wannier90: Tutorial

Version 2.0

14th October 2013

Contents

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 [4] and to a paper [5] 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 is used in examples 17–19
  http://www.python.org
  http://www.numpy.org
  http://matplotlib.org

Parallel execution

Presently, wannier90.x is a serial-only executable, so it cannot be run in parallel using MPI libraries. On the contrary, postw90.x 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`. 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. The tutorial examples work with PWSCF v5.0.0 - v5.0.2 with the exception of the Berry phase related properties in examples 17-19 which require the very latest version of `pw2wannier90.f90`. This can be found in the directory `pwscf/v5.0` in the wannier distribution. It should be moved to `PP/src` in the PWSCF distribution and compiled using `make pp`. We expect this will be included as standard in the next release of PWSCF

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 and 3rd MLWFs use

   ```
   wannier_plot_list = 1 3
   ```

   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
   ```

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:* sp^3 *hybrid orbitals*

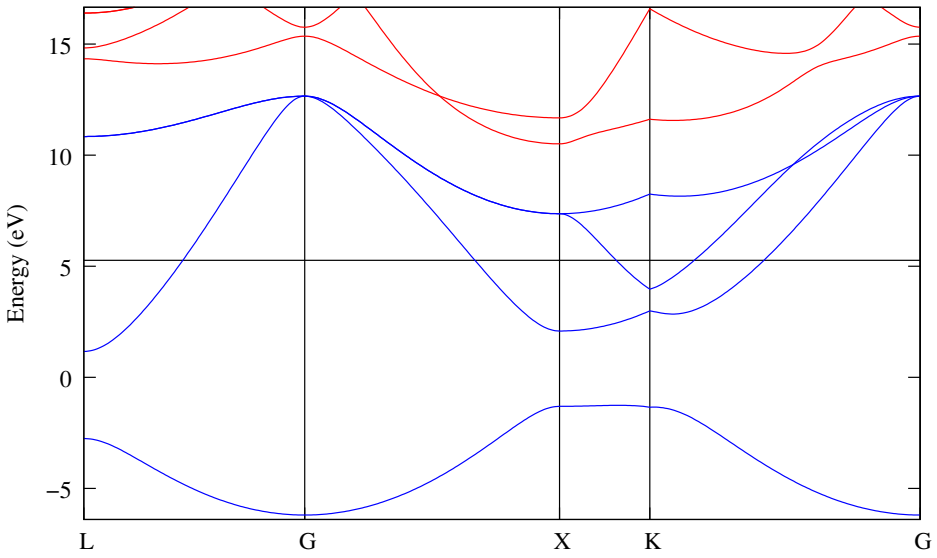

Figure 1: Bandstructure of lead showing the position of the Fermi level. Only the lowest four bands are included in the calculation.

- 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.

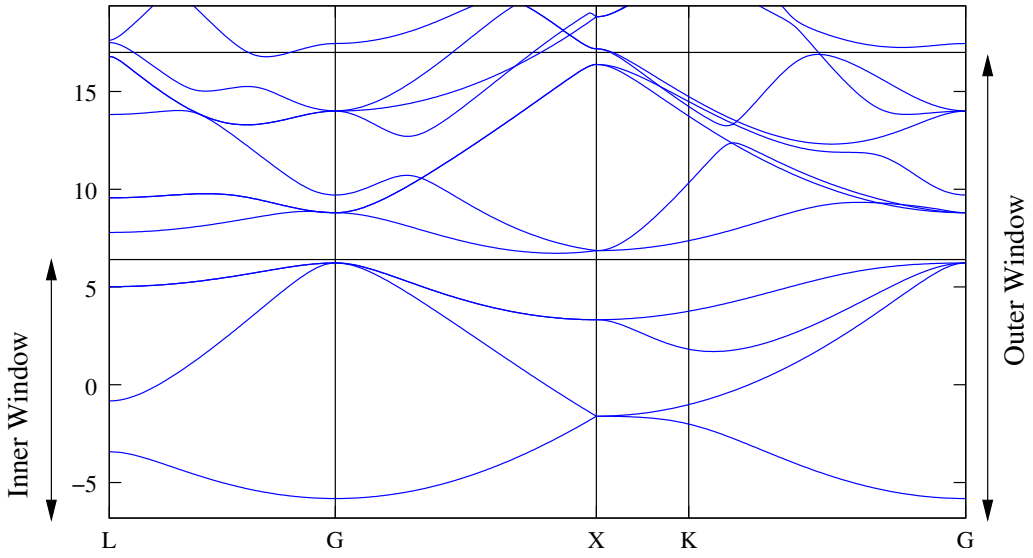

Figure 2: Bandstructure of silicon showing the position of the outer and inner energy windows.

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 [6] 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.

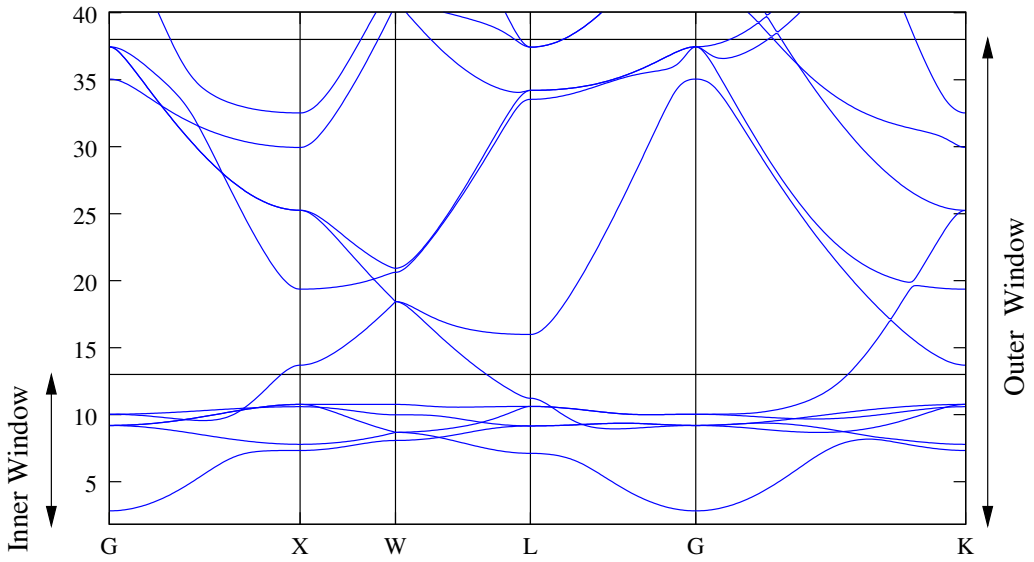

Figure 3: Bandstructure of copper showing the position of the outer and inner energy windows.

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.

Note that both the `PWSCF` executable `pw.x` *and* `pw2wannier90.x` can be run in parallel, which for large calculations can reduce the computation time very significantly. This requires compiling the code in its parallel version, using the MPI libraries. Refer to the QUANTUM-ESPRESSO package for the documentation on how to do so. 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-only, `postw90.x` can be run in parallel. In this case any number of processors can be used, independently of the number 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. To plot the bandstructure you can use the PWSCF tool `bands.x` or the small fortran program available at `http://www.tcm.phy.cam.ac.uk/~jry20/bands.html`.
- 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, DOS, projected WFs versus MLWFs

- Outline: *Generate both maximally-localized and projected Wannier functions for ferromagnetic bcc Fe. Calculate the total and orbital-projected density of states by Wannier interpolation.*
- Directory: `examples/example8/`
- Input Files
  - `iron.scf` *The* PWSCF *input file for the spin-polarized ground state calculation*
  - `iron.nscf` *The* PWSCF *input file to obtain Bloch states on a uniform grid*
  - `iron_{up,down}.pw2wan` *Input files for* `pw2wannier90`
  - `iron_{up,down}.win` *Input files for* `wannier90` *and* `postw90`
- Note that in a spin-polarized calculation the spin-up and spin-down MLWFs are computed separately. (The more general case of spinor WFs will be treated in Example 17.)

1. Run PWSCF to obtain the ferromagnetic ground state of bcc Fe
   `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 `.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 `.mmn` and `.amn` files).
   `pw2wannier90.x < iron_up.pw2wan > pw2wan_up.out`
   `pw2wannier90.x < iron_dn.pw2wan > pw2wan_dn.out`
5. Run `wannier90` to compute the MLWFs.
   `wannier90.x iron_up`
   `wannier90.x iron_dn`

Density of states

To compute the DOS using a $25 \times 25 \times 25$ k-point grid add to the two `.win` files

```
dos = true
dos_kmesh = 25
```

run `postw90`,

```
postw90.x iron_up
postw90.x iron_dn
```

and 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
```

Energies are referred to the Fermi level (12.6256 eV, from `scf.out`). Note the exchange splitting between the up-spin and down-spin DOS. Check the convergence by repeating the DOS calculations with more k-points.

Projected versus maximally-localized Wannier functions

In the calculations above we chose s, p, and d-type trial orbitals in the `.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-centred, with spreads organized in three groups, 1+3+5: one s, three p, and five d. Now look at the final state towards the end of the file. The Wannier spreads have re-organized in two groups, 6+3; moreover, the six more diffuse WFs are off-centred: the initial atomic-like orbitals hybridized with one another, becoming more localized in the process. It is instructive to visualize the final-state MLWFs using `XCrySDen`, following Example 1. For more details, see Sec. IV.B of Ref. [7].

Let us plot the evolution of the spread functional Ω,

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

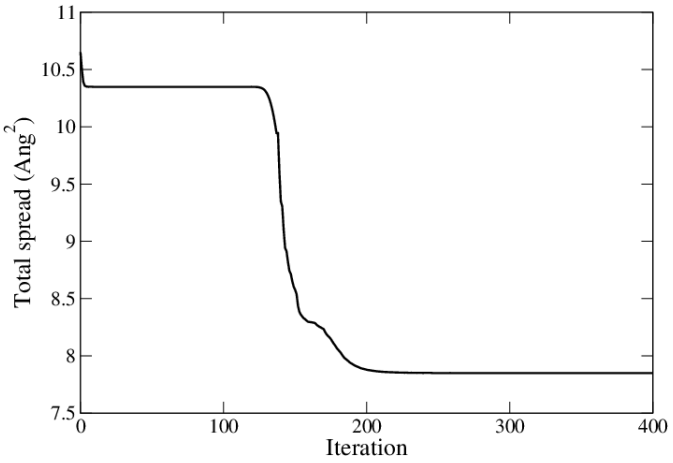

Figure 4: Evolution of the Wannier spread Ω of the minority (spin-up) bands of bcc Fe during the iterative minimization of $\widetilde{\Omega}$, starting from s, p, and d-type trial orbitals.

The first plateau corresponds to atom-centred WFs of separate s, p, and d character, and the sharp drop signals the onset of the hybridization. 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
```

With this choice the minimization converges much more rapidly.

Any reasonable set of localized WFs spanning the states of interest can be used to compute physical quantities (they are "gauge-invariant"). Let us recompute the DOS using, instead of MLWFs, the WFs obtained by projecting onto s, p, and d-type trial orbitals, without further iterative minimization of the spread functional. This can be done by setting

```
num_iter = 0
```

But note that we still need to do disentanglement! Recalculate the DOS to confirm that it is almost identical to the one obtained earlier using the hybridized set of MLWFs. Visualize the projected WFs using `XCrySDen`, to see that they retain the pure orbital character of the individual trial orbitals.

Orbital-projected DOS and exchange splitting

With projected WFs the total DOS can be separated into s, p and d contributions, in a similar way to the orbital decomposition of the energy bands in Example 4.

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

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

and re-run `postw90`. Plot the projected DOS for both up- and down-spin bands. Repeat for the s and d projections.

Projected WFs can also be used to quantify more precisely the exchange splitting between majority and minority states. Re-run `wannier90` after setting `dos=false` and adding to the `.win` files

```
write_hr_diag = true
```

This instructs `wannier90` to print in the output file the on-site energies $\langle \mathbf{0}n|H|\mathbf{0}n\rangle$. The difference between corresponding values in `iron_up.wout` and in `iron_dn.wout` gives the exchange splittings for the individual orbitals. Compare their magnitudes with the splittings displayed by the orbital-projected DOS plots. In agreement with the Stoner criterion, the largest exchange splittings occur for the localized d-states, which contribute most of the density of states at the Fermi level.

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 [8], 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. [8], 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 low-lying conduction states

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.

Further ideas

- Compare the Wannier-interpolated bandstructure with the full PWSCF bandstructure. Recompute the 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 [9].
- Compute four MLWFs spanning the low-lying conduction states (see Ref. [3]).

12: Benzene – Valence and low-lying conduction states

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 $\Omega_{\rm I}$. Then, we minimise $\Omega_{\rm O} + \Omega_{\rm OD}$.
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 [10]. 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.

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*

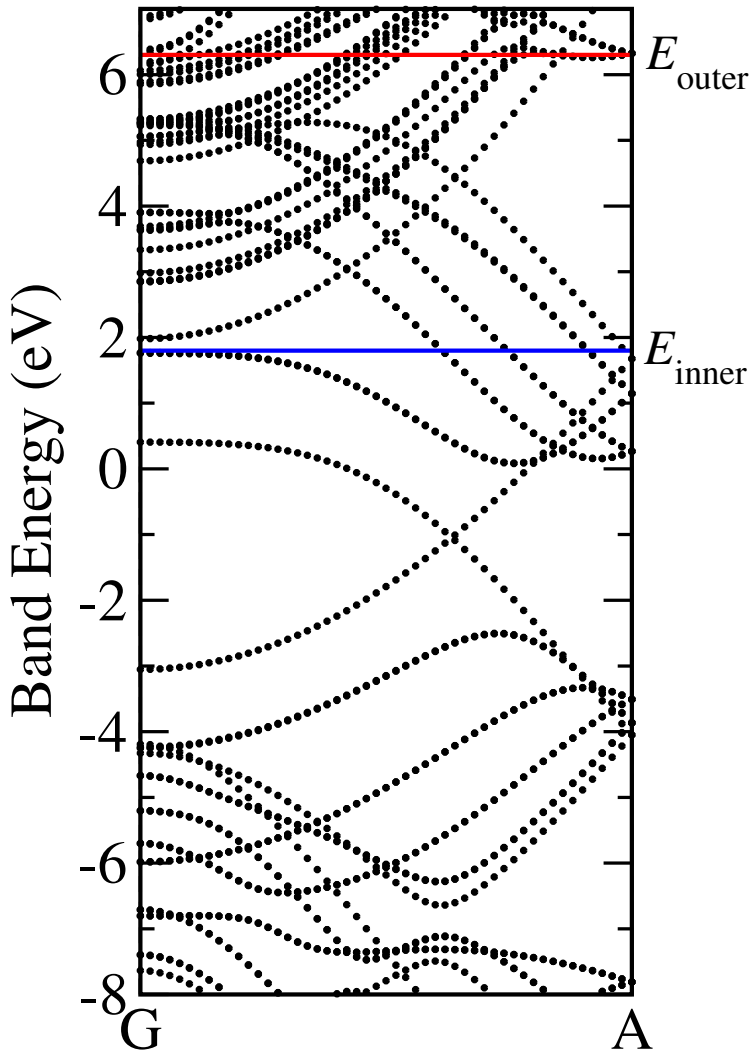

Figure 5: Bandstructure of (5,5) carbon nanotube showing the position of the outer and inner energy windows.

- `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
```

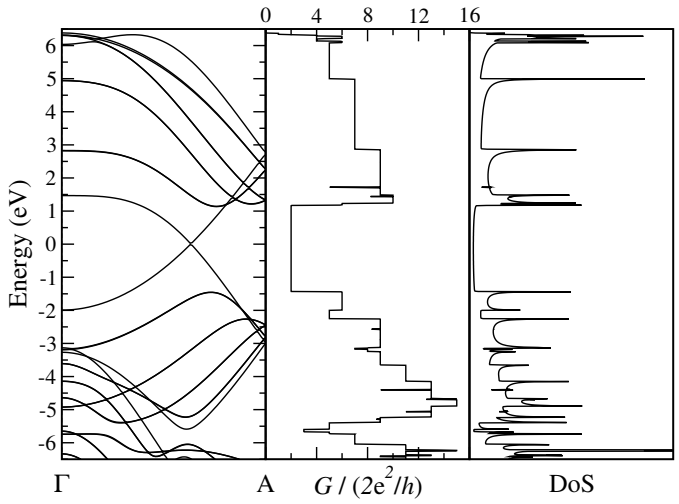

Figure 6: Wannier interpolated bandstructure, quantum conductance and density of states of (5,5) carbon nanotube. Note that the Fermi level has been shifted by 1.06eV with respect to Fig. 5.

```
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 contains 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.

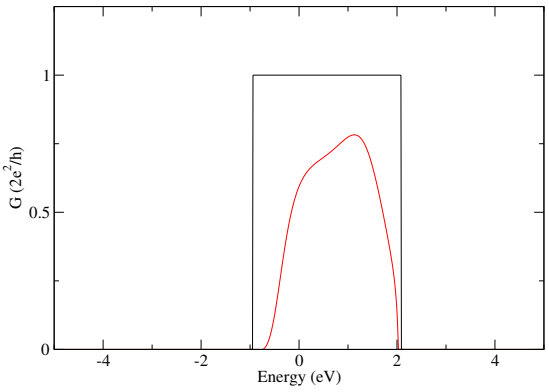

Figure 7: Quantum conductance of periodic Sodium chain (black) compared to that of the defected Sodium chain (red).

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.

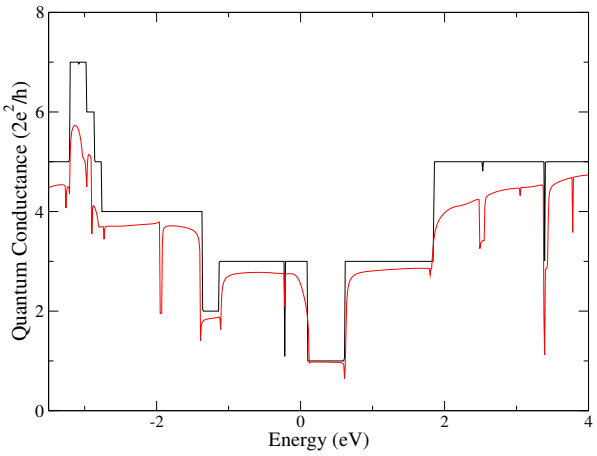

Figure 8: Quantum conductance of infinite pristine nanotube (black) compared to that of the infinite nanotube with the substitutional silicon defect (red).

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.*

If you want to use Quantum ESPRESSO

- Directory: `examples/example16-withqe/`
- 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*

If you do not want to use Quantum ESPRESSO

- Directory: `examples/example16-noqe/`
- Input Files
  - `Si.win` *The* `wannier90` *and* `postw90` *input file*
  - `Si.mmn` *The overlap matrices* $\mathbf{M}^{(\mathbf{k},\mathbf{b})}$
  - `Si.amn` *Projection* $\mathbf{A}^{(\mathbf{k})}$ *of the Bloch states onto a set of trial localised orbitals*
  - `Si.eig` *The Bloch eigenvalues at each k-point. For interpolation only*

Note the first five steps in the following are the same of Example 11, and are needed only if you want to use the `PWscf` code of Quantum ESPRESSO. Otherwise, if you have already run Example 11 with Quantum ESPRESSSO (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.

If instead you do not have Quantum ESPRESSO installed, or you do not want to use it, you can start from step 5 using the files in the `examples/example16-noqe/` folder.

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 MPI processes)

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 negligibile (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: It is recommended that you go through Example 8 first (bcc Fe without spin-orbit).

Note: This example requires a recent version of the `pw2wannier90` interface.

- Outline: *Plot the spin-orbit-coupled bands of ferromagnetic bcc Fe. 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 = true
```

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

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 $\langle \psi_{n\mathbf{k}} | \sigma_i | \psi_{m\mathbf{k}} \rangle$, $i = x, y, z$ (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.
   `postw90.x Fe` (serial execution)

   Note: the routines for `kpath=true` and `kslice=true` are currently not parallelized over k-points.

[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 = true
kpath_task = bands
kpath_bands_colour = spin
kpath_num_points=500
```

To plot the bands using `gnuplot` (version 4.2 or higher) issue

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

or, using `python`,

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

Next we plot the Fermi-surface contours on the (010) plane $k_y = 0$, using the `kslice` module. Set `kpath = false` and uncomment the following instructions in `Fe.win`,

```
kslice = true
kslice_task = fermi_lines
kslice_fermi_level = [insert your value here]
kslice_corner = 0.0 0.0 0.0
kslice_b1 = 0.5 -0.5 -0.5
kslice_b2 = 0.5 0.5 0.5
kslice_2dkmesh = 200 200
```

taking the Fermi level value from `scf.out`. The energy eigenvalues are computed on a 200×200 k-point grid covering the BZ slice. The lines of intersection between the Fermi surface and the (010) plane can be visualized with the `gnuplot` or `python` scripts generated at runtime,

```
myshell> gnuplot
gnuplot> load 'Fe-kslice-fermi_lines.gnu'
```

or

```
myshell> python Fe-kslice-fermi_lines.py
```

The Fermi lines can be colour-coded by the spin expectation value $\langle S_z \rangle$ of the states on the Fermi surface. Add to `Fe.win` the line

```
kslice_fermi_lines_colour = spin
```

and re-run `postw90`. The names of the `gnuplot` and `python` scripts generated at runtime are unchanged. (However, the plotting algorithm is different in this case, and the lines are not as smooth as before. You may want to increase `kslice_2dkmesh`.)

Further ideas

- Redraw the Fermi surface contours on the (010) plane starting from a calculation without spin-orbit coupling, by adding to the input files `iron_{up,down}.win` in Example 8 the lines

```
kslice = true
kslice_task = fermi_lines
kslice_fermi_level = [insert your value here]
kslice_corner = 0.0 0.0 0.0
kslice_b1 = 0.5 -0.5 -0.5
kslice_b2 = 0.5 0.5 0.5
kslice_2dkmesh = 200 200
```

  before running `postw90`,

```
postw90.x iron_up
postw90.x iron_dn
```

  The `python` scripts generated at runtime draw the up- and down-spin Fermi lines on separate figures. To draw them together, use the script `iron_updn-kslice-fermi_lines.py` provided with Example 17 (or merge the two generated scripts). Compare the Fermi lines with and without spin-orbit, and note the spin-orbit-induced avoided crossings.

- In Example 8 we obtained MLWFs separately for the up- and down-spin channels of bcc Fe without spin-orbit. The Wannier-interpolated DOS was therefore automatically separated into minority and majority contributions. For a spinor calculation we can still spin-decompose the DOS, using

```
dos = true
spin_decomp = true
dos_kmesh = 25 25 25
```

  The data file `Fe-dos.dat` created by `postw90` contains the up-spin and down-spin contributions in the third and fourth columns,

```
myshell> gnuplot
gnuplot> plot 'Fe-dos.dat' u (-$3):($1-12.6285) w l,'Fe-dos.dat' u ($4):($1-12.6285)
w l
```

  (You should replace 12.6285 with your value of the Fermi energy). An alternative approach is to project the DOS onto the up-spin and down-spin WFs separately. To find the DOS projected onto the up-spin (odd-numbered) WFs replace `spin_decomp = true` with

```
dos_project = 1,3,5,7,9,11,13,15,17
```

  and re-run `postw90`. This approach has the advantage that it does not require the `Fe.spn` file.

18: Iron – Berry curvature, anomalous Hall conductivity and optical conductivity

Note: This example requires a recent version of the `pw2wannier90` interface.

- Outline: *Calculate the Berry curvature, anomalous Hall conductivity, and (magneto)optical conductivity of ferromagnetic bcc Fe with spin-orbit coupling. In preparation for this example it may be useful to read Ref. [11] and Ch. 11 of the User Guide.*
- 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`
   `postw90.x Fe` (serial execution)
   `mpirun -np 8 postw90.x Fe` (example of parallel execution with 8 MPI processes)

Berry curvature plots

The Berry curvature $\Omega_{\alpha\beta}(\mathbf{k})$ of the occupied states is defined in Eq. (11.18) of the User Guide. The following lines in `Fe.win` are used to calculate the energy bands and the Berry curvature (in bohr^2) along high-symmetry lines in k-space.

```
fermi_energy = [insert your value here]
berry_curv_unit = bohr2
kpath = true
kpath_task = bands+curv
kpath_bands_colour = spin
kpath_num_points = 1000
```

After executing `postw90`, plot the Berry curvature component $\Omega_z(\mathbf{k}) = \Omega_{xy}(\mathbf{k})$ along the magnetization direction using the script generated at runtime,

```
myshell> python Fe-bands+curv_z.py
```

and compare with Fig. 2 of Ref. [11].

In Example 17 we plotted the Fermi lines on the (010) plane $k_y = 0$. To combine them with a heatmap plot of (minus) the Berry curvature set `kpath = false`, uncomment the following lines in `Fe.win`,

```
kslice = true
kslice_task = curv+fermi_lines
kslice_corner = 0.0 0.0 0.0
kslice_b1 = 0.5 -0.5 -0.5
kslice_b2 = 0.5 0.5 0.5
kslice_2dkmesh = 200 200
```

re-run `postw90`, and issue

```
myshell> python Fe-kslice-curv_z+fermi_lines.py
```

Compare with Fig. 3 in Ref. [11]. Note how the Berry curvature "hot-spots" tend to occur near spin-orbit-induced avoided crossings (the Fermi lines with and without spin-orbit were generated in Example 17).

Anomalous Hall conductivity

The intrinsic anomalous Hall conductivity (AHC) is proportional to the BZ integral of the Berry curvature. In bcc Fe with the magnetization along $\hat{\mathbf{z}}$, the only nonzero components are $\sigma_{xy} = -\sigma_{yx}$. To evaluate the AHC using a $25 \times 25 \times 25$ k-point mesh, set `kslice = false`, uncomment the following lines in `Fe.win`,

```
berry = true
berry_task = ahc
berry_kmesh = 25 25 25
```

and re-run `postw90`. The AHC is written in the output file `Fe.wpout` in vector form. For bcc Fe with the magnetization along [001], only the z-component σ_{xy} is nonzero.

As a result of the strong and rapid variations of the Berry curvature across the BZ, the AHC converges rather slowly with k-point sampling, and a $25 \times 25 \times 25$ does not yield a well-converged value.

- Increase the BZ mesh density by changing `berry_kmesh`.
- To accelerate the convergence, adaptively refine the mesh around spikes in the Berry curvature, by adding to `Fe.win` the lines

```
berry_curv_adpt_kmesh = 5
berry_curv_adpt_kmesh_thresh = 100.0
```

This adds a $5 \times 5 \times 5$ fine mesh around those points where $|\mathbf{\Omega}(\mathbf{k})|$ exceeds 100 bohr^2. The percentage of points triggering adaptive refinement is reported in `Fe.wpout`.

Compare the converged AHC value with those obtained in Refs. [7] and [11].

The Wannier-interpolation formula for the Berry curvature comprises three terms, denoted D-D, D-$\overline{A}$, and $\overline{\Omega}$ in Ref. [7], and $J2$, $J1$, and $J0$ in Ref. [12]. To report in `Fe.wpout` the decomposition of the total AHC into these three terms, set `iprint` (verbosity level) to a value larger than one in `Fe.win`.

Optical conductivity

The optical conductivity tensor of bcc Fe with magnetization along $\hat{\mathbf{z}}$ has the form

$$\boldsymbol{\sigma} = \boldsymbol{\sigma}^{\mathrm{S}} + \boldsymbol{\sigma}^{\mathrm{A}} = \begin{pmatrix} \sigma_{xx} & 0 & 0 \\ 0 & \sigma_{xx} & 0 \\ 0 & 0 & \sigma_{zz} \end{pmatrix} + \begin{pmatrix} 0 & \sigma_{xy} & 0 \\ -\sigma_{xy} & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

where "S" and "A" stand for the symmetric and antisymmetric parts and $\sigma_{xx} = \sigma_{yy} \neq \sigma_{zz}$. The dc AHC calculated earlier corresponds to σ_{xy} in the limit $\omega \to 0$. At finite frequency $\sigma_{xy} = -\sigma_{yx}$ acquires an imaginary part which describes magnetic circular dichoism (MCD).

To compute the complex optical conductivy for $\hbar\omega$ up to 7 eV, replace

```
berry_task = ahc
```

with

```
berry_task = kubo
```

add the line

```
kubo_freq_max = 7.0
```

and re-run `postw90`. Reasonably converged spectra can be obtained with a $125 \times 125 \times 125$ k-point mesh. Let us first plot the ac AHC in S/cm, as in the lower panel of Fig. 5 in Ref. [11],

```
myshell> gnuplot
gnuplot> plot 'Fe-kubo-A_xy.dat' u 1:2 w l
```

Comapare the $\omega \to 0$ limit with the result obtained earlier by integrating the Berry curvature.[3]

Next we plot the MCD spectrum. Following Ref. [11], we plot $\mathrm{Im}[\omega\sigma_{xy}(\hbar\omega)]$, in units of 10^{29} sec^{-2}. The needed conversion factor is $9 \times 10^{-18} \times e/\hbar \simeq 0.0137$ (e and $\hbar$ in SI units),

```
gnuplot> set yrange[-5:15]
gnuplot> plot 'Fe-kubo-A_xy.dat' u 1:($1)*($3)*0.0137 w l
```

[3]The calculation of the AHC using `berry_task = kubo` involves a truncation of the sum over empty states in the Kubo-Greenwood formula: see description of the keyword `kubo_eigval_max` in the User Guide. As discussed around Eq. (11.17) of the User Guide, no truncation is done with `berry_task = ahc`.

Further ideas

- Recompute the AHC and optical spectra of bcc Fe using projected s, p, and d-type Wannier functions instead of the hybridrized MLWFs (see Example 8), and compare the results.
- A crude way to model the influence of heterovalent alloying on the AHC is to assume that its only effect is to donate or deplete electrons, i.e., to shift the Fermi level of the pure crystal [13].

  Recalculate the AHC of bcc Fe for a range of Fermi energies within ±0.5 eV of the true Fermi level. This calculation can be streamlined by replacing in `Fe.win`

  ```
  fermi_energy = [insert your value here]
  ```

  with

  ```
  fermi_energy_min = [insert here your value minus 0.5]
  fermi_energy_max = [insert here your value plus 0.5]
  ```

  Use a sufficiently dense BZ mesh with adaptive refinement. To plot σ_{xy} versus ε_F, issue

  ```
  myshell> gnuplot
  gnuplot> plot 'Fe-ahc-fermiscan.dat' u 1:4 w lp
  ```

19: Iron – Orbital magnetization

Note: This example requires a recent version of the `pw2wannier90` interface.

- Outline: *Calculate the orbital magnetization of ferromagnetic bcc Fe by Wannier interpolation.*
- 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)

   ```
   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 MPI processes)

The orbital magnetization is computed as the BZ integral of the quantity $\mathbf{M}^{\rm orb}(\mathbf{k})$ defined in Eq. (11.20) of the User Guide. The relevant lines in `Fe.win` are

```
berry = true
berry_task = morb
berry_kmesh = 25 25 25
fermi_energy = [insert your value here]
```

After running `postw90`, compare the value of the orbital magnetization reported in `Fe.wpout` with the spin magnetization in `scf.out`. Set `iprint = 2` to report the decomposition of $\mathbf{M}^{\rm orb}$ into the terms $J0$, $J1$, and $J2$ defined in Ref. [12].

To plot $M_z^{\rm orb}(\mathbf{k})$ along high-symmetry lines set `berry = false` and uncomment in `Fe.win` the block of instructions containing

```
kpath = true
kpath_task = bands+morb
```

After running `postw90`, issue

```
myshell> python Fe-bands+morb_z.py
```

Compare with Fig. 2 of Ref. [12], bearing in mind the factor of $-1/2$ difference in the definition of $\mathbf{M}^{\rm orb}(\mathbf{k})$ (see Ch. 11 in the User Guide).

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

```
kslice = true
kslice_task = morb+fermi_lines
```

re-run `postw90`, and issue

```
myshell> python Fe-kslice-morb_z+fermi_lines.py
```

$M_z^{\rm orb}(\mathbf{k})$ is much more evenly distributed in k-space than the Berry curvature (see Example 18). As a result, the integrated orbital magnetization converges more rapidly with the BZ sampling.

20: Disentanglement only in small (spherical) regions of k space

$LaVO_3$

- Outline: *Obtain disentangled MLWFs for strained $LaVO_3$.*
- Directory: `examples/example20/`
- Input Files
  - `LaVO3.scf` *The* PWSCF *input file for ground state calculation*
  - `LaVO3.nscf` *The* PWSCF *input file to obtain Bloch states on a uniform grid*
  - `LaVO3.pw2wan` *Input file for* `pw2wannier90`
  - `LaVO3.win` *The* `wannier90` *input file*

1. Run PWSCF to obtain the ground state of silicon
   `pw.x < LaVO3.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 < LaVO3.nscf > nscf.out`
3. Run `wannier90` to generate a list of the required overlaps (written into the `LaVO3.nnkp` file).
   `wannier90.x -pp LaVO3`
4. Run `pw2wannier90` to compute the overlap between Bloch states and the projections for the starting guess (written in the `LaVO3.mmn` and `LaVO3.amn` files).
   `pw2wannier90.x < LaVO3.pw2wan > pw2wan.out`
5. Run `wannier90` to compute the MLWFs.
   `wannier90.x LaVO3`

Inspect the output file `LaVO3.wout`. In the initial summary, you will see that the disentanglement was performed only within one sphere of radius 0.2 arount the point $A = (0.5, 0.5, 0.5)$ in reciprocal space:

```
|  Number of spheres in k-space              :                 1             |
|   center n.   1 :     0.500   0.500   0.500,    radius   =   0.200         |
```

Compare the band structure that Wannier90 produced with the one obtained using Quantum ESPRESSO. You should get something similar to Fig. 9.

Further ideas

- Try to disentangle the Wannier functions without using the mesh. You will see that this is extremely hard due to the bands touching only near the A point (for unstrained $LaVO_3$, instead, there is a gap so disentanglement is not needed).
- Try to run similar simulations for $SrMnO_3$. The input files can be found in the $SrMnO_3$ subfolder. There are three different `.pw2wan` and `.win` files for projections on different states. Accordingly, different spheres are taken into account.

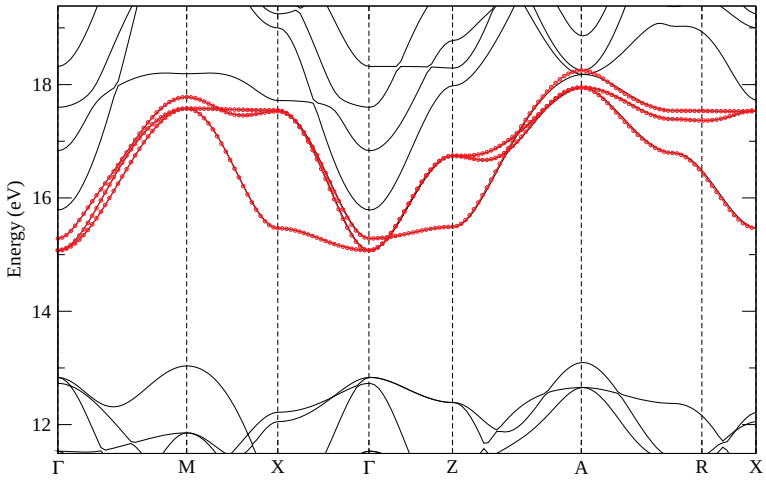

Figure 9: Bandstructure of strained $LaVO_3$. Black: ab-initio bands; red circles: Wannier-interpolated band structure. In order to disentangle the bands, only a small sphere around the A point has been used.

References

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

[2] N. Marzari and D. Vanderbilt, Phys. Rev. B **56**, 12847 (1997).

[3] I. Souza, N. Marzari, and D. Vanderbilt, Phys. Rev. B **65**, 035109 (2001).

[4] N. Marzari, A. A. Mostofi, J. R. Yates, I. Souza, and D. Vanderbilt, Rev. Mod. Phys. **84**, 1419 (2012).

[5] A. A. Mostofi, J. R. Yates, Y.-S. Lee, I. Souza, D. Vanderbilt, and N. Marzari, Comput. Phys. Commun. **178**, 685 (2008).

[6] D. Vanderbilt, Phys. Rev. B **41**, 7892 (1990).

[7] X. Wang, J. R. Yates, I. Souza, and D. Vanderbilt, Phys. Rev. B **74**, 195118 (2006).

[8] N. Marzari and D. Vanderbilt, arXiv:9802210 (1998).

[9] J. R. Yates, X. Wang, D. Vanderbilt, and I. Souza, Phys. Rev. B **75**, 195121 (2007).

[10] Y.-S. Lee, M. B. Nardelli, and N. Marzari, Phys. Rev. Lett. **95**, 076804 (2005).

[11] Y. Yao, L. Kleinman, A. H. MacDonald, J. Sinova, T. Jungwirth, D.-S. Wang, E. Wang, and Q. Niu, Phys. Rev. Lett. **92**, 037204 (2004).

[12] M. G. Lopez, D. Vanderbilt, T. Thonhauser, and I. Souza, Phys. Rev. B **85**, 014435 (2012).

[13] Y. Yao, Y. Liang, D. Xiao, Q. Niu, S.-Q. Shen, X. Dai, and Z. Fang, Phys. Rev. B **75**, 020401 (2007).