np 4 /home/takao/bin/hvccfp0 --job=0 > lvcc
0:00:09.884064 mpirun -np 4 /home/takao/bin/hsfp0_sc --job=1 >lsx
0:00:10.644292 mpirun -np 4 /home/takao/bin/hrcxq > lrcxq
0:00:11.482931 mpirun -np 4 /home/takao/bin/hsfp0_sc --job=2 > lsc
0:00:12.460776 mpirun -np 1 /home/takao/bin/hqpe_sc > lqpe
0:00:13.019735 mpirun -np 4 /home/takao/bin/lmf si >llmf
===== QSGW iteration end iter 1 ===
OK! ===== All calclation finished for gwsc =====
```

...

The console outputs are redirected to log files `l*`. `lsxC` is the exchange self-energy due to cores. `lsx` is for exchange. `lsc` is correlation. `lvcc` is for Coulomb matrix.

In this calculation we run `gwsc -np 8 1 si`, where 1 is the number of QSGW iteration.

If you repeat `gwsc`, we have additional QSGW iterations on top the previous calculations.

a case of La₂CuO₄

For La₂CuO₄, I had

```
2025-06-27 19:09:01.465241 mpirun -np 1 echo --- Start gwsc ---
--- Start gwsc ---
option= -vssig=0.8
### START gwsc: ITERADD= 5, MPI size= 32, 32 TARGET= lcuo
===== Ititial band structure =====
--> We use existing sigm file
0:00:00.041902 mpirun -np 32 /home/takao/bin/lmf lcuo -vssig=0.8
>llmf_start
We found QPU.5 -->start to generate QPU.6...
===== QSGW iteration start iter 6 ===
0:00:18.111042 mpirun -np 1 /home/takao/bin/lmf lcuo -vssig=0.8 --
jobgw=0 >llmf gw00
0:00:18.233440 mpirun -np 1 /home/takao/bin/qg4gw -vssig=0.8 --job=1
> lqg4gw
0:00:18.488197 mpirun -np 32 /home/takao/bin/lmf lcuo -vssig=0.8 --
jobgw=1 >llmf gw01
0:00:40.910973 mpirun -np 1 /home/takao/bin/heftet --job=1 -
vssig=0.8 > leftet
0:00:41.052760 mpirun -np 1 /home/takao/bin/hbasfp0 --job=3 -
vssig=0.8 > lbasC
0:00:43.290214 mpirun -np 32 /home/takao/bin/hvccfp0 --job=3 -
vssig=0.8 > lvccC
0:01:00.327823 mpirun -np 32 /home/takao/bin/hsfp0_sc --job=3 -
vssig=0.8 >lsxC
0:01:45.547149 mpirun -np 1 /home/takao/bin/hbasfp0 --job=0 -
vssig=0.8 > lbas
0:01:46.806858 mpirun -np 32 /home/takao/bin/hvccfp0 --job=0 -
vssig=0.8 > lvcc
0:01:59.034735 mpirun -np 32 /home/takao/bin/hsfp0_sc --job=1 -
vssig=0.8 >lsx
0:02:45.526614 mpirun -np 32 /home/takao/bin/hrcxq -vssig=0.8 > lrcxq
0:06:51.996781 mpirun -np 32 /home/takao/bin/hsfp0_sc --job=2 -
vssig=0.8 > lsc
0:23:31.022636 mpirun -np 1 /home/takao/bin/hqpe_sc -vssig=0.8 >
lqpe
0:23:33.062303 mpirun -np 32 /home/takao/bin/lmf lcuo -vssig=0.8
>llmf
===== QSGW iteration end iter 6 ===== QSGW iteration start iter 7
===
0:24:12.969096 mpirun -np 1 /home/takao/bin/lmf lcuo -vssig=0.8 --
jobgw=0 >llmf gw00
...
```

This is without GPU. We see one QSGW iteration requires 24 minutes (start timing is shown at the top of lines).

Since I had 5th-QSGW iteration finished (checked by the existence of QPU.5run), it start from 6th iteration.

a case of ba2pdo2cl2.

Run

```
mkGWinput ba2pdo2cl2
```

to generate GWinput.tmp, which is a setting file for QSGW.

After copying this to GWinput, you may need to edit GWinput.

Minimum thing to edit is the number of k points for the self energy ($n_1 n_2 n_3$).

Compared with k points in ctrl (nk_1, nk_2, nk_3), we use small numbers.

(We often use 1/2 or 2/3 of k points given in ctrl as nk_1, nk_2, nk_3).

There are another setting in GWinput. However, we usually do not need to touch things except $n_1 n_2 n_3$ if you treat non-magnetic semiconductors.

Then you can run QSGW calculation with

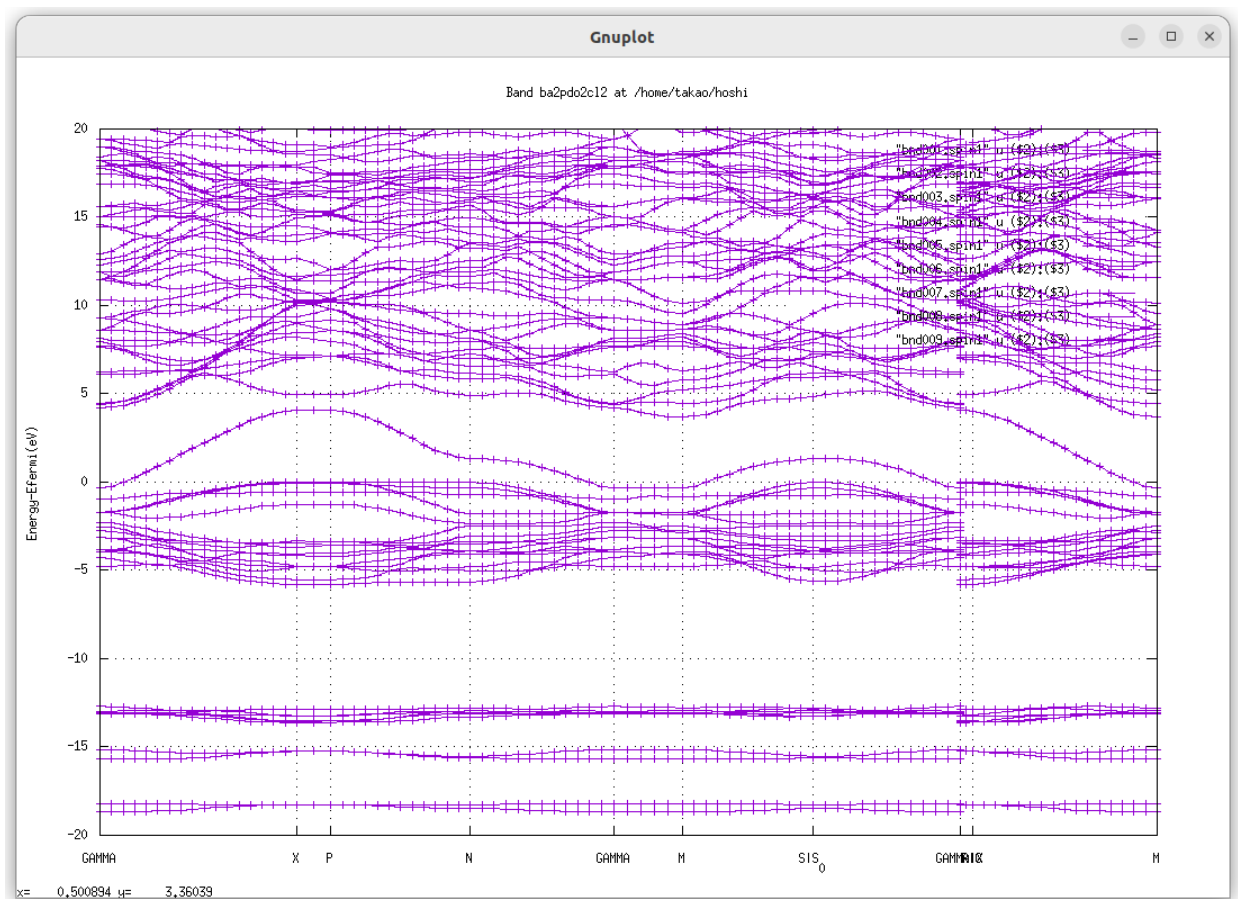
```
gwsc -np 32 1 ba2pdo2cl2
```

. Here 1 means the number of QSGW iterations. QSGW iteration is quite time-consuming. **gwsc** gives minimum help (we need to explain options elsewhere).

The iteration is kept in **rst.foobar**:electron density, **sigm.*:vxcqsgw**.

(Remove these files in addition to *run files/directories if you like to start from the beginning).

- It requires 53 minutes to run one iteration of QSGW.
- job_band ba2pdo2cl2 -np 32 gives the following picture. QSGW one-shot changes band structure around E_f from that in LDA. But still metallic, no band gaps.

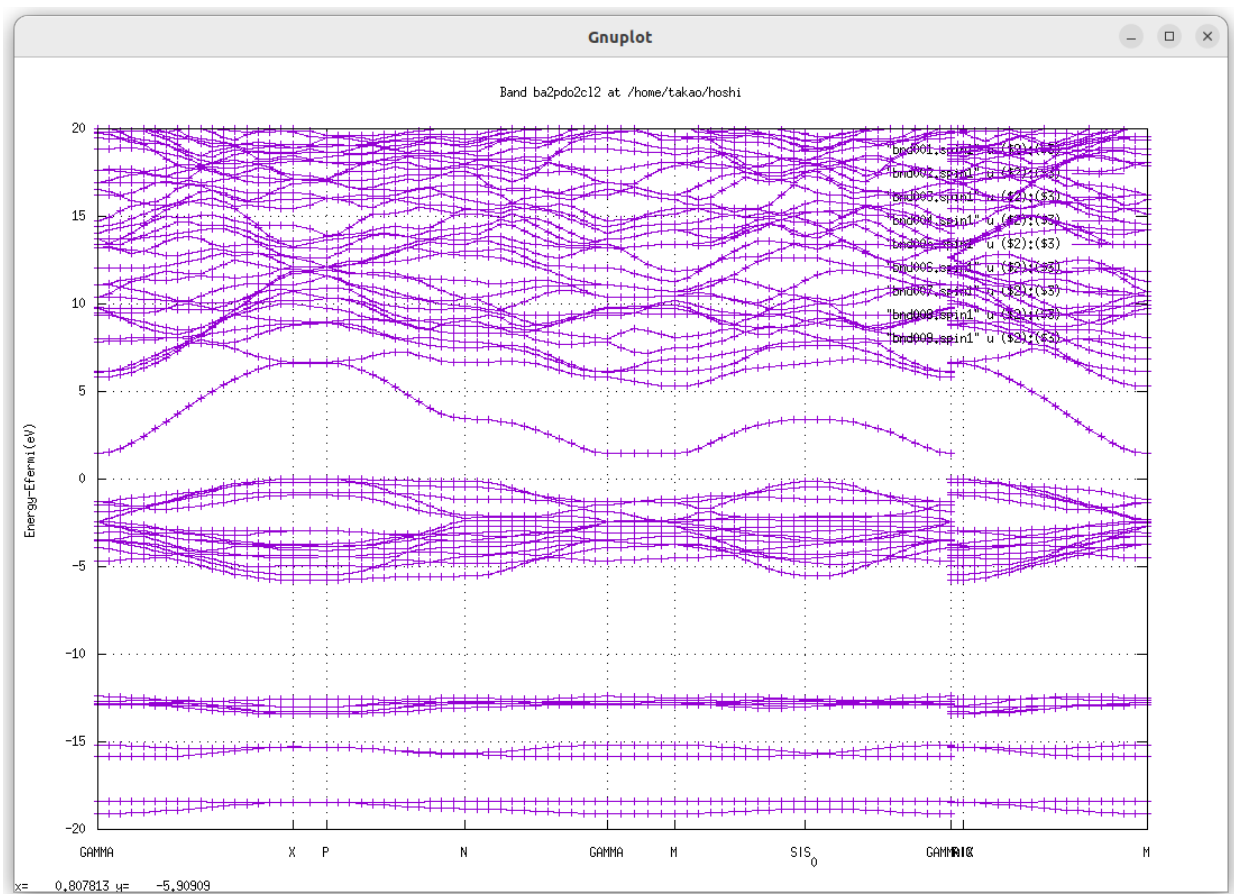


- To continue QSGW iteration, run

```
gwsc -np 32 nx ba2pdo2c12
```

Since you did 1 already. You will have the results of 1+nx QSGQ iteration.

- when we run 8 iterations as for ba2pdo2c12, we had band gap 2.1 eV. We saw band gap after 4th iteration.



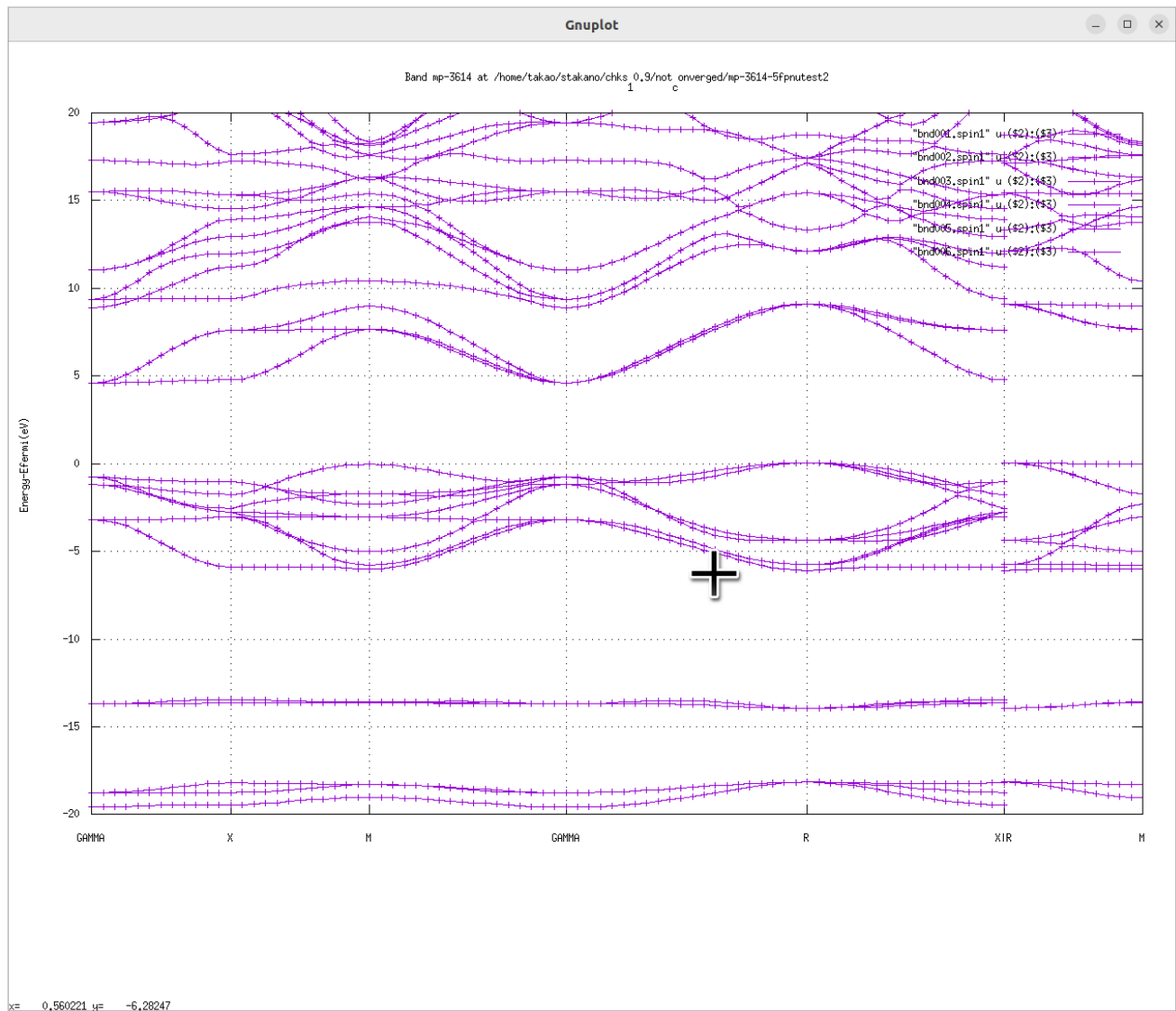
- For comparison with experiments, we recommend to use `ssig=0.8` (set in `ctrl.foobar` file), which is called as QSGW80.
- Spin-orbit coupling. After you obtain `sigm.foobar`, you set `SO=1` (LdotS scheme) and run `lmf`. Then you can include effect of SOC. Since `SO=1` is not implemented in the whole `gwsc` cycle, we have to include SOC just at the end step (We include SOC after we fix `VxcQSGW`).
- If you run

```
gwsc -np 32 5 ba2pdo2cl2 -vssig=0.8
```

, this override `ssig`, which is defined in `ctrl.ba2pdo2cl2`, in `lmf` calculations. (Check it in `save.ba2pdo2cl2`)

a case of KTaO3

Example of QSGW for KTaO₃ (perovskite, mp-3614)



lmchk

lmchk mp-2534

is to check the crystal symmetry. In addition determine MT radius. and Check the overlap of MTs. Defaults setting is with -3% overlap.(no overlap).

- symmetry
- MT overlap

If you have less symmetry rather than the symmetry of lattice for magnetic systems, you have to set crystal symmetry by hand.

This can be done by adding space group symmetry generator to SYMGRP (instead of **find**). We need to pay attention for this point in the case of SOC.

how to write space-group operation

For example r3x means 3-fold axis along x. How to express space-group operations in ecalj are explained at [SYMGRP section at lmf.md](#).

How to start over calcuations

Remove mix* rst* (mix* is mixing files)

If MT radius are changed, start over from lmfa (remove atm* files)

- As long as converged, no problem.
- If you have 3d spaghetti-like entangled bands at Ef, need caution.

jobmaterials.py: mini database for computational tests

At ecalj/MATERIALS/, type `./jobmaterials.py`. It shows a help with a list of materials.

It contains samples of simple materials. It performs LDA calculations and generates GWinput for materials.

- How to run

```
./job_materials.py
```

gives a help, showing a list of materials. Then

```
./job_materials.py Si
```

performs LDA calculation of Si at ecalj/MATERIALS/Si/. '--all' works as well instead of 'Si'.

job_materials.py works as follows for given names.

Step 1. Generate ctrl*. file for Materials.ctrls.database. (names are in DATASECTION:)

Step 2. Generate ctrl by ctrlgenM1.py

Step 3. Make directtory such as Si/ and run lmf, lmfa, mkGWinput.

- Key input files are
`ctrls.si, ctrl.si`
. See sections below. `rst.si` contains self-consistent electron density. Check iterations with the output file `save.si`. The console output of lmf is in llmf. Not need to know all the console outputs.
- Before QSGW, it is better to confirm the LDA level calculations are fine. In order to do the confirmation, band plot is convenient.
For band plot we need the symmetry line as `syml.si` which can be generated by

```
getsym1 si
```

Then run

```
job_band si -np 8
```

results band plots in the gnuplot.

This is the end of GetStarted. Goto [UsageDetailed](#)