wannier90: Tutorial

Version 2.0

XXth November 2012

Contents

TO DO:

- IS: For same reason, we should think of using bibtex for the growing number references (to automatically order them). GP: I agree.
- GP: would it be good to reduce the number of pages, since now manual and tutorials are becoming quite long? E.g., I was thinking to remove the cleardoublepage commands (now that we also have a table of contents). I also reduced the margins a little bit.

Preliminaries

Welcome to wannier90! The examples contained in this tutorial are designed to help you become familiar with the procedure of generating, analysing and using maximally-localised Wannier functions (MLWFs). As a first step, install wannier90 following the instructions in the README file of the wannier90 distribution. For an introduction to the theory underlying MLWFs, you are encouraged to refer to the brief overview given in the wannier90 User Guide [1], to the two seminal papers of Refs. [2, 3], a recent review article [**?**] and to a paper [4] describing wannier90.

The following additional programs may be installed in order to visualise the output of wannier90 (they are optional, not all of them are necessary)

- gnuplot is used to plot bandstructures. It is available for many operating systems and is often installed by default on Unix/Linux distributions
  http://www.gnuplot.info
- xmgrace may also be used to plot bandstructures.
  http://plasma-gate.weizmann.ac.il/Grace
- XCrySDen is used to visualise crystal structures, MLWFs, and Fermi surfaces. It is available for Unix/Linux, Windows (using cygwin), and OSX. To correctly display files from wannier90, version 1.4 or later must be used.
  http://www.xcrysden.org
- vmd can also be used to visualise crystal structures and MLWFs.
  http://www.ks.uiuc.edu/Research/vmd
- python with the numpy and matplotlib modules (used e.g. in example 18 on Berry curvature)
  http://www.python.org
  http://www.numpy.org
  http://matplotlib.org

Parallel execution

Presently, the wannier90 is instead a serial-only executable, so it cannot be run in parallel using MPI libraries. On the contrary, the postw90 executable can be run in parallel to speed up the calculations, using the MPI libraries.

To enable the parallel version to be built, you must specify some flags in the `make.sys` file of `wannier90` and `postw90`; for further information, please refer to the `README.install` file in the top directory of the `wannier90` distribution.

Then, to run e.g. with 8 processors, you typically need to run a command similar to `postw90` as follows:

```
mpirun -np 8 postw90.x seedname
```

(the `mpirun` command and its flags may differ depending on the MPI libraries installed on your system: refer to your MPI manual and/or to your system administrator for further information).

About this tutorial

The first part of this tutorial comprises four examples taken from Refs. [2, 3]: gallium arsenide, lead, silicon and copper. All of the `wannier90` input files have been provided.

The second part of the tutorial covers the generation of `wannier90` input files starting from a full electronic structure calculation. We have provided input files for the PWSCF interface (`http://www.quantum-espresso.org`) to `wannier90`. say that new versions of QE don't need to be patched? Say that new features that require uHu will not be ported back to QE prior to 5.0? Therefore, you will need to install and compile elements of the `quantum-espresso` package, namely `pw.x` and `pw2wannier90.x`, in order to run these examples. Please visit `http://www.quantum-espresso.org` to download the package, and for installation instructions.

There are Interfaces to a number of other electronic structure codes including ABINIT (`http://www.abinit.org`), FLEUR (`http://www.flapw.de`), VASP (`http://www.vasp.at`), and WIEN2K (`http://www.wien2k.at`)

For images of MLWFs, see our gallery at `http://www.wannier.org/gallery.html`. If you have any images that you would like to submit to the gallery, please email us.

Contact us

If you have any suggestions regarding ways in which this tutorial may be improved, then send us an email.

For other questions, email the `wannier90` forum at `wannier@quantum-espresso.org`. Note that first you will need to register in order to post emails. Emails from non-registered users are deleted automatically. You can register by following the links at
`http://www.wannier.org/forum.html`.

1: Gallium Arsenide – MLWFs for the valence bands

- Outline: *Obtain and plot MLWFs for the four valence bands of GaAs.*
- Generation details: *From* PWSCF*, using norm-conserving pseudopotentials and a 2×2×2 k-point grid. Starting guess: four bond-centred Gaussians.*
- Directory: `examples/example1/`
- Input Files
  - `gaas.win` *The master input file*
  - `gaas.mmn` *The overlap matrices* $\mathbf{M}^{(\mathbf{k},\mathbf{b})}$
  - `gaas.amn` *Projection* $\mathbf{A}^{(\mathbf{k})}$ *of the Bloch states onto a set of trial localised orbitals*
  - `UNK00001.1` *The Bloch states in the real space unit cell. For plotting only.*

1. Run `wannier90` to minimise the MLWFs spread

   ```
   wannier90.x gaas
   ```

   Inspect the output file `gaas.wout`. The total spread converges to its minimum value after just a few iterations. Note that the geometric centre of each MLWF lies along a Ga-As bond, slightly closer to As than Ga. Note also that the memory requirement for the minimisation of the spread is very low as the MLWFs are defined at each k-point by just the 4×4 unitary matrices $\mathbf{U}^{(\mathbf{k})}$.

2. Plot the MLWFs by adding the following keywords to the input file `gaas.win`

   ```
   wannier_plot = true
   ```

   and re-running `wannier90`. To visualise the MLWFs we must represent them explicitly on a real space grid (see Ref. [1]). As a consequence, plotting the MLWFs is slower and uses more memory than the minimisation of the spread. The four files that are created (`gaas_00001.xsf`, etc.) can be viewed using XCrySDen,[1] e.g.,

   ```
   xcrysden -xsf gaas_00001.xsf
   ```

   For large systems, plotting the MLWFs may be time consuming and require a lot of memory. Use the keyword `wannier_plot_list` to plot a subset of the MLWFs. E.g., to plot the 1st, 2nd and 7th MLWFs use

   ```
   wannier_plot_list = 1 2 7
   ```

   The MLWFs are plotted in a supercell of the unit cell. The size of this supercell is set through the keyword `wannier_plot_supercell`. The default value is 2 (corresponding to a supercell with eight times the unit cell volume). We recommend not using values great than 3 as the memory and computational cost scales cubically with supercell size.

   Plot the 3rd MLWFs in a supercell of size 3. Choose a low value for the isosurface (say 0.5). Can you explain what you see?

   Hint: For a finite k-point mesh, the MLWFs are in fact periodic and the period is related to the spacing of the k-point mesh. For mesh with n divisions in the i^{th} direction in the Brillouin zone, the MLWFs "live" in a supercell n times the unit cell.

[1] Once XCrySDen starts, click on `Tools` → `Data Grid` in order to specify an isosurface value to plot.

2: Lead – Wannier-interpolated Fermi surface

- Outline: *Obtain MLWFs for the four lowest states in lead. Use Wannier interpolation to plot the Fermi surface.*
- Generation Details: *From* PWSCF*, using norm-conserving pseudopotentials and a 4×4×4 k-point grid. Starting guess: atom-centred sp^3 hybrid orbitals*
- Directory: `examples/example2/`
- Input Files
  - `lead.win` *The master input file*
  - `lead.mmn` *The overlap matrices* $\mathbf{M}^{(\mathbf{k},\mathbf{b})}$
  - `lead.amn` *Projection* $\mathbf{A}^{(\mathbf{k})}$ *of the Bloch states onto a set of trial localised orbitals*
  - `lead.eig` *The Bloch eigenvalues at each k-point. For interpolation only*

The four lowest valence bands in lead are separated in energy from the higher conduction states (see Fig. 1). The MLWFs of these states have partial occupancy. MLWFs describing only the occupied states would be poorly localised.

1. Run `wannier90` to minimise the MLWFs spread

   ```
   wannier90.x lead
   ```

   Inspect the output file `lead.wout`.

2. Use Wannier interpolation to generate the Fermi surface of lead. Rather than re-running the whole calculation we can use the unitary transformations obtained in the first calculation and restart from the plotting routine. Add the following keywords to the `lead.win` file:

   ```
   restart = plot
   fermi_energy = 5.2676
   fermi_surface_plot = true
   ```

   and re-run `wannier90`. The value of the Fermi energy (5.2676 eV) was obtained from the initial first principles calculation. `wannier90` calculates the band energies, through wannier interpolation, on a dense mesh of k-points in the Brillouin zone. The density of this grid is controlled by the keyword `fermi_surface_num_points`. The default value is 50 (i.e., 50^3 points). The Fermi surface file `lead.bxsf` can be viewed using XCrySDen, e.g.,

   ```
   xcrysden -bxsf lead.bxsf
   ```

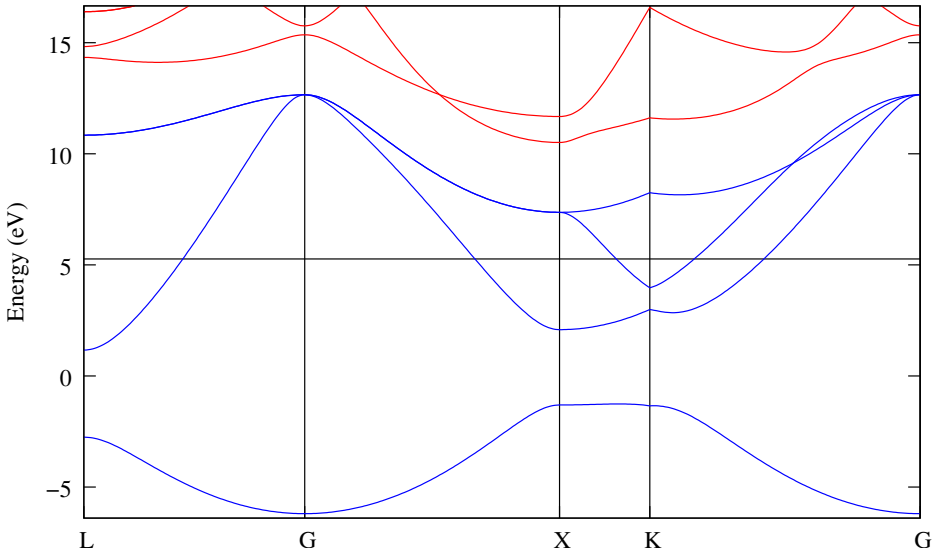

Figure 1: Bandstructure of lead showing the position of the Fermi level. Only the lowest four bands are included in the calculation.

3: Silicon – Disentangled MLWFs

- Outline: *Obtain disentangled MLWFs for the valence and low-lying conduction states of Si. Plot the interpolated bandstructure*
- Generation Details: *From* PWSCF*, using norm-conserving pseudopotentials and a 4×4×4 k-point grid. Starting guess: atom-centred sp^3 hybrid orbitals*
- Directory: `examples/example3/`
- Input Files
  - `silicon.win` *The master input file*
  - `silicon.mmn` *The overlap matrices* $\mathbf{M}^{(\mathbf{k},\mathbf{b})}$
  - `silicon.amn` *Projection* $\mathbf{A}^{(\mathbf{k})}$ *of the Bloch states onto a set of trial localised orbitals*
  - `silicon.eig` *The Bloch eigenvalues at each k-point*

The valence and lower conduction states can be represented by MLWFs with sp^3-like symmetry. The lower conduction states are not separated from the higher states by an energy gap. In order to form localised WF, we use the disentanglement procedure introduced in Ref. [3]. The position of the inner and outer energy windows are shown in Fig. 2.

1. Run `wannier90`.

   ```
   wannier90.x silicon
   ```

   Inspect the output file `silicon.wout`. The minimisation of the spread occurs in a two-step procedure [3]. First, we minimise $\Omega_{\rm I}$ – this is the extraction of the optimal subspace in the disentanglement procedure. Then, we minimise $\Omega_{\rm D} + \Omega_{\rm OD}$.

2. Plot the energy bands by adding the following commands to the input file `silicon.win`

   ```
   restart = plot
   bands_plot = true
   ```

   and re-running `wannier90`. The files `silicon_band.dat` and `silicon_band.gnu` are created. To plot the bandstructure using gnuplot

   ```
   myshell> gnuplot
   gnuplot> load 'silicon_band.gnu'
   ```

   The k-point path for the bandstructure interpolation is set in the `kpoint_path` block. Try plotting along different paths.

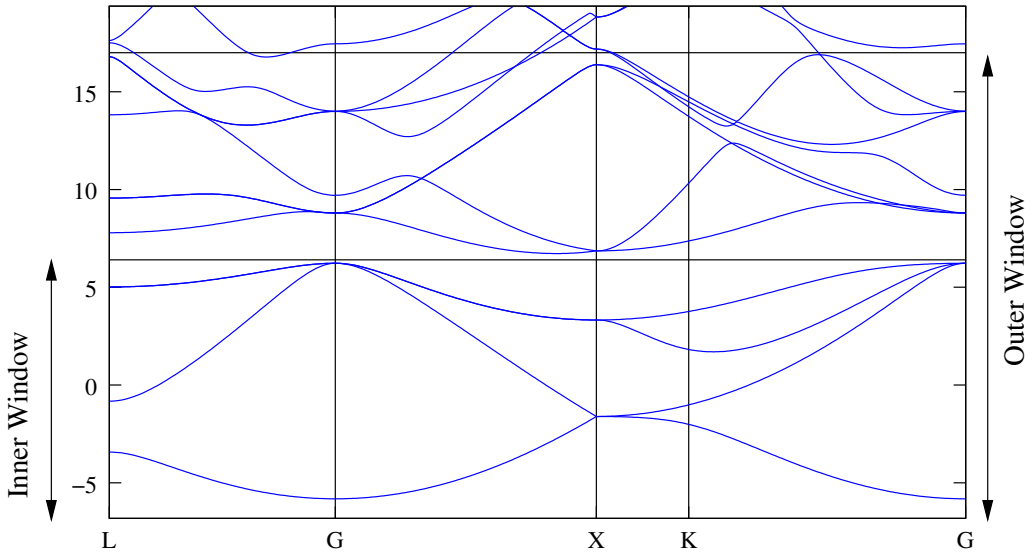

Figure 2: Bandstructure of silicon showing the position of the outer and inner energy windows.

4: Copper – Fermi surface, orbital character of energy bands

- Outline: *Obtain MLWFs to describe the states around the Fermi-level in copper*
- Generation Details: *From* PWSCF*, using ultrasoft pseudopotentials [5] and a 4×4×4 k-point grid. Starting guess: five atom-centred d orbitals, and two s orbitals centred on one of each of the two tetrahedral interstices.*
- Directory: `examples/example4/`
- Input Files
  - `copper.win` *The master input file*
  - `copper.mmn` *The overlap matrices* $\mathbf{M}^{(\mathbf{k},\mathbf{b})}$
  - `copper.amn` *Projection* $\mathbf{A}^{(\mathbf{k})}$ *of the Bloch states onto a set of trial localised orbitals*
  - `copper.eig` *The Bloch eigenvalues at each k-point*

1. Run `wannier90` to minimise the MLWFs spread

   ```
   wannier90.x copper
   ```

   Inspect the output file `copper.wout`.

2. Plot the Fermi surface, it should look familiar! The Fermi energy is at 12.2103 eV.

3. Plot the interpolated bandstructure. A suitable path in k-space is

   ```
   begin kpoint_path
   G 0.00 0.00 0.00 X 0.50 0.50 0.00
   X 0.50 0.50 0.00 W 0.50 0.75 0.25
   W 0.50 0.75 0.25 L 0.00 0.50 0.00
   L 0.00 0.50 0.00 G 0.00 0.00 0.00
   G 0.00 0.00 0.00 K 0.00 0.50 -0.50
   end kpoint_path
   ```

4. Plot the contribution of the interstitial WF to the bandstructure. Add the following keyword to `copper.win`

   ```
   bands_plot_project = 6,7
   ```

   The resulting file `copper_band_proj.gnu` can be opened with gnuplot. Red lines correspond to a large contribution from the interstitial WF (blue line are a small contribution; ie a large d contribution).

Investigate the effect of the outer and inner energy window on the interpolated bands.

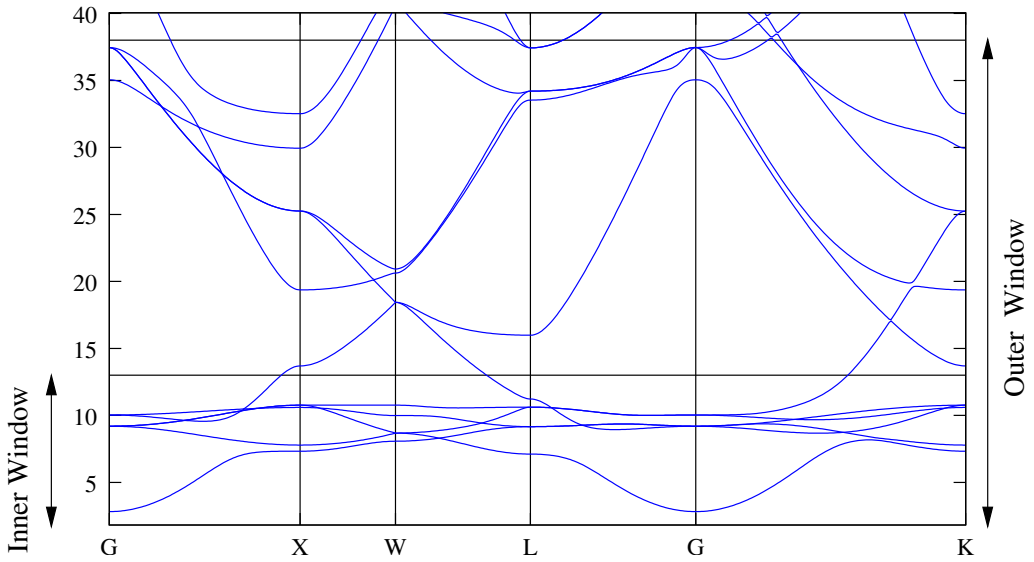

Figure 3: Bandstructure of copper showing the position of the outer and inner energy windows.

Examples Using the PWSCF Interface

The PWSCF plane-wave, density-functional theory code, which is available as part of the QUANTUM-ESPRESSO distribution (`http://www.quantum-espresso.org`), is fully interfaced to `wannier90` via the `pw2wannier90` post-processing code that is also available as part of QUANTUM-ESPRESSO. The latest version of `pw2wannier90` is included as part of the `wannier90` distribution. Please see the `pwscf` directory for instructions on how to incorporate it into PWSCF.

Ivo: Should we explain somewhere in the tutorial (here?) how to execute `pw2wannier90` in parallel? (and also `pw.x`) Note that in Example 16 Giovanni explained how to execute `postw90` in parallel, and then I did the same for Examples 17–20. For 17–20 at least, running `pw2wannier90` is even more time-consuming than running `postw90`...

GP: I try to add the following, to decide if this is the correct place. Feel free to modify/adapt/correct. I put here only the discussion for the parallel execution of pw2wannier90, and instead the part for wannier+postw90 in the preliminaries.

Note that, as `pw.x`, also the `pw2wannier90.x` executable can be run in parallel (and for large calculations, this is extremely useful to significantly reduce the computation time). This requires that the code is compiled in its parallel version, using the MPI libraries. Refer to the QUANTUM-ESPRESSO package for the documentation on how to compile the code in parallel. Note that, unless you specify `wf_collect=.true.` in your `pw.x` input file, you must run `pw2wannier90` with the same number of processors as `pw.x`.

Moreover we remind here that as discussed in the "Parallel execution" section at page 3, while the `wannier90` executable is serial and cannot be run in parallel, the `postw90.x` can be run in parallel and in this case any number of processors can be used, independently of what has been used for `pw.x` and `pw2wannier90.x`.

5: Diamond – MLWFs for the valence bands

- Outline: *Obtain MLWFs for the valence bands of diamond*
- Directory: `examples/example5/`
- Input Files
  - `diamond.scf` *The* PWSCF *input file for ground state calculation*
  - `diamond.nscf` *The* PWSCF *input file to obtain Bloch states on a uniform grid*
  - `diamond.pw2wan` *The input file for* `pw2wannier90`
  - `diamond.win` *The* `wannier90` *input file*

1. Run PWSCF to obtain the ground state of diamond
   `pw.x < diamond.scf > scf.out`
2. Run PWSCF to obtain the Bloch states on a uniform k-point grid
   `pw.x < diamond.nscf > nscf.out`
3. Run `wannier90` to generate a list of the required overlaps (written into the `diamond.nnkp` file).
   `wannier90.x -pp diamond`
4. Run `pw2wannier90` to compute the overlap between Bloch states and the projections for the starting guess (written in the `diamond.mmn` and `diamond.amn` files).
   `pw2wannier90.x < diamond.pw2wan > pw2wan.out`
5. Run `wannier90` to compute the MLWFs.
   `wannier90.x diamond`

6: Copper – Fermi surface

- Outline: *Obtain MLWFs to describe the states around the Fermi-level in copper*
- Directory: `examples/example6/`
- Input Files
  - `copper.scf` *The* PWSCF *input file for ground state calculation*
  - `copper.nscf` *The* PWSCF *input file to obtain Bloch states on a uniform grid*
  - `copper.pw2wan` *Input file for* `pw2wannier90`
  - `copper.win` *The* `wannier90` *input file*

1. Run PWSCF to obtain the ground state of copper
   `pw.x < copper.scf > scf.out`
2. Run PWSCF to obtain the Bloch states on a uniform k-point grid
   `pw.x < copper.nscf > nscf.out`
3. Run `wannier90` to generate a list of the required overlaps (written into the `copper.nnkp` file).
   `wannier90.x -pp copper`
4. Run `pw2wannier90` to compute the overlap between Bloch states and the projections for the starting guess (written in the `copper.mmn` and `copper.amn` files).
   `pw2wannier90.x < copper.pw2wan > pw2wan.out`
5. Run `wannier90` to compute the MLWFs.
   `wannier90.x copper`

Inspect the output file `copper.wout`.

1. Use Wannier interpolation to obtain the Fermi surface of copper. Rather than re-running the whole calculation we can use the unitary transformations obtained in the first calculation and restart from the plotting routine. Add the following keywords to the `copper.win` file:

   ```
   restart = plot
   fermi_energy = [insert your value here]
   fermi_surface_plot = true
   ```

   and re-run `wannier90`. The value of the Fermi energy can be obtained from the initial first principles calculation. `wannier90` calculates the band energies, through wannier interpolation, on a dense mesh of k-points in the Brillouin zone. The density of this grid is controlled by the keyword `fermi_surface_num_points`. The default value is 50 (i.e., 50^3 points). The Fermi surface file `copper.bxsf` can be viewed using XCrySDen, e.g.,

   ```
   xcrysden -bxsf copper.bxsf
   ```

2. Plot the interpolated bandstructure. A suitable path in k-space is

```
begin kpoint_path
G 0.00 0.00 0.00 X 0.50 0.50 0.00
X 0.50 0.50 0.00 W 0.50 0.75 0.25
W 0.50 0.75 0.25 L 0.00 0.50 0.00
L 0.00 0.50 0.00 G 0.00 0.00 0.00
G 0.00 0.00 0.00 K 0.00 0.50 -0.50
end kpoint_path
```

Further ideas

- Compare the Wannier interpolated bandstructure with the full PWSCF bandstructure. Obtain MLWFs using a denser k-point grid. [Provide script for this? Link to Jonathan's URL?]
- Investigate the effects of the outer and inner energy windows on the interpolated bands.
- Instead of extracting a subspace of seven states, we could extract a nine dimensional space (i.e., with s, p and d character). Examine this case and compare the interpolated bandstructures.

7: Silane (SiH_4) – Molecular MLWFs using Γ-point sampling

- Outline: *Obtain MLWFs for the occupied states of molecular silane.* Γ*-point sampling*
- Directory: `examples/example7/`
- Input Files
  - `silane.scf` *The* PWSCF *input file for ground state calculation*
  - `silane.nscf` *The* PWSCF *input file to obtain Bloch states on a uniform grid*
  - `silane.pw2wan` *Input file for* `pw2wannier90`
  - `silane.win` *The* `wannier90` *input file*

1. Run PWSCF to obtain the ground state of silane
   `pw.x < silane.scf > scf.out`
2. Run PWSCF to obtain the Bloch states on a uniform k-point grid
   `pw.x < silane.nscf > nscf.out`
3. Run `wannier90` to generate a list of the required overlaps (written into the `silane.nnkp` file).
   `wannier90.x -pp silane`
4. Run `pw2wannier90` to compute the overlap between Bloch states and the projections for the starting guess (written in the `silane.mmn` and `silane.amn` files).
   `pw2wannier90.x < silane.pw2wan > pw2wan.out`
5. Run `wannier90` to compute the MLWFs.
   `wannier90.x silane`

8: Iron – Spin-polarized WFs, total and projected DOS

- Outline: ~~*Obtain MLWFs to describe the states around the Fermi-level in iron.*~~ *Generate Wannier functions for ferromagnetic iron, and use them to calculate the density of states (DOS) by Wannier interpolation.*
- Directory: `examples/example8/`
- Input Files
  - `iron.scf` *The* PWSCF *input file for ground state calculation*
  - `iron.nscf` *The* PWSCF *input file to obtain Bloch states on a uniform grid*
  - `iron_up.pw2wan` *Input file for* `pw2wannier90` *- spin up*
  - `iron_up.win` *The* `wannier90` *and* `postw90` *input file - spin up*
  - `iron_dn.pw2wan` *Input file for* `pw2wannier90` *- spin down*
  - `iron_dn.win` *The* `wannier90` *and* `postw90` *input file - spin down*
- Note that we obtain the MLWFs for spin up and spin down in separate calculations.

1. Run PWSCF to obtain the ground state of iron
   ```
   pw.x < iron.scf > scf.out
   ```
2. Run PWSCF to obtain the Bloch states on a uniform k-point grid
   ```
   pw.x < iron.nscf > nscf.out
   ```
3. Run `wannier90` to generate a list of the required overlaps (written into the `iron_up.nnkp` and `iron_dn.nnkp` files).
   ```
   wannier90.x -pp iron_up
   wannier90.x -pp iron_dn
   ```
4. Run `pw2wannier90` to compute the overlap between Bloch states and the projections for the starting guess (written in the `iron_up.mmn`, `iron_dn.mmn`, `iron_up.amn` and `iron_dn.amn` files).
   ```
   pw2wannier90.x < iron_up.pw2wan > pw2wan.out
   pw2wannier90.x < iron_dn.pw2wan > pw2wan.out
   ```
5. Run `wannier90` to compute the MLWFs.
   ```
   wannier90.x iron_up
   wannier90.x iron_dn
   ```
6. Compute the DOS on a $25 \times 25 \times 25$ k-point grid. First add the following lines to the two `.win` input files,

   ```
   dos = T
   dos_task = dos_plot
   dos_kmesh = 25
   ```

   and then run `postw90`,
   ```
   postw90.x iron_up
   postw90.x iron_dn
   ```
7. Plot the DOS with `gnuplot`,

```
myshell> gnuplot
gnuplot> plot 'iron_up_dos.dat' u (-$2):($1-12.6256) w l,'iron_dn_dos.dat'
u 2:($1-12.6256) w l
```

We have set the zero of the energy scale at the Fermi level by subtracting the value of the Fermi energy taken from `scf.out`. Check if the DOS is sufficiently well-converged by increasing the parameter `dos_kmesh`.

In the calculations above we chose s, p, and d-type trial orbitals (see `.win` files),

```
Fe:s;p;d
```

Let us analyze the evolution of the WFs during the gauge-selection step. Open one of the `.wout` files and search for "`Initial state`"; those are the *projected* WFs. As expected they are atom-centered, with spreads organized in three groups, 1+3+5: one s, three p, and five d. Now look at the final state at the end of the file. The Wannier spreads have re-organized in two groups, 6+3; moreover, the six more diffuse WFs are now off-centered. The initial atomic-like orbitals hybridized with one another, becoming more localized in the process.

To see when the hybridization took place, let us plot the evolution of the total spread Ω,

```
myshell> grep SPRD iron_up.wout >sprd_up
myshell> gnuplot
gnuplot> plot 'sprd_up' u 6 w l
```

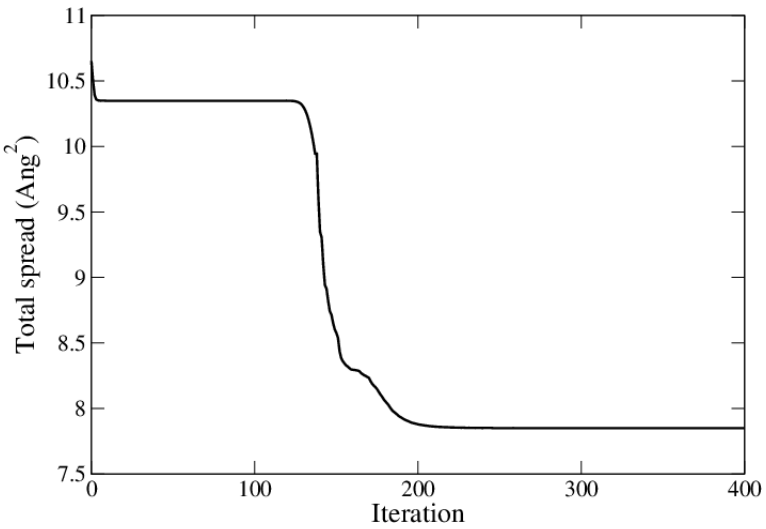

Figure 4: Evolution of the Wannier spread Ω of bcc Fe during the iterative minimization of the gauge-dependent part $\widetilde{\Omega}$, starting from s, p, and d-type trial orbitals.

The first plateau corresponds to atom-centered WFs of separate s, p, and d character, same as the trial orbitals, and the sharp drop at around iteration 120 signals the onset of the hybridization. For more details, see Sec. IV.B of Ref. [9]. With hindsight, we can redo steps 4 and 5 more efficiently using trial orbitals with the same character as the final MLWFs,

```
Fe:sp3d2;dxy;dxz,dyz
```

Physical quantities should come out essentially the same computed with any reasonable set of localized WFs, not necessarily maximally-localized. Let us re-run steps 4 and 5 again starting from s, p, and d-type trial orbitals, but stopping the minimization of $\widetilde{\Omega}$ before the onset of hybridization,

```
num_iter = 50
```

[Jonathan: This may be a good place to use projected only WFs in the tutorial!]

Now recalculate the DOS, and check that it is almost identical to the one computed earlier using the hybridized set of MLWFs.

Further ideas

[In the following it is assumed that *non-hybridized* WFs of separate s, p, and d character have been obtained as described above.]

- In order to obtain the DOS projected onto the p-type WFs, add to the `.win` files the lines

  ```
  dos_project = 2,3,4
  ```

  (compare with Example 4) and re-run `postw90`. Plot the p-projected DOS for both up- and down-spin bands. Repeat for the s and and d projections.

- Re-run `wannier90` after adding to the `.win` files the instruction

  ```
  write_hr_diag = T
  ```

  This tells `wannier90` to write in the output file the diagonal ("on-site") Wannier matrix elements of the Hamiltonian. The difference between corresponding values in `iron_up.wout` and in `iron_dn.wout` gives the exchange splittings for the individual s, p, and d-type WFs. Compare their magnitudes with the splittings displayed by the total and orbital-projected DOS.

9: Cubic $BaTiO_3$

- Outline: *Obtain MLWFs for a perovskite*
- Directory: `examples/example9/`
- Input Files
  - `batio3.scf` *The* PWSCF *input file for ground state calculation*
  - `batio3.nscf` *The* PWSCF *input file to obtain Bloch states on a uniform grid*
  - `batio3.pw2wan` *Input file for* `pw2wannier90`
  - `batio3.win` *The* `wannier90` *input file*

To start with, we are going to obtain MLWFs for the oxygen 2p states. From the bandstructure [6], these form an isolated group of bands. We use the `wannier90` keyword `exclude_bands` to remove all but the 2p bands from the calculation of the overlap and projection matrices (we don't have to do this, but it saves time).

1. Run PWSCF to obtain the ground state of $BaTiO_3$
   `pw.x < BaTiO3.scf > scf.out`
2. Run PWSCF to obtain the Bloch states on a uniform k-point grid
   `pw.x < BaTiO3.nscf > nscf.out`
3. Run `wannier90` to generate a list of the required overlaps (written into the `BaTiO3.nnkp` file).
   `wannier90.x -pp BaTiO3`
4. Run `pw2wannier90` to compute the overlap between Bloch states and the projections for the starting guess (written in the `BaTiO3.mmn` and `BaTiO3.amn` files).
   `pw2wannier90.x < BaTiO3.pw2wan > pw2wan.out`
5. Run `wannier90` to compute the MLWFs.
   `wannier90.x BaTiO3`

Inspect the output file `BaTiO3.wout`.

Plot the second MLWF, as described in Section 1, by adding the following keywords to the input file `BaTiO3.win`

```
wannier_plot = true
restart = plot
wannier_plot_list = 2
wannier_plot_supercell = 3
```

and re-running `wannier90`. Visualise it using XCrySDen,

```
xcrysden -xsf BaTiO3_00002.xsf
```

We can now simulate the ferroelectric phase by displacing the Ti atom. Change its position to

```
Ti 0.505 0.5 0.5
```

and regenerate the MLWFs (i.e., compute the ground-state charge density and Bloch states using PWSCF, etc.) and look at the change in the second MLWF.

Further ideas

- Look at MLWFs for other groups of bands. What happens if you form MLWFs for the whole valence manifold?
- Following Ref. [6], compute the Born effective charges from the change in Wannier centres under an atomic displacement.

10: Graphite

- Outline: *Obtain MLWFs for graphite (AB, Bernal)*
- Directory: `examples/example10/`
- Input Files
  - `graphite.scf` *The* PWSCF *input file for ground state calculation*
  - `graphite.nscf` *The* PWSCF *input file to obtain Bloch states on a uniform grid*
  - `graphite.pw2wan` *Input file for* `pw2wannier90`
  - `graphite.win` *The* `wannier90` *input file*

1. Run PWSCF to obtain the ground state of graphite
   `pw.x < graphite.scf > scf.out`
2. Run PWSCF to obtain the Bloch states on a uniform k-point grid
   `pw.x < graphite.nscf > nscf.out`
3. Run `wannier90` to generate a list of the required overlaps (written into the `graphite.nnkp` file).
   `wannier90.x -pp graphite`
4. Run `pw2wannier90` to compute the overlap between Bloch states and the projections for the starting guess (written in the `graphite.mmn` and `graphite.amn` files).
   `pw2wannier90.x < graphite.pw2wan > pw2wan.out`
5. Run `wannier90` to compute the MLWFs.
   `wannier90.x graphite`

11: Silicon – Valence and conduction states, scissors correction

Valence States

- Outline: *Obtain MLWFs for the valence bands of silicon.*
- Directory: `examples/example11/`
- Input Files
  - `silicon.scf` *The* PWSCF *input file for ground state calculation*
  - `silicon.nscf` *The* PWSCF *input file to obtain Bloch states on a uniform grid*
  - `silicon.pw2wan` *Input file for* `pw2wannier90`
  - `silicon.win` *The* `wannier90` *input file*

1. Run PWSCF to obtain the ground state of silicon
   `pw.x < silicon.scf > scf.out`
2. Run PWSCF to obtain the Bloch states on a uniform k-point grid. Note that we request the lower 4 (valence) bands
   `pw.x < silicon.nscf > nscf.out`
3. Run `wannier90` to generate a list of the required overlaps (written into the `silicon.nnkp` file).
   `wannier90.x -pp silicon`
4. Run `pw2wannier90` to compute the overlap between Bloch states and the projections for the starting guess (written in the `silicon.mmn` and `silicon.amn` files).
   `pw2wannier90.x < silicon.pw2wan > pw2wan.out`
5. Run `wannier90` to compute the MLWFs.
   `wannier90.x silicon`

Inspect the output file `silicon.wout`. The total spread converges to its minimum value after just a few iterations. Note that the geometric centre of each MLWF lies at the centre of the Si-Si bond. Note also that the memory requirement for the minimisation of the spread is very low as the MLWFs are defined by just the 4×4 unitary matrices $\mathbf{U}^{(\mathbf{k})}$.

Plot the MLWFs by adding the following keywords to the input file `silicon.win`

```
wannier_plot = true
```

and re-running `wannier90`. Visualise them using XCrySDen, e.g.,

```
xcrysden -xsf silicon_00001.xsf
```

Valence + Conduction States

- Outline: *Obtain MLWFs for the valence and low-lying conduction-band states of Si. Plot the interpolated bandstructure. Apply a scissors correction to the conduction bands.*

- Input Files
  - `silicon.scf` *The* PWSCF *input file for ground state calculation*
  - `silicon.nscf` *The* PWSCF *input file to obtain Bloch states on a uniform grid*
  - `silicon.pw2wan` *Input file for* `pw2wannier90`
  - `silicon.win` *The* `wannier90` *input file*

The valence and lower conduction states can be represented by MLWFs with sp^3-like symmetry. The lower conduction states are not separated by an energy gap from the higher states. In order to form localised WF we use the disentanglement procedure introduced in Ref. [3]. The position of the inner and outer energy windows are shown in Fig. 2.

1. Modify the input file and run PWSCF and `wannier90`.
   Inspect the output file `silicon.wout`. The minimisation of the spread occurs in a two-step procedure. First, we minimise $\Omega_{\rm I}$ – this is the extraction of the optimal subspace in the disentanglement procedure. Then, we minimise $\Omega_{\rm O} + \Omega_{\rm OD}$.

2. Plot the bandstructure by adding the following commands to the input file `silicon.win`

   ```
   restart = plot
   bands_plot = true
   ```

   and re-running `wannier90`. The files `silicon_band.dat` and `silicon_band.gnu` are created. To plot the bandstructure using gnuplot

   ```
   myshell> gnuplot
   gnuplot> load 'silicon_band.gnu'
   ```

   The k-point path for the bandstructure interpolation is set in the `kpoint_path` block. Try plotting along different paths.

3. Shift the conduction bands upwards in energy by 0.65 eV. This *ad-hoc* "scissors correction" is applied to the Hamiltonian in the Wannier basis using the instructions

   ```
   num_valence_bands = 4
   scissors_shift = 0.65
   ```

   Plot together the original and scissors-corrected bands.

Further ideas

- Compare the Wannier interpolated bandstructure with the full PWSCF bandstructure. Compute MLWFs using a finer k-point grid (e.g., 6×6×6 or 8×8×8) and note how the accuracy of the interpolation increases [7].
- Compute four MLWFs spanning the low-lying conduction states (see Ref. [3]).

12: Benzene

Valence States

- Outline: *Obtain MLWFs for the valence states of benzene*
- Directory: `examples/example12/`
- Input Files
  - `benzene.scf` *The* PWSCF *input file for ground state calculation*
  - `benzene.pw2wan` *Input file for* `pw2wannier90`
  - `benzene.win` *The* `wannier90` *input file*

1. Run PWSCF to obtain the ground state of benzene
   `pw.x < benzene.scf > scf.out`
2. Run `wannier90` to generate a list of the required overlaps (written into the `benzene.nnkp` file).
   `wannier90.x -pp benzene`
3. Run `pw2wannier90` to compute the overlap between Bloch states and the projections for the starting guess (written in the `benzene.mmn` and `benzene.amn` files).
   `pw2wannier90.x < benzene.pw2wan > pw2wan.out`
4. Run `wannier90` to compute the MLWFs.
   `wannier90.x benzene`

Inspect the output file `benzene.wout`. The total spread converges to its minimum value after just a few iterations.

Plot the MLWFs by adding the following keywords to the input file `benzene.win`

```
restart = plot
wannier_plot = true
wannier_plot_format = cube
wannier_plot_list = 2-4
```

and re-running `wannier90`. Visualise them using, e.g., XCrySDen.

Valence + Conduction States

- Outline: *Obtain MLWFs for the valence and low-lying conduction states of benzene.*
- Input Files
  - `benzene.scf` *The* PWSCF *input file for ground state calculation*
  - `benzene.nscf` *The* PWSCF *input file to obtain Bloch states for the conduction states*
  - `benzene.pw2wan` *Input file for* `pw2wannier90`
  - `benzene.win` *The* `wannier90` *input file*

In order to form localised WF we use the disentanglement procedure. The position of the inner energy window is set to lie in the energy gap; the outer energy window is set to 4.0 eV. Modify the input file appropriately.

1. Run PWSCF and `wannier90`.
   Inspect the output file `benzene.wout`. The minimisation of the spread occurs in a two-step procedure. First, we minimise Ω_{I}. Then, we minimise $\Omega_{\mathrm{O}} + \Omega_{\mathrm{OD}}$.

2. Plot the MLWFs by adding the following commands to the input file `benzene.win`

   ```
   restart = plot
   wannier_plot = true
   wannier_plot_format = cube
   wannier_plot_list = 1,7,13
   ```

   and re-running `wannier90`. Visualise them using, e.g., XCrySDen.

13: (5,5) Carbon Nanotube – Transport properties

- Outline: *Obtain the bandstructure, quantum conductance and density of states of a metallic (5,5) carbon nanotube*
- Directory: `examples/example13/`
- Input Files
  - `cnt55.scf` *The* PWSCF *input file for ground state calculation*
  - `cnt55.nscf` *The* PWSCF *input file to obtain Bloch states for the conduction states*
  - `cnt55.pw2wan` *Input file for* `pw2wannier90`
  - `cnt55.win` *The* `wannier90` *input file*

In order to form localised WF that describe both the occupied and unoccupied π and π^* manifolds, we use the disentanglement procedure to extract a smooth manifold of states that has dimension equal to 2.5 times the number of carbon atoms per unit cell [8]. The positions of the energy windows are shown in Fig. 5.

The part of the `wannier90` input file that controls the transport part of the calculation looks like:

```
transport = true
transport_mode = bulk
one_dim_axis = z
dist_cutoff = 5.5
fermi_energy = -1.06
tran_win_min = -6.5
tran_win_max = 6.5
tran_energy_step = 0.01
dist_cutoff_mode = one_dim
translation_centre_frac = 0.0 0.0 0.0
```

Descriptions of these and other keywords related to the calculation of transport properties can be found in the User Guide.

1. Run PWSCF and `wannier90`.
   Inspect the output file `cnt55.wout`. The minimisation of the spread occurs in a two-step procedure. First, we minimise $\Omega_{\rm I}$. Then, we minimise $\Omega_{\rm O} + \Omega_{\rm OD}$.
2. Note that the initial p_z projections on the carbon atoms are oriented in the radial direction with respect to the nanotube axis.
3. The interpolated bandstructure is written to `cnt55_band.agr` (since `bands_plot_format = xmgr` in the input file).
4. The quantum conductance and density of states are written to the files `cnt55_qc.dat` and `cnt55_dos.dat`, respectively. Note that this part of the calculation may take some time. You can follow its progress by monitoring the output to these files. Use a package such as `gnuplot` or `xmgrace` in order to visualise the data. You should get something that looks like Fig. 6.

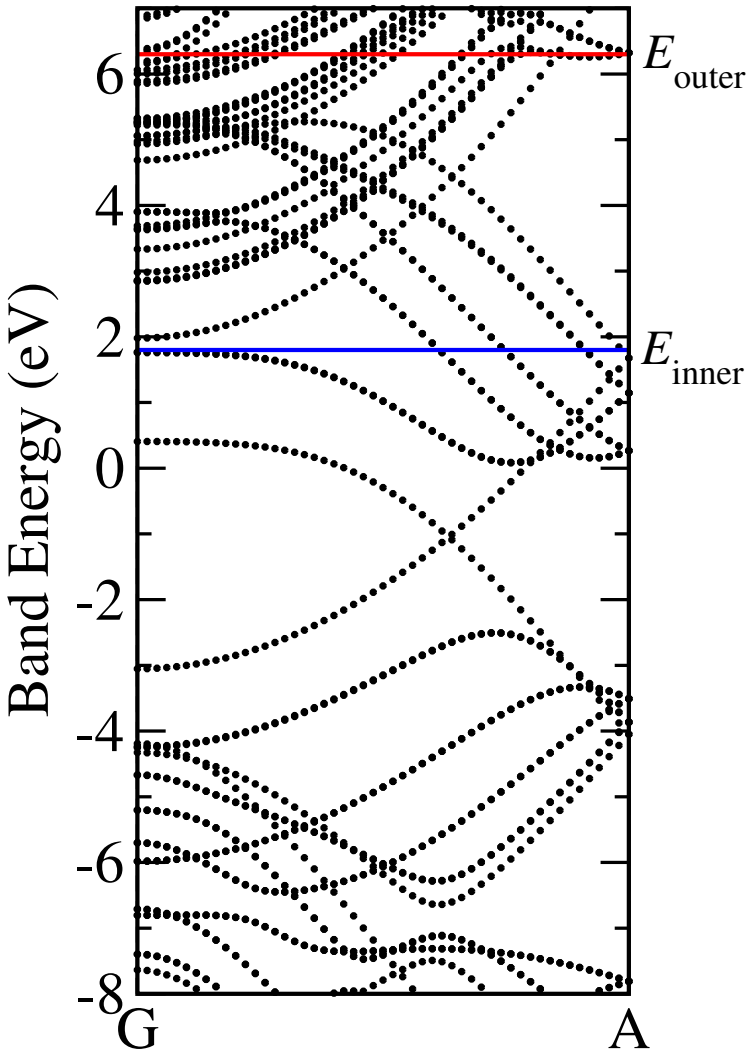

Figure 5: Bandstructure of (5,5) carbon nanotube showing the position of the outer and inner energy windows.

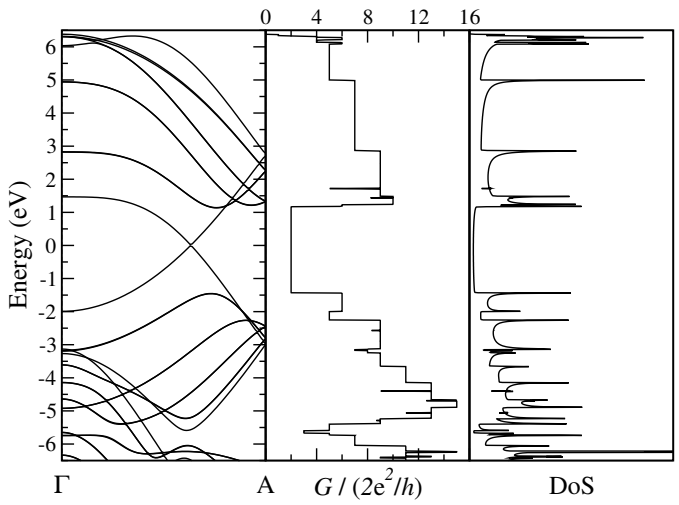

Figure 6: Wannier interpolated bandstructure, quantum conductance and density of states of (5,5) carbon nanotube. Note that the Fermi level has been shifted by 1.06eV with respect to Fig. 5.

14: Linear Sodium Chain – Transport properties

- Outline: *Compare the quantum conductance of a periodic linear chain of Sodium atoms with that of a defected chain*
- Directories: `examples/example14/periodic`
  `examples/example14/defected`
- Input Files
  - `Na_chain.scf` *The* PWSCF *input file for ground state calculation*
  - `Na_chain.nscf` *The* PWSCF *input file to obtain Bloch states for the conduction states*
  - `Na_chain.pw2wan` *Input file for* `pw2wannier90`
  - `Na_chain.win` *The* `wannier90` *input file*

The periodic system contains two unit cells evenly distributed along the supercell. Transport calculations are performed using `transport_mode = bulk` and so the resulting quantum conductance represents that of an infinite periodic chain.

The part of the `wannier90` input file that controls the transport part of the calculation looks like:

```
transport = true
transport_mode = bulk
tran_read_ht = false
one_dim_axis = x
fermi_energy = -2.7401
tran_win_min = -5.0
tran_win_max = 5.0
tran_energy_step = 0.01
translation_centre_frac = 0.5 0.5 0.5
tran_num_bb = 2
```

The defected system uses a 13 atom supercell with the central atom position altered to break symmetry. Setting `transport_mode = lcr` with tell `wannier90` to treat the system as an infinte sytsem with the defect at its centre. The supercell is chosen so that is conforms to the 2c2 geometry (see User Guide for details). Each principal layer is 2 atoms long so that the conductor region contains the defected atom plus a single atom on either side.

The transport section of the input file contians these key differences:

```
transport_mode = lcr
tran_num_ll = 2
tran_num_cell_ll = 2
```

Descriptions of these and other keywords related to the calculation of transport properties can be found in the User Guide.

1. Run PWSCF and `wannier90` for the periodic system.

2. Run PWSCF and `wannier90` for the defected system.

3. The quantum conductance is written to the files `periodic/Na_chain_qc.dat` and `defected/Na_chain_dos.dat`, respectively. Compare the quantum conductance of the periodic (bulk) calculation with the defected (LCR) calculation. Your plot should look like Fig. 7.

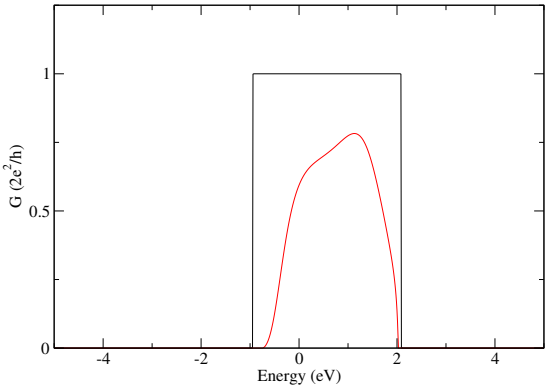

Figure 7: Quantum conductance of periodic Sodium chain (black) compared to that of the defected Sodium chain (red).

15: (5,0) Carbon Nanotube – Transport properties

Note that these systems require reasonably large-scale electronic structure calculations.

Bulk Transport properties

- Outline: *Obtain the quantum conductance of a pristine single-walled carbon nanotube*
- Directory: `examples/example14/periodic`
- Input Files
  - `cnt.scf` *The* PWSCF *input file for ground state calculation*
  - `cnt.nscf` *The* PWSCF *input file to obtain Bloch states for the conduction states*
  - `cnt.pw2wan` *Input file for* `pw2wannier90`
  - `cnt.win` *The* `wannier90` *input file*

First we consider a single unit cell, with 10 k-points. With `transport_mode = bulk` we compute the transport properties of a pristine, infinite, periodic (5,0) carbon nanotube. Later, we will compare the quantum conductance of this system with a defected nanotube.

1. Run PWSCF and `wannier90`.

2. The quantum conductance and density of states are written to the files `cnt_qc.dat` and `cnt_dos.dat`, respectively.

LCR transport properties – Defected nanotube

- Outline: *Use the automated LCR routine to investigate the effect of a single silicon atom in a infinite (5,0) carbon nanotube.*
- Directory: `examples/example15/defected`
- Input Files
  - `cnt+si.scf` *The* PWSCF *input file for ground state calculation*
  - `cnt+si.nscf` *The* PWSCF *input file to obtain Bloch states for the conduction states*
  - `cnt+si.pw2wan` *Input file for* `pw2wannier90`
  - `cnt+si.win` *The* `wannier90` *input file*

In this calculation an 11-atom supercell is used with a single silicon substitutional defect in the central unit cell. The supercell is chosen so that is conforms to the 2c2 geometry (see User Guide for details) with principal layers set to be two unit cells long.

1. Run PWSCF and `wannier90`. Again these are large calculations, progress can be monitored by viewing respective output files.

2. The quantum conductance is written to `cnt+si_qc.dat`. Compare the quantum conductance with the periodic (bulk) calculation. Your plot should look like Fig. 8.

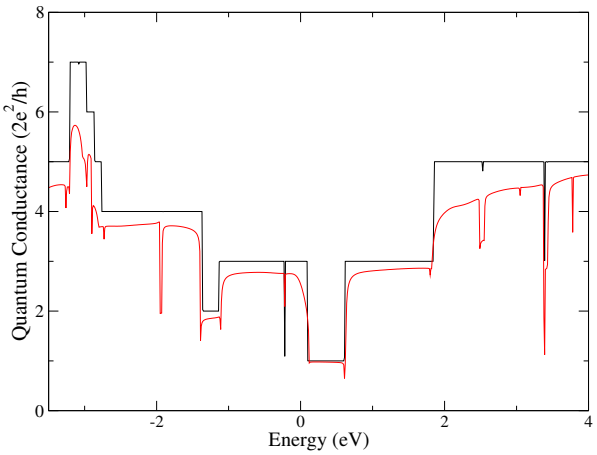

Figure 8: Quantum conductance of infinite pristine nanotube (black) compared to that of the infinite nanotube with the substitutional silicon defect (red).

Further ideas

- Set `hr_plot = true` in the bulk case. Consider the magnitude of Hamiltonian elements between Wannier functions in increasingly distant unit cells. Are two unit cell principal layers really large enough, or are significant errors introduced?
- Does one unit cell either side of the defected unit cell shield the disorder so that the leads are ideal? Does the quantum conductance change if these 'buffer' regions are increased?

16: Silicon – Boltzmann transport

- Outline: *Obtain MLWFs for the valence and low-lying conduction states of Si. Calculate the electrical conductivity, the Seebeck coefficient and the thermal conductivity in the constant relaxation time approximation using the* `BoltzWann` *module.*
- Directory: `examples/example16/`
- Input Files
  - `Si.scf` *The* PWSCF *input file for ground state calculation*
  - `Si.nscf` *The* PWSCF *input file to obtain Bloch states on a uniform grid*
  - `Si.pw2wan` *Input file for* `pw2wannier90`
  - `Si.win` *The* `wannier90` *and* `postw90` *input file*

Note the first five steps in the following are the same of Example 11. If you have already run Example 11 (in particular, the section "Valence + Conduction States") you can start from those files and continue from point 6, after having added the `BoltzWann` flags to the input file.

Note that in the following, steps 1, 2, 4 and 6 can be possibly run in parallel on multiple processors (using `mpirun`), while steps 3 and 5 must be run in serial.

1. Run PWSCF to obtain the ground state of silicon
   `pw.x < Si.scf > scf.out`
2. Run PWSCF to obtain the Bloch states on a uniform k-point grid. Details on the disentanglement procedure are discussed in Example 11.
   `pw.x < Si.nscf > nscf.out`
3. Run `wannier90` to generate a list of the required overlaps (written into the `Si.nnkp` file).
   `wannier90.x -pp Si`
4. Run `pw2wannier90` to compute the overlap between Bloch states and the projections for the starting guess (written in the `Si.mmn` and `Si.amn` files).
   `pw2wannier90.x < Si.pw2wan > pw2wan.out`
5. Run `wannier90` to compute the MLWFs.
   `wannier90.x Si`

   Inspect the output file `Si.wout` and check if the convergence was reached both in the disentanglement and in the wannierisation steps (as discussed in further detail in Example 11). You may also want to plot the Wannier functions and the interpolated band structure.
6. Run `postw90` to calculate the transport coefficients.
   `postw90.x Si` (serial execution)
   `mpirun -np 8 postw90.x Si` (example of parallel execution with 8 nodes)

Inspect the output file `Si.wpout`. It summarizes the main details of the calculation (more details can be obtained by setting a larger value of the `iprint` flag). Check if no warnings are issued. Note that if no special flags are passed to `BoltzWann`, it assumes that the ab-initio calculation did not include magnetization effects, and thus it sets to 2 the number of electrons per state.

Note also that the value of the relaxation time $\tau = 10$ fs in the example is set only as a representative value; note also that only the electrical and thermal conductivity depend on τ, while the Seebeck coefficient is independent of τ.

Using your favourite plotting program, plot the `Si_boltzdos.dat` file to inspect the DOS.

Using your favourite plotting program, plot columns 1 and 3 of the `Si_seebeck.dat` file to inspect the S_{xx} component of the Seebeck coefficient as a function of the chemical potential μ, at $T = 300$ K.

Further ideas

- Change the interpolation to a $60 \times 60 \times 60$ mesh and run again `postw90` to check if the results for the transport properties are converged.
- Change the `Si.win` input file so that it calculates the transport coefficients for temperatures from 300 to 700 K, with steps of 200 K. Rerun `postw90` and verify that the increase in execution time is neglibile (in fact, most of the time is spent to interpolate the band structure on the k mesh).

  Plot the Seebeck coefficient for the three temperatures $T = 300$ K, $T = 500$ K and $T = 700$ K. To do this, you have to filter the `Si_seebeck.dat` to select only those lines where the second column is equal to the required temperature. A possible script to select the S_{xx} component of the Seebeck coefficient for $T = 500$ K using the `awk/gawk` command line program is the following:

  ```
  awk '{if ($2 == 500) {print $1, $3;}}' < Si_seebeck.dat \
      > Si_seebeck_xx_500K.dat
  ```

  Then, you can plot columns 1 and 2 of the output file `Si_seebeck_xx_500K.dat`.
- Try to calculate the Seebeck coefficient as a function of the temperature, for a $n-$doped sample with, e.g., $n = 10^{18}$ cm^{-3}. Note that to this aim, you need to calculate consistently the value $\mu(T)$ of the chemical potential as a function of the temperature, so as to reproduce the given value of n. Then, you have to write a small program/script to interpolate the output of `BoltzWann`, that you should have run on a suitable grid of (μ, T) points.

17: Iron – Spin-orbit-coupled bands and Fermi surface contours

Note: You should go through Example 8 (iron without spin-orbit) before doing this Example.

- Outline: *Plot the spin-orbit-coupled bands of ferromagnetic iron. Plot the Fermi-surface contours on a plane in the Brillouin zone.*
- Directory: `examples/example17/`
- Input files
  - `Fe.scf` *The* PWSCF *input file for ground state calculation*
  - `Fe.nscf` *The* PWSCF *input file to obtain Bloch states on a uniform grid*
  - `Fe.pw2wan` *The input file for* `pw2wannier90`
  - `Fe.win` *The* `wannier90` *and* `postw90` *input file*

Note that `num_wann =18` in `Fe.win`, but only nine trial orbitals are provided. The line

```
spinors = T
```

tells `wannier90` to use in step 3 below the specified trial orbitals on both the up- and down-spin channels, effectively doubling their number.

[Jonathan, is the above explanation accurate?]

1. Run PWSCF to obtain the ferromagnetic ground state of iron[2]
   `pw.x < Fe.scf > scf.out`
2. Run PWSCF to obtain the Bloch states on a uniform k-point grid
   `pw.x < Fe.nscf > nscf.out`
3. Run `wannier90` to generate a list of the required overlaps (written into the `Fe.nnkp` file)
   `wannier90.x -pp Fe`
4. Run `pw2wannier90` to compute:
   - The overlaps $\langle u_{n\mathbf{k}} | u_{m\mathbf{k}+\mathbf{b}} \rangle$ between *spinor* Bloch states (written in the `Fe.mmn` file)
   - The projections for the starting guess (written in the `Fe.amn` file)
   - The spin matrix elements $\langle \psi_{n\mathbf{k}} | \sigma_i | \psi_{m\mathbf{k}} \rangle$ (written in the `Fe.spn` file)

   `pw2wannier90.x < Fe.pw2wan > pw2wan.out`
5. Run `wannier90` to compute the MLWFs.
   `wannier90.x Fe`
6. Run `postw90` to compute the energy eigenvalues and spin expectation values by Wannier interpolation.
   `postw90.x Fe` (serial execution)
   `mpirun -np 8 postw90.x Fe` (example of parallel execution with 8 nodes)

[2]Please note the following counterintiutive feature in `pwscf`: in order to obtain a ground state with magnetization along the *positive* z-axis, one should use a *negative* value for the variable `starting_magnetization`.

In this example we use the module `kpath` to plot the energy bands coloured by the expectation value of the spin along [001]:

```
kpath = T
kpath_task = bands
kpath_bands_colour = spin
kpath_num_points=500
```

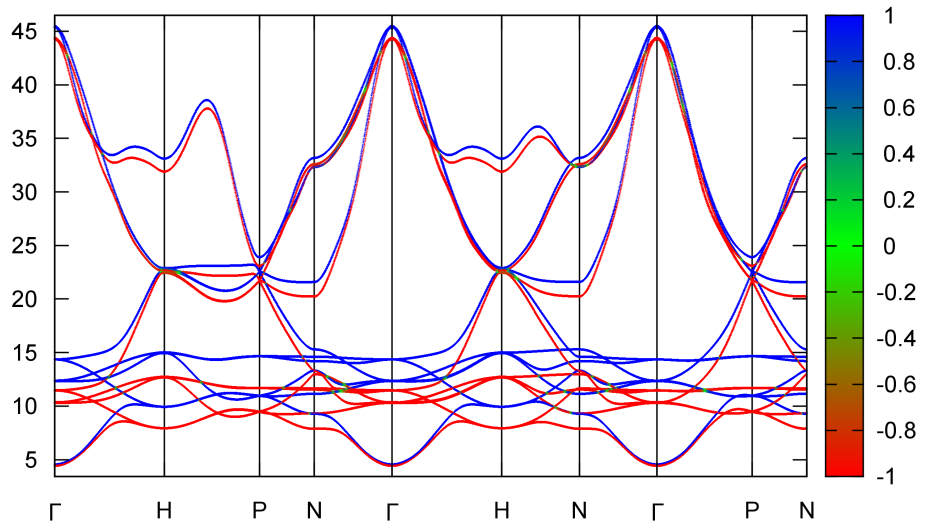

Figure 9: Relativistic bandstructure of bcc iron with the magnetization along the positive [001] direction. Majority bands (with $\langle S_z \rangle = -\hbar/2$) are in red, minority bands ($\langle S_z \rangle = +\hbar/2$) in blue. Because of spin-orbit coupling S_z is not a good quantum number, and intermediate colors occur in the vicinity of (avoided) crossings between up- and down-spin bands. The Fermi level is at 12.6 eV.

A suitable path is

```
begin kpoint_path
G 0.0000 0.0000 0.0000 H 0.500 -0.5000 -0.5000
H 0.500 -0.5000 -0.5000 P 0.7500 0.2500 -0.2500
P 0.7500 0.2500 -0.2500 N 0.5000 0.0000 -0.5000
N 0.5000 0.0000 -0.5000 G 0.0000 0.0000 0.000
G 0.0000 0.0000 0.000 H 0.5 0.5 0.5
H 0.5 0.5 0.5 N 0.5 0.0 0.0
N 0.5 0.0 0.0 G 0.0 0.0 0.0
G 0.0 0.0 0.0 P 0.75 0.25 -0.25
P 0.75 0.25 -0.25 N 0.5 0.0 0.0
end kpoint_path
```

To plot the spin-colored bandstructure using gnuplot (version 4.2 or higher), issue

```
myshell> gnuplot
gnuplot> load 'Fe-band.gnu'
```

We can also inspect the constant-energy surfaces across an entire plane, using the module `kslice`. To obtain the energy eigenvalues on a 200×200 grid covering the (010) plane in the BZ, uncomment the following instructions in `Fe.win`,

```
kslice = T
kslice_task = energy_cntr
kslice_cntr_energy = 12.6285
kslice_kmesh = 200 200
kslice_corner = 0.0 0.0 0.0
kslice_b1 = 0.5 -0.5 -0.5
kslice_b2 = 0.5 0.5 0.5
```

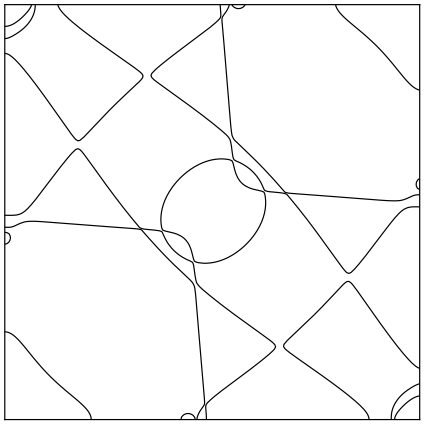

Figure 10: Fermi-surface contours on the (010) plane of bcc Fe.

where we chose the Fermi energy 12.6285 eV (taken from `scf.out`) as the energy level for the isocontours. The Fermi-surface contours (lines of intersection between the Fermi surface and the slice), can be visualized with the script `Fe-energy_cntr.gnu` generated at runtime,

```
myshell> gnuplot
gnuplot> load 'Fe-energy_cntr.gnu'
```

(Do not be concerned by the warning messages, they are caused by the fact that not all bands cross the Fermi energy.) Alternative, the python script `Fe-energy_cntr.py` can be used,

```
myshell> python Fe-energy_cntr.py
```

Further ideas

- Generate the Fermi-surface contours on the (001) plane perpendicular to the magnetization. Note the higher symmetry compared to Fig. 10.

- Redraw the Fermi-surface contours on the (010) plane starting from an *ab initio* calculation without spin-orbit coupling (as in Example 8). Note how the connectivity of the Fermi contours changes.

18: Iron – Berry curvature and anomalous Hall conductivity

- Outline: *Calculate the k-space Berry curvature and anomalous Hall conductivity by Wannier interpolation, as described in Ref. [9].*
- Directory: `examples/example18/`
- Input files
  - `Fe.scf` *The* PWSCF *input file for ground state calculation*
  - `Fe.nscf` *The* PWSCF *input file to obtain Bloch states on a uniform grid*
  - `Fe.pw2wan` *The input file for* `pw2wannier90`
  - `Fe.win` *The* `wannier90` *and* `postw90` *input file*

The sequence of steps below is the same of Example 17. If you have already run that example, you can reuse the output files from steps 1–5, and only step 6 must be carried out again using the new input file `Fe.win`.

1. Run PWSCF to obtain the ground state of iron
   `pw.x < Fe.scf > scf.out`
2. Run PWSCF to obtain the Bloch states on a uniform k-point grid
   `pw.x < Fe.nscf > nscf.out`
3. Run `wannier90` to generate a list of the required overlaps (written into the `Fe.nnkp` file).
   `wannier90.x -pp Fe`
4. Run `pw2wannier90` to compute the overlaps between Bloch states and the projections for the starting guess (written in the `Si.mmn` and `Si.amn` files).
   `pw2wannier90.x < Fe.pw2wan > pw2wan.out`
5. Run `wannier90` to compute the MLWFs.
   `wannier90.x Fe`
6. Run `postw90` to compute the Berry curvature.
   `postw90.x Fe` (serial execution)
   `mpirun -np 8 postw90.x Fe` (example of parallel execution with 8 nodes)

The Berry curvature of a Bloch state $\psi_{n\mathbf{k}} = e^{i\mathbf{k}\cdot\mathbf{r}}u_{n\mathbf{k}}$ is a vector property defined as

$$\mathbf{\Omega}_n(\mathbf{k}) = -\mathrm{Im}\langle\boldsymbol{\nabla}_{\mathbf{k}}u_{n\mathbf{k}}| \times |\boldsymbol{\nabla}_{\mathbf{k}}u_{n\mathbf{k}}\rangle.$$

We are interested in the sum over occupied states,

$$\mathbf{\Omega}(\mathbf{k}) = \sum_n^{\mathrm{occ}} \mathbf{\Omega}_n(\mathbf{k}).$$

The following instructions in `Fe.win` are used to calculate $-\mathbf{\Omega}(\mathbf{k})$ along lines in k-space,

```
fermi_energy = 12.6285
kpath = T
kpath_task = curv
kpath_num_points=1000
```

The path specification in `Fe.win` is the same as in Example 17, and the value of the Fermi energy, 12.6285 eV, was taken from `scf.out`.

Carry out the calculation by executing `postw90` (step 6 above). To plot the z component $-\Omega^z(\mathbf{k})$, use the `gnuplot` script `Fe-curv_z.gnu` generated at runtime,

```
myshell> gnuplot
gnuplot> load 'Fe-curv_z.gnu'
```

(compare with Fig. 2 of Ref. [10]). You can similarly plot the x and y components.

In Example 17 we plotted the Fermi-surface contours on the (010) plane in k-space. To add on top a heatmap plot of $-\boldsymbol{\Omega}(\mathbf{k})$, uncomment the following instructions in `Fe.win`,

```
kslice = T
kslice_task = energy_cntr+curv_heatmap
kslice_kmesh = 200
kslice_corner = 0.0 0.0 0.0
kslice_b1 = 0.5 -0.5 -0.5
kslice_b2 = 0.5 0.5 0.5
```

and re-run `postw90`. For the visualization we use the three `python` scripts generated at runtime. To plot the z component $-\Omega^z(\mathbf{k})$, for example, issue

```
myshell> python Fe-curv_z-heatmap.py
```

Compare the plot with Fig. 3 in Ref. [10].

The intrinsic anomalous Hall conductivity (AHC) $\sigma^{\rm AH}_{\alpha\beta} = -\sigma^{\rm AH}_{\beta\alpha}$ is proportional to the integral of the Berry curvature over the entire Brillouin zone. To evaluate the AHC using a $25\times25\times25$ k-point mesh, uncomment the following lines in `Fe.win`,

```
berry = T
berry_task = ahc
berry_kmesh = 25 25 25
```

(the last instruction can be simplified to `berry_kmesh = 25`) and re-run `postw90`. The AHC in S/cm is written in the output file `Fe.wpout` in (pseudo)vector form. The x, y, and z components correspond to $\sigma^{\rm AH}_{yz}$, $\sigma^{\rm AH}_{zx}$, and $\sigma^{\rm AH}_{xy}$ respectively. For bcc Fe with the magnetization along [001], only the last component is nonzero.

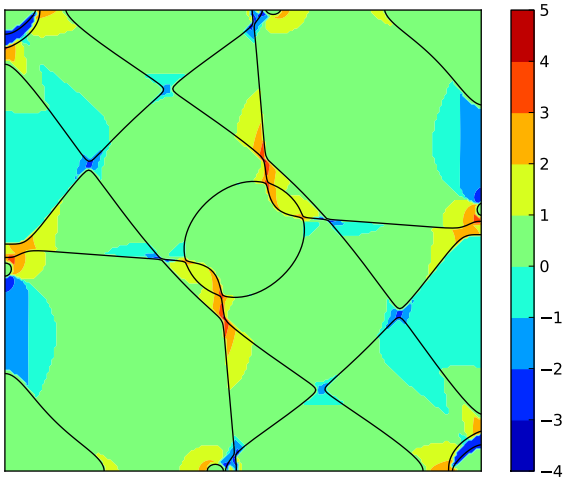

Figure 11: Berry curvature $-\Omega^z(\mathbf{k})$ on the (010) plane of bcc Fe, in atomic units. The color scale is as follows: a positive value, e.g., 2, means 10^2 bohr^2, while a negative value, e.g., -2, means -10^2 bohr^2.

Further ideas

Study the convergence of the AHC with respect to the density of the integration mesh:

- Increase the mesh density uniformly by changing `berry_kmesh`.
- In addition, it helps to adaptively refine the mesh in regions where the Berry curvature is large and rapidly varying (such regions can be seen in Fig. 11). Add to `Fe.win` the lines

  ```
  berry_adpt_kmesh = 5
  berry_adpt_kmesh_thresh = 27.98
  ```

  in order to interpose a $5 \times 5 \times 5$ fine mesh around those points where $|\Omega^z(\mathbf{k})|$ exceeds 27.98 Å^2 $\simeq$ 100 bohr^2. The percentage of points triggering adaptive refinement is given in `Fe.wpout`.
- Compare the converged AHC value with those reported in Refs. [9] and [10].

19: Iron – Orbital magnetization

- Outline: *Calculate the bulk orbital magnetization by Wannier interpolation, as described in Ref. [11].*
- Directory: `examples/example19/`
- Input files
  - `Fe.scf` *The* PWSCF *input file for ground state calculation*
  - `Fe.nscf` *The* PWSCF *input file to obtain Bloch states on a uniform grid*
  - `Fe.pw2wan` *The input file for* `pw2wannier90`
  - `Fe.win` *The* `wannier90` *and* `postw90` *input file*

The sequence of steps below is the same of Examples 17 and 18. If you have already run one of those examples, you can reuse the output files from steps 1–3 and 5. Steps 4 and 6 should be carried out again using the new input files `Fe.pw2wan` and `Fe.win`.

1. Run PWSCF to obtain the ground state of iron
   `pw.x < Fe.scf > scf.out`
2. Run PWSCF to obtain the Bloch states on a uniform k-point grid
   `pw.x < Fe.nscf > nscf.out`
3. Run `wannier90` to generate a list of the required overlaps (written into the `Fe.nnkp` file).
   `wannier90.x -pp Fe`
4. Run `pw2wannier90` to compute:
   - The overlaps $\langle u_{n\mathbf{k}} | u_{m\mathbf{k}+\mathbf{b}} \rangle$ (written in the `Fe.mmn` file)
   - The projections for the starting guess (written in the `Fe.amn` file)
   - The matrix elements $\langle u_{n\mathbf{k}+\mathbf{b}_1} | H_{\mathbf{k}} | u_{m\mathbf{k}+\mathbf{b}_2} \rangle$ (written in the `Fe.uHu` file)
   - The spin matrix elements $\langle \psi_{n\mathbf{k}} | \sigma_i | \psi_{m\mathbf{k}} \rangle$ (written in the `Fe.spn` file)

   `pw2wannier90.x < Fe.pw2wan > pw2wan.out`
5. Run `wannier90` to compute the MLWFs.
   `wannier90.x Fe`
6. Run `postw90` to compute the orbital magnetization.
   `postw90.x Fe` (serial execution)
   `mpirun -np 8 postw90.x Fe` (example of parallel execution with 8 nodes)

The orbital magnetization is computed in the `berry` module of `postw90` as the BZ integral of a quantity $\mathbf{M}_{\rm orb}(\mathbf{k})$ similar to the Berry curvature (see Ch. 11 in the `wannier90` User Guide)

```
berry = T
berry_task = morb
berry_kmesh = 25 25 25
```

As in the case of the AHC, it is necessary to specify the Fermi level in order to compute $\mathbf{M}_{\rm orb}$,

```
fermi_energy = 12.6285
```

Compare the value of the orbital magnetization in `Fe.wpout` with the spin magnetization in `scf.out`.[3] Both point along $\hat{\mathbf{z}}$, but the orbital contribution is much smaller (it is strongly quenched by the crystal field).

Similarly to the AHC in Example 18, the distribution of the orbital magnetization in k-space can be visualized by plotted along lines or planes in the BZ. To plot $M^z_{\rm orb}(\mathbf{k})$ along high-symmetry lines, uncomment in `Fe.win` the block of instructions containing

```
kpath_task = morb
```

After running `postw90`, do

```
myshell> gnuplot
gnuplot> load 'Fe-bands.gnu'
gnuplot> load 'Fe-morb_z.gnu'
```

Compare the two plots with Fig. 2 of Ref. [11].

To plot $M^z_{\rm orb}(\mathbf{k})$ together with the Fermi contours on the (010) BZ plane, uncomment in `Fe.win` the block of instructions containing

```
kslice_task = energy_cntr+morb_heatmap
```

re-run `postw90`, and issue

```
myshell> python Fe-morb_z-heatmap.py
```

The orbital magnetization is more evenly distributed in k-space than the AHC (Fig. 11), and therefore converges more rapidly with the number of k-points used to sample the BZ.

[3] Alternatively, the spin magnetization can be computed by `postw90`. This requires adding the instruction `spin_moment = T` to `Fe.win`, assuming that `pw2wannier90` was executed with `write_spn = .true.` in `Fe.pw2wan`.

20: Iron – Optical conductivity and joint density-of-states

- Outline: *Calculate the joint density-of-states (JDOS) and interband optical conductivity by Wannier interpolation, as described in Ref. [7].*
- Directory: `examples/example20/`
- Input files
  - `Fe.scf` *The* PWSCF *input file for ground state calculation*
  - `Fe.nscf` *The* PWSCF *input file to obtain Bloch states on a uniform grid*
  - `Fe.pw2wan` *The input file for* `pw2wannier90`
  - `Fe.win` *The* `wannier90` *and* `postw90` *input file*

The sequence of steps below is the same of Examples 17–19. If you have already run one of those examples, you can reuse the output files from steps 1–5, and only step 6 must be carried out again using the new input file `Fe.win`.

1. Run PWSCF to obtain the ground state of iron
   `pw.x < Fe.scf > scf.out`
2. Run PWSCF to obtain the Bloch states on a uniform k-point grid
   `pw.x < Fe.nscf > nscf.out`
3. Run `wannier90` to generate a list of the required overlaps (written into the `Fe.nnkp` file).
   `wannier90.x -pp Fe`
4. Run `pw2wannier90` to compute:
   - The overlaps $\langle u_{n\mathbf{k}} | u_{m\mathbf{k}+\mathbf{b}} \rangle$ (written in the `Fe.mmn` file)
   - The projections for the starting guess (written in the `Fe.amn` file)
   - The spin matrix elements $\langle \psi_{n\mathbf{k}} | \sigma_i | \psi_{m\mathbf{k}} \rangle$ (written in the `Fe.spn` file)[4]

   `pw2wannier90.x < Fe.pw2wan > pw2wan.out`
5. Run `wannier90` to compute the MLWFs.
   `wannier90.x Fe`
6. Run `postw90` to compute the JDOS and optical conductivity.
   `postw90.x Fe` (serial execution)
   `mpirun -np 8 postw90.x Fe` (example of parallel execution with 8 nodes)

The following lines in `Fe.win`

```
berry = T
berry_kmesh = 25
berry_task = optics
optics_energy_max = 7.0
optics_time_parity = even
```

[4] These are only used in connection with `spin_decomp` in "Further ideas."

instruct `postw90` to calculate the time-even (symmetric) part $\mathrm{Re}\,\sigma^{\mathrm{S}}_{\alpha\beta}(\omega) = \mathrm{Re}\,\sigma^{\mathrm{S}}_{\beta\alpha}(\omega)$ of the absorptive optical conductivity tensor in the range $\omega \in [0, 7]$ eV, using a 25^3 k-point mesh. The JDOS is also calculated over the same spectral range. Similarly to the AHC (Example 18) and orbital magnetization (Example 19), the value of the Fermi energy must be specified in `Fe.win`,

```
fermi_energy = 12.6285
```

To plot the DOS using `gnuplot`, do

```
myshell> gnuplot
gnuplot> plot 'Fe-jdos.dat' w l
```

In general $\mathrm{Re}\,\sigma^{\mathrm{S}}_{\alpha\beta}(\omega)$ has six independent components, but in bcc Fe only the diagonal components are nonzero. To plot $\mathrm{Re}\,\sigma^{\mathrm{S}}_{xx}(\omega)$ with `gnuplot`, use

```
myshell> gnuplot
gnuplot> plot 'Fe-sigS_xx.dat' w l
```

(As indicated in `Fe.wpout`, the conductivity is plotted in units of 10^{15} sec^{-1}.) The difference in shape between the JDOS and the optical conductivity can be attributed to the variability in the dipole matrix elements describing the optical transitions. Note also that the calculated spectrum is "sharp," while experimental spectra are broadened by lifetime effects which are not taken into account in the calculation.

The optical conductivity tensor of iron also has an antisymmetric (time-odd) part associated with magneto-optical effects. To calculate the absorptive part $\mathrm{Im}\,\sigma^{\mathrm{A}}_{\alpha\beta}(\omega)$, change in `Fe.win`,

```
optics_time_parity = odd
```

and re-run `postw90`. The only nonzero component is $\mathrm{Im}\,\sigma^{\mathrm{A}}_{xy}(\omega) = -\mathrm{Im}\,\sigma^{\mathrm{A}}_{yx}(\omega)$. It describes the differential absorption of left- anf right-circularly-polarized light travelling along the magnetization direction $\hat{\mathbf{z}}$. To plot this "magnetic circular dichroism" (MCD) spectrum, issue

```
myshell> gnuplot
gnuplot> plot 'Fe-sigA_xy.dat' w l
```

Following a common convention in the literature, what is plotted is $\omega\mathrm{Im}\,\sigma^{\mathrm{A}}_{xy}(\omega)$, in units of 10^{29} sec^{-2}. Compare the spectrum with the one displayed in Ref. [10]. To improve the agreement, you need to increase the number of k-points from 25^3 to, e.g., 125^3.

Further ideas

- Recompute the ordinary and dichroic optical conductivities after adding to the input file the line

  ```
  spin_decomp = T
  ```

  This adds three extra columns to the *sig*.dat output files, containing the breakdown of $\sigma(\omega)$ into up $\to$ up, down $\to$ down, and spin-flip transitions.

- The MCD spectrum is related to the AHC by a Kramers-Kronig relation,

  $$\sigma_{xy}^{\mathrm{AH}} = \frac{2}{\pi}\int_0^\infty \frac{1}{\omega}\mathrm{Im}\,\sigma_{xy}^{\mathrm{A}}(\omega)\,d\omega$$

  This integral is evaluated numerically by `postw90` by replacing the limits of integration with $\int_{\omega_{\mathrm{min}}}^{\omega_{\mathrm{max}}}$, where ω_{max} equals `optics_max_energy` and ω_{min} is scanned in the range $[0,\omega_{\mathrm{max}}]$. To plot the result (in S/cm) versus ω_{min}, do

  ```
  myshell> gnuplot
  gnuplot> plot 'Fe-kk_xy.dat' w l
  ```

  Compare the value at $\omega_{\mathrm{min}} = 0$ with the AHC value calculated in Example 18 by directly integrating the k-space Berry curvature.

- More generally, the Kramers-Kronig relation allows to compute the AHC at finite frequencies, starting from the MCD spectrum:

  $$\sigma_{xy}^{\mathrm{AH}}(\omega) = \frac{2}{\pi}\int_0^\infty \frac{\mathrm{Im}\,\sigma_{xy}^{\mathrm{A}}(\omega')}{\omega'^2-\omega^2}\,d\omega'$$

  Write a script/program that reads the MCD datafile generated by `postw90`, and carries out the (truncated) Kramers-Kronig integral over frequencies to calculate $\sigma_{xy}^{\mathrm{AH}}(\omega)$. Plot the result versus ω, and compare with Fig. 5 (lower panel) of Ref. [10].

References

[1] A. A. Mostofi, G. Pizzi, I. Souza and J. R. Yates, wannier90: User Guide, available at http://www.wannier.org/user_guide.html.

[2] N. Marzari and D. Vanderbilt, Maximally Localized Generalized Wannier Functions for Composite Energy Bands, *Phys. Rev. B* **56**, 12847 (1997).

[3] I. Souza, N. Marzari and D. Vanderbilt, Maximally Localized Wannier Functions for Entangled Energy Bands, *Phys. Rev. B* **65**, 035109 (2001).

[4] A. A. Mostofi, J. R. Yates, Y.-S. Lee, I. Souza, D. Vanderbilt and N. Marzari, wannier90: A Tool for Obtaining Maximally-Localized Wannier Functions, *Comput. Phys. Commun.*, 178, 685 (2008) and http://arxiv.org/abs/0708.0650.

[5] D. Vanderbilt, Soft Self-Consistent Pseudopotentials in a Generalized Eigenvalue Formalism, *Phys. Rev. B* **41** (11), 7892 (1990).

[6] N. Marzari and D. Vanderbilt, Maximally-Localized Wannier Functions in Perovskites: $BaTiO_3$, http://arxiv.org/abs/cond-mat/9802210.

[7] J. R. Yates *et al.*, Spectral and Fermi Surface Properties from Wannier Interpolation, *Phys. Rev. B* **75**, 195121 (2007).

[8] Y.-S. Lee, M. B. Nardelli and N. Marzari, Band Structure and Quantum Conductance of Nanostructures from Maximally Localized Wannier Functions: The Case of Functionalized Carbon Nanotubes, *Phys. Rev. Lett.* **95**, 076804 (2005).

[9] X. Wang *et al.*, Ab-initio calculation of the anomalous Hall conductivity by Wannier interpolation, *Phys. rev. B* **74**, 195118 (2006).

[10] Y. Yao *et al.*, First Principles Calculation of Anomalous Hall Conductivity in Ferromagnetic bcc Fe, *Phys. Rev. Lett.* **92**, 037204 (2004).

[11] M. G. Lopez, D. Vanderbilt, T. Thonhauser, and I. Souza, Wannier-based calculation of the orbital magnetization in crystals, Phys. Rev. B **85**, 014435 (2012).