

Evaluating orbital energies from Wannier functions

Introduction

We use to think of electronic structures in terms of individual atomic orbitals and their energies. However, crystals feature bands instead of orbitals, so we can get band energies or total energies rather than local orbital energies. In fact, orbital energies in a solid are not defined unambiguously. Two possible definitions are as follows:

- Construct Wannier functions by making a Fourier transform of the k -dependent band structure. This is equivalent to a so-called tight-binding representation of the band structure, where bands are treated as a product of overlapping orbitals localized on the atoms:

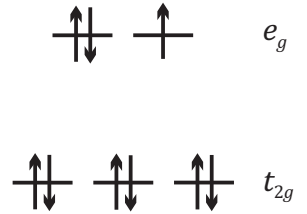
$$\hat{H}(\mathbf{k}) = \sum_{\langle ij \rangle} t_{ij} e^{i\mathbf{k}\mathbf{r}_{ij}}.$$

Here, \mathbf{r}_{ij} connects neighboring atoms, where the orbitals are centered. For $\mathbf{r}_{ij} = 0$, $t_0 = \varepsilon_0$ is the on-site energy or orbital energy.

- Artificially populate different orbitals and compare the resulting total energies. Standard LDA leaves no room for manipulating orbital occupations. However, the so-called LSDA+ U method entails an additional energy term that is dependent on orbital occupations and provides a simple “handle” to control the orbital state. This is essential for strongly correlated insulators, where LDA fails to reproduce even basic features of the electronic structure.

This tutorial will show how the former technique can be realized in FPLO. For LSDA+ U , see another tutorial.

In both tutorials, we will consider a simple magnetic insulator CuSb_2O_6 and analyze orbital energies of Cu^{2+} . Standard crystal-field theory suggests that the octahedral local environment of Cu^{2+} splits five $3d$ orbitals into the t_{2g} and e_g levels. Then nine electrons are distributed over these five orbitals and form an orbitally degenerate state, where one of the e_g orbitals is half-filled (1 electron) and the other one is fully filled (2 electrons). Weak distortions of the CuO_6 octahedron will lift this degeneracy, but it is hard to say *a priori*, which of the e_g orbitals will have lower energy.



1 LDA and Wannier functions

1.1. Set up the calculation

Create a new directory

- Create a new directory for this tutorial: e.g., `CuSb206`
- Create a sub-directory: e.g., `CuSb206/lda`

Main menu

- `K-mesh subdivison : 6 6 6`
The default $12 \times 12 \times 12$ mesh will take much longer to converge. Even if you need a finer k -mesh, always start with a reasonably small number of k -points and increase it later. This improves convergence and saves computing time

Symmetry

- Space group: $P4_2/mnm$ (136)
- Lattice constants: $a = b = 4.6291 \text{ \AA}$, $c = 9.2882 \text{ \AA}$
- Cu (0,0,0)
- Sb (0,0,0.33249)
- O (0.3095,0.3095,0)
- O (0.3014,0.3014,0.3271)

1.2. Run the self-consistent calculation

- `fplo9.09-43-x86_64 > out` (≈ 15 mins)

In the meantime, read through the rest of the tutorial. In particular, pay attention to 1.4, where you will have to define local coordinate axes. This may take a while.

- Check if the calculation is converged: `grep "st de" out`

1.3. Calculate density of states and band structure

- `fedit9.09-43-x86_64`, go to the Bandplot menu

- Bandstructure plot: check

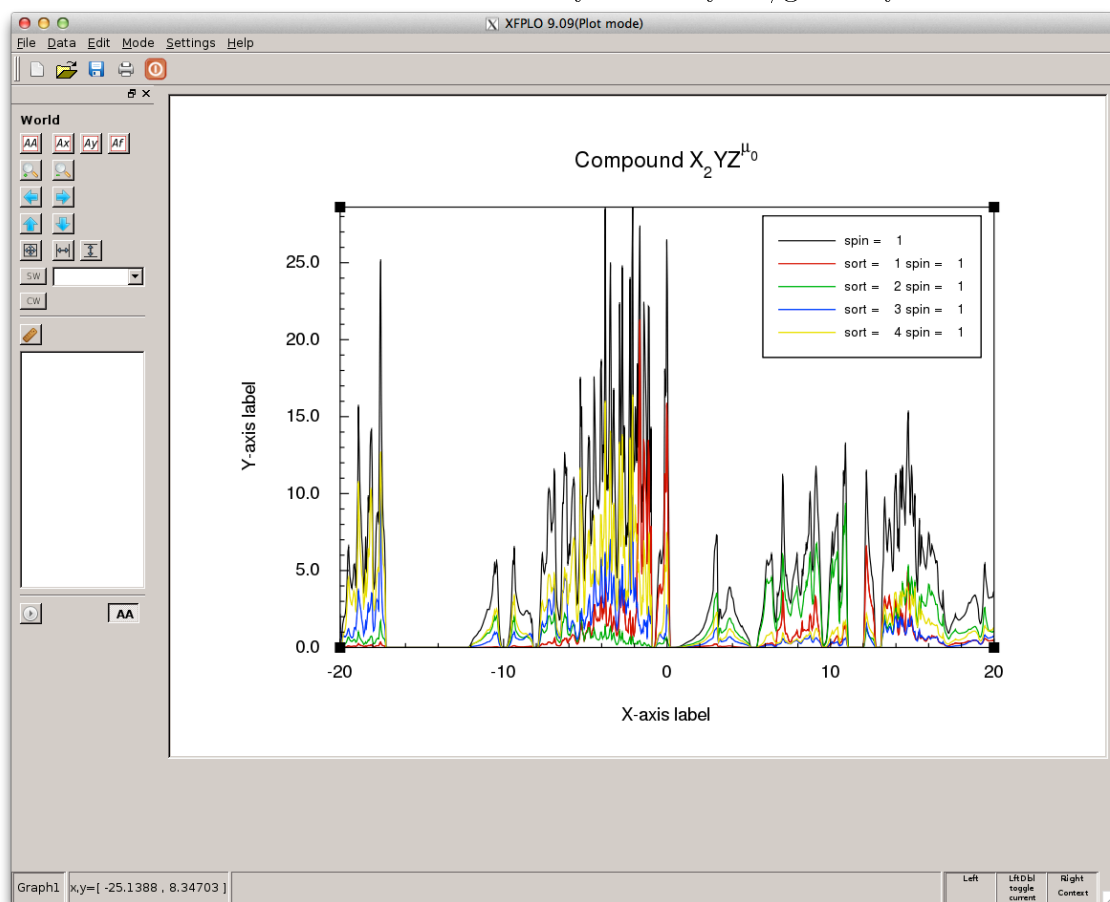
- `fplo9.09-43-x86_64 > out-bandplot` (≈ 5 min)

Compare the iteration time with and without the calculation of the band structure. We could choose to calculate band structure from the very beginning, but in that case DOS will be produced on each step, thus slowing down the calculation. This becomes increasingly important for large crystal structures.

- Plot total and atom-resolved DOS:

`xfplqt4 +dos.total +dos.sort001 +dos.sort002 +dos.sort003 +dos.sort004`

What are the states at the Fermi level? Can you already see/guess crystal-field levels of Cu^{2+} ?



- Plot the band structure: `xfbp +band`. How many bands cross the Fermi level? How many bands do you expect to see there? Compare the DOS and bandplot pictures with the scheme of crystal-field levels on the first page.

1.4. Analyze orbital contributions to the Cu 3d DOS

- Look at the crystal structure and figure out the directions of local axes.
The CuO₆ octahedra feature two Cu–O distances of 2.026 Å and four Cu–O distance of 2.066 Å. Therefore, it is natural to put z along the 2.026-Å bond and x along one of the 2.066-Å bonds. Then the xy , yz , and xz states will be t_{2g} ; the $x^2 - y^2$ and $3z^2 - r^2$ states will be e_g .
Note that we will need the x and z directions evaluated in the Cartesian system (same coordinate system as in the `out` file)
- `fedit9.09-43-x86_64`, go to the Bandplot menu:
 - Local DOS sites: 1 or 2
 - Transform quant.axis: check
 - X-Axis, Z-Axis: provide the vectors you found
 - Weights: check
 - Lower energy bound, Upper energy bound: choose appropriate values based on the DOS plot
 - Restrict bands to window: check

Notes:

1. Different Cu atoms may have different local coordinate axes (and in our case they are different indeed). Therefore, you have to specify the number of an atom in the unit cell, not the number of a Wyckoff position. Relevant numbers can be found in the `out` file (search for the words “UNIT CELL CREATION”)
 2. At this point, it makes sense to set the lower and upper boundaries and restrict the output to this energy window, because we are not interested in the bands at very low and very high energies.
- Run the calculation, orbital-resolved DOS will be written into the `+ldos.site00x.nl00y` files. You can figure out what “nl00y” means by reading the header of each file. In fact, FPLO simply calculates orbital-resolved DOS for all valence states of a given atom.
 - Plot orbital-resolved DOS. Make sure it follows the anticipated crystal-field picture: the $d_{x^2-y^2}$ and $d_{3z^2-r^2}$ states (e_g) should lie higher than d_{xy} , d_{yz} , d_{xz} (t_{2g}). The notation is as follows:

$$m = -2 \longrightarrow d_{xy}$$

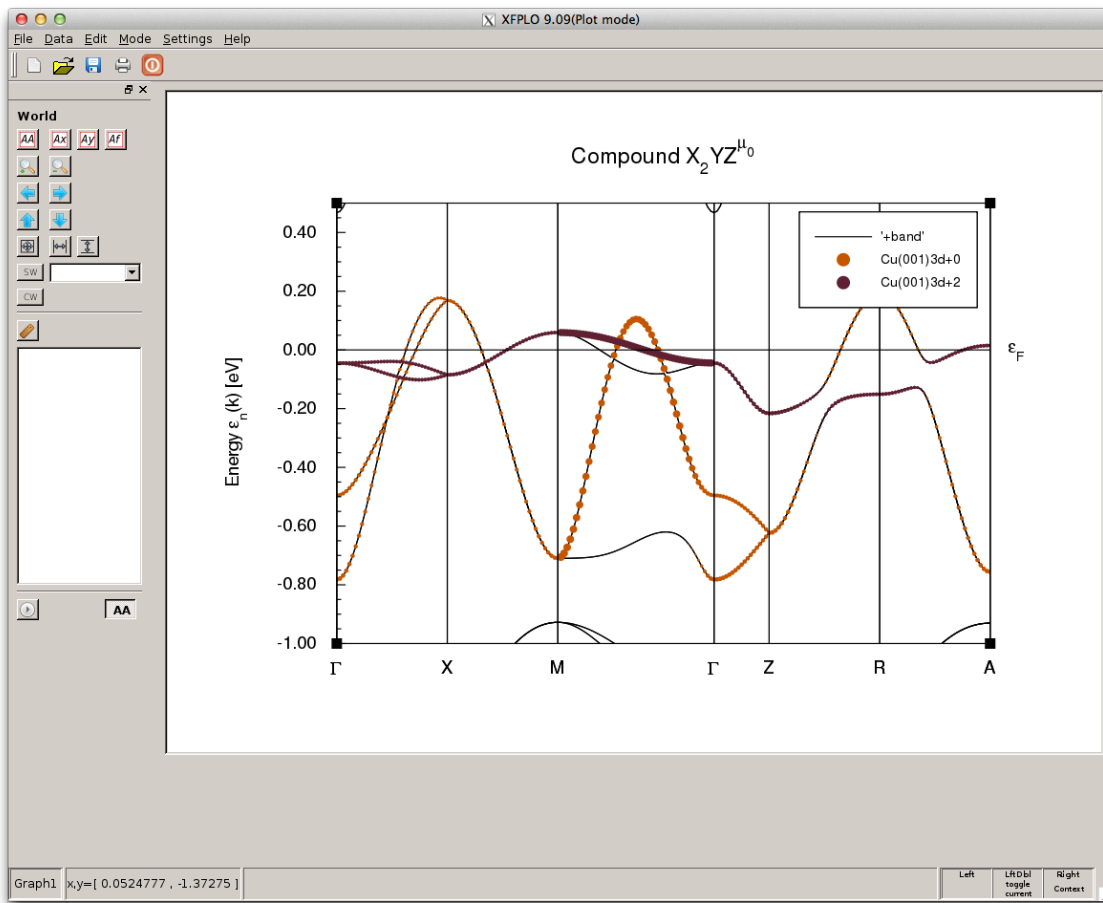
$$m = -1 \longrightarrow d_{yz}$$

$$m = 0 \longrightarrow d_{3z^2-r^2}$$

$$m = 1 \longrightarrow d_{xz}$$

$$m = 2 \longrightarrow d_{x^2-y^2}$$

- Plot bandweights for $d_{x^2-y^2}$ and $d_{3z^2-r^2}$. Run `xfbp: +bweights`. By default, it shows 3s states of Cu. Choose the relevant d -states instead.
This is now a more stringent test of your local coordinate system. Different bands should have different characters. If the characters are mixed, your local coordinate system is wrong. Try again!



1.5. Construct Wannier functions

- Use the sample `=.wandef` file. Its structure is described in a separate manual, which is available on the FPLO website. The minimum information you should know is that:
 - the definition of each Wannier function is preceded by **wandef** and **on**
 - the parameters **emin**, **emax**, and **de** describe the Wannier function as a whole
 - contributions of individual atomic orbitals are preceded by the word **contrib**. We have a simple case, where each Wannier function is based on a single atomic orbital only.
- Wannier functions are obtained by a Fourier transformation of the band structure in a given energy range. You have to define this energy range and, naturally, decide on two things:
 - Number of Wannier functions (equals to number of bands in the relevant energy range)
 - Symmetries of Wannier functions. Each Wannier function is built on a single atomic orbital or their combination (incl. molecular orbitals). It is essentially your decision which starting orbitals to use, depending on what kind of information you want to get

Here, we are interested in the energies of individual d -levels. Therefore, we choose atomic d -orbitals as a starting point for the construction of Wannier functions.

- Now, we have two options:
 - Restrict ourselves to only two orbitals ($d_{x^2-y^2}$, $d_{3z^2-r^2}$) that contribute to the bands between -0.8 eV and 0.2 eV: 2 orbitals per Cu atom \times 2 Cu atoms in a cell = 4 Wannier functions, the energy range from -0.8 eV to 0.2 eV
 - Consider all five d -orbitals of a Cu atom: 5 orbitals per Cu atom \times 2 Cu atoms in a cell = 10 Wannier functions, the energy range from ~ -2 eV to 0.2 eV.

The first option looks more attractive, because we have an isolated band complex. In the second option, we have to decide on the lower boundary of the energy range (there is no clear border between the Cu 3*d* and O 2*p* states, -2 eV is a rough guess only). However, the first option is also more difficult to implement.

- Suppose we want to make Wannier functions for 4 bands in the vicinity of the Fermi level (more precisely, for bands between -0.8 eV and 0.2 eV). Therefore, we should specify 4 Wannier functions, two for each of the Cu sites.

- **emin**: -0.8, **emax**: 0.2
- **de**: 0.1
- **site**: 1 or 2
- **xaxis**, **zaxis**: use the local coordinate system(s) determined and tested in 1.4.
- **orb**: 3d+2 or 3d+0

Notes:

1. The parameter **de** defines the contributions of states below **emin** and above **emax**. Here, we have a relatively simple situation, where four bands are isolated from the rest of the band structure. By choosing a small **de**, we restrict ourselves to this energy window and truncate all bands that lie beyond it.
2. Cu1 and Cu2 require different *x* and *z* vectors

For 10 Wannier functions, you need to do the same, but:

- change **emin** to -2 eV
- increase **de**
- use all five *d* orbitals: 3d-2, 3d-1, 3d+0, 3d+1, 3d+2
- Run `fplo9.09-43-x86_64 > out` for 4 Wannier functions. You will likely find that FPLO stops and complains about the symmetries. Indeed, your input should be compatible with the symmetry of the crystal structure. The problem is that in our case the $d_{x^2-y^2}$ and $d_{3z^2-r^2}$ orbitals do not always form their own subspace under symmetry transformations of the crystal structure. Therefore, you should choose one of the following:
 - Stick to only 2 orbitals per Cu atom and 4 Wannier functions in total, with
x-axis = 1, -1, 1.414213562
z-axis = 1, 1, 0 for Cu1
 For Cu2, change signs accordingly
 - Opt for 5 orbitals per Cu atom, i.e., 10 Wannier functions in total. Then you can use the local axes determined earlier in this tutorial

Ideally, you should try to do both and compare the results.

Note: you have probably realized that $\mathbf{x} = (1, -1, \sqrt{2})$. Then $\mathbf{y} = [\mathbf{z} \times \mathbf{x}] = \sqrt{2} (1, -1, -\sqrt{2})$. This ensures that the *z*-projection of our *x* vector is equal to its *xy*-projection. This is required to match the σ_z symmetry of the crystal structure (mirror plane perpendicular to the *c* axis). This mirror plane transforms $\mathbf{x} = (1, -1, \sqrt{2})$ to $(1, -1, -\sqrt{2}) \sim \propto \mathbf{y}$. If we choose any other *x* vector, such as $\mathbf{x} = (1, -1, 1.5)$, its symmetry-transformed version $(1, -1, -1.5)$ will no longer match *y*, and FPLO will produce an error.

- If you were successful with setting up the `=.wandef` file, FPLO will make 1-2 iterations and write the `+wancoeff` file. Now you can calculate Wannier functions by running FPLO again: `fplo9.09-43-x86_64 > out-Wannier`. We use a different out-file, because it refers to the Wannier fit, not to the standard self-consistent cycle.

- Check the quality of the Wannier fit to the band structure: `xfplot4 +band +wanband`
Now you get a picture with two sets of bands: black lines are from LDA, red lines are from the Wannier functions. They should match well. If they don't, check your energy windows (`emin`, `emax`) in the Wannier definition file

- Inspect the `out-Wannier` file. It contains a list of hopping parameters t_{ij} :

`T=` is the Cu–Cu bond vector (Cartesian coordinates, Bohr radii units)

`hop=` is the hopping integral in eV

The energies at $T = 0$ are on-site orbital energies. They give us a measure of crystal-field levels of Cu^{2+} in CuSb_2O_6

- You should find that the $d_{x^2-y^2}$ and $d_{3z^2-r^2}$ orbitals have similar energies, but different bandwidths:

$$\varepsilon_{x^2-y^2} = -0.078 \text{ eV}, |t_{x^2-y^2}| < 0.022 \text{ eV}$$

$$\varepsilon_{3z^2-r^2} = -0.227 \text{ eV}, |t_{3z^2-r^2}| = 0.197 \text{ eV}$$

The $d_{3z^2-r^2}$ orbitals interact strongly along the z direction, hence larger bandwidth. If you can see this from your `out-Wannier` file, you were good so far and can probably construct Wannier functions for more complex situations.