wannier90: Tutorial

Version 3.1

20th February 2020

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

postw90.x and wannier90.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.inc 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 v5.1.x and v6.0.x. The exception are the examples on symmetry adapted Wannier functions which require v6.0.x together with the very latest version of pw2wannier90.f90. This can be found in the directory pwscf/v6.0 in the wannier distribution. It should be moved to PP/src in the PWSCF distribution and compiled using make pp. Later versions v6.x.x should have the most up-to-date version of pw2wannier90.f90 already included in the Quantum ESPRESSO distribution.

There are interfaces to a number of other electronic structure codes including ABINIT (http://www.abinit.org), FLEUR (http://www.flapw.de), OpenMX (http://www.openmx-square.org/), GPAW (https://wiki.fysik.dtu.dk/gpaw/), VASP (http://www.vasp.at), and Wien2k (http://www.wien2k.at)

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

   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*

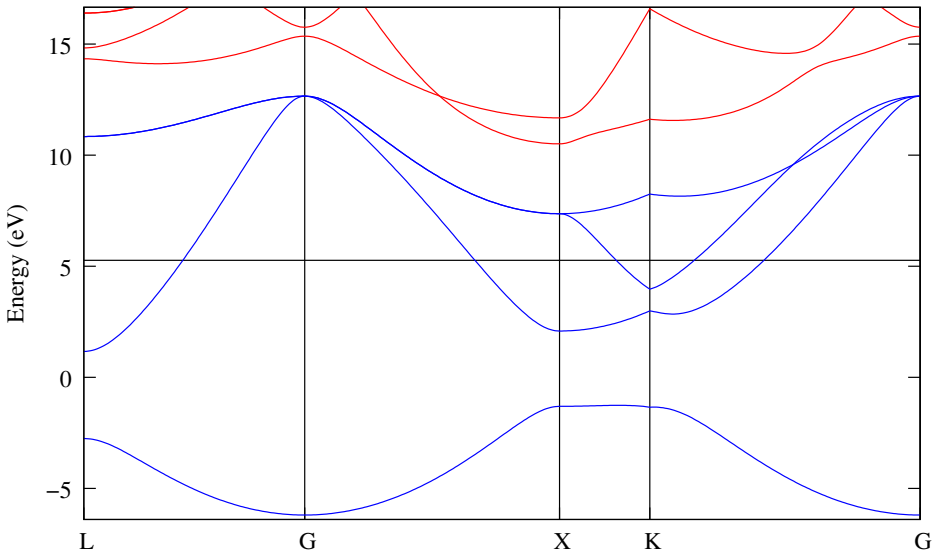


Figure 1: Bandstructure of lead showing the position of the Fermi level. Only the lowest four bands are included in the calculation.

- 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.

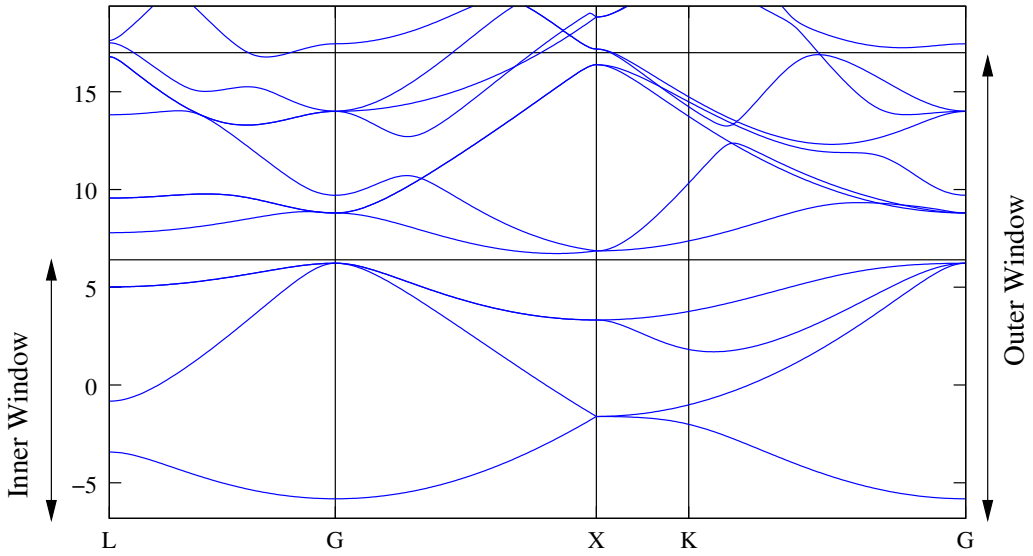


Figure 2: Bandstructure of silicon showing the position of the outer and inner energy windows.

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 [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.

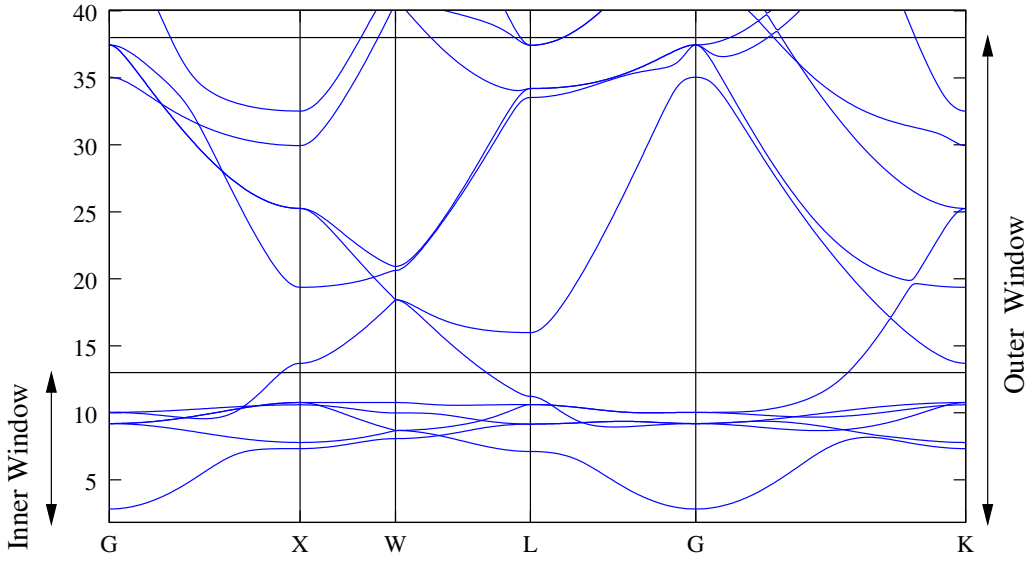


Figure 3: Bandstructure of copper showing the position of the outer and inner energy windows.

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 both the `wannier90` executable and `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
```

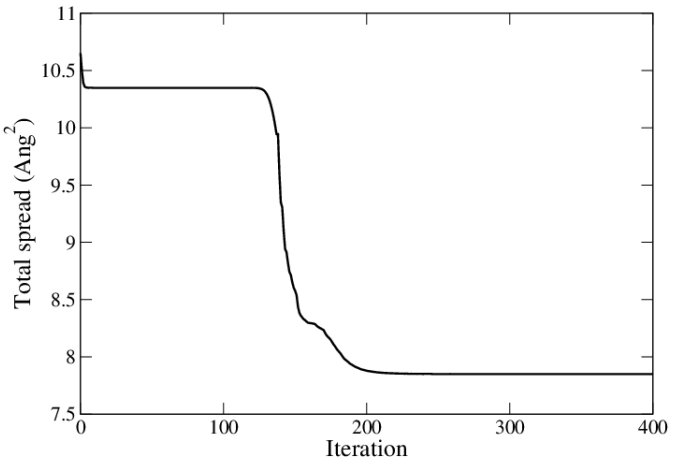


Figure 4: Evolution of the Wannier spread Ω of the minority (spin-up) bands of bcc Fe during the iterative minimization of $\widetilde{\Omega}$, starting from s, p, and d-type trial orbitals.

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*

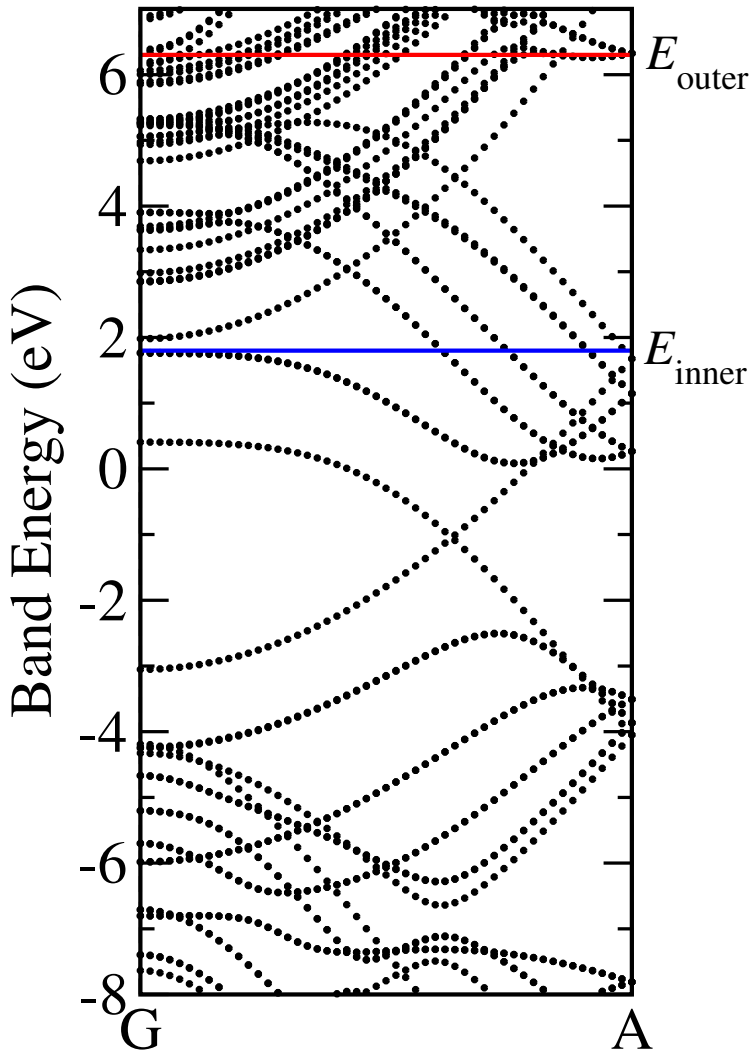


Figure 5: Bandstructure of (5,5) carbon nanotube showing the position of the outer and inner energy windows.

- `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
```

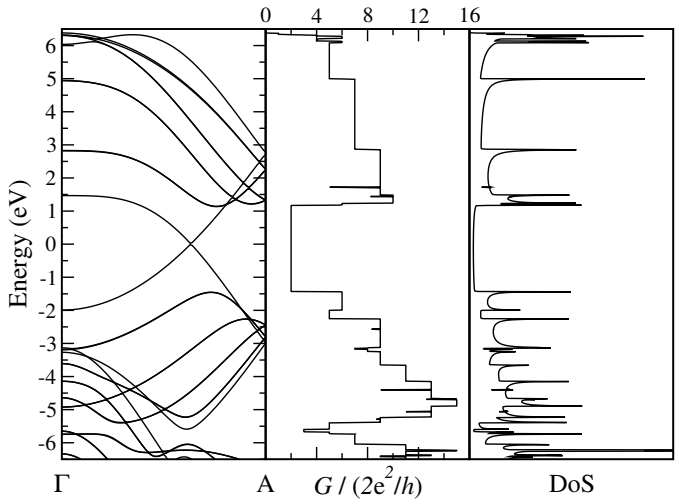


Figure 6: Wannier interpolated bandstructure, quantum conductance and density of states of (5,5) carbon nanotube. Note that the Fermi level has been shifted by 1.06eV with respect to Fig. 5.

```
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 infinite system 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.

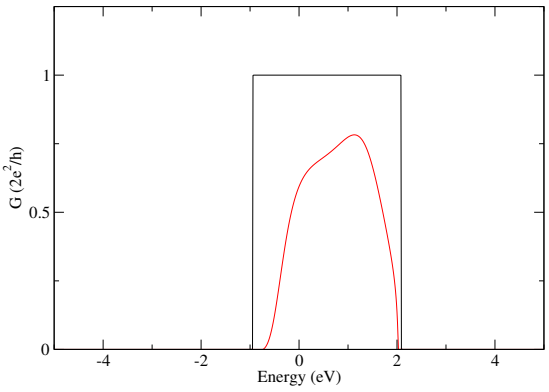


Figure 7: Quantum conductance of periodic Sodium chain (black) compared to that of the defected Sodium chain (red).

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.

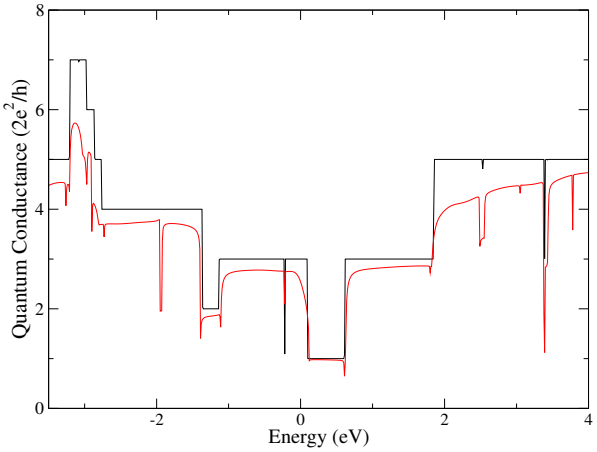


Figure 8: Quantum conductance of infinite pristine nanotube (black) compared to that of the infinite nanotube with the substitutional silicon defect (red).

Further ideas

- Set `write_hr = 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.

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 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: 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)
   `mpirun -np 8 postw90.x Fe` (example of parallel execution with 8 MPI processes)

[2]Please note the following counterintuitive 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
fermi_energy = [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
fermi_energy = [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 conductivity 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 restricted inside 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 $LaVO_3$.
   `pw.x < LaVO3.scf > scf.out`
2. Run PWSCF to obtain the Bloch states on a uniform k-point grid.
   `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. Notice how the t_{2g}-bands are entangled with other bands at A and the Wannier-interpolated band structure deviates from the Bloch bands only in a small region around that k-point. It is important to keep in mind that all symmetry equivalent k-points within the first Brillouin zone must be written explicitly in the list of sphere centers. For instance, the A point in the simple tetragonal lattice of this example is non-degenerate, while the X point has degeneracy two, hence one must specify both $(1/2, 0, 0)$ and $(0, 1/2, 0)$ (see the $SrMnO_3$ example here below).

Further ideas

- Try to obtain the Wannier functions using the standard disentanglement procedure (without spheres, `dis_spheres_num = 0`). You will notice that the Wannier-interpolated band structure

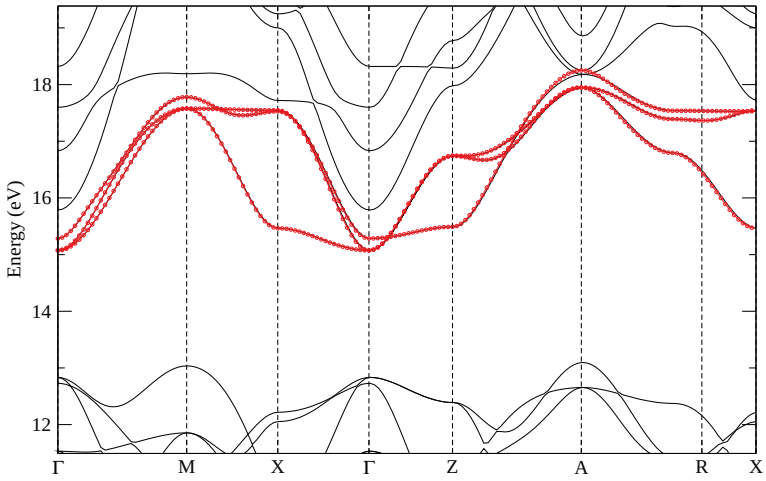


Figure 9: Band structure of epitaxially-strained (tetragonal) $LaVO_3$. Black: Bloch bands; red circles: Wannier-interpolated band structure. The disentanglement was performed only for k-points within a sphere of radius 0.2 $Å^{-1}$ centered in A.

now shows deviations also in regions of k-space far away from A, where disentanglement is actually not necessary. If you disable the disentanglement completely, instead, the Wannierisation procedure does not converge.

- In order to illustrate all possible cases, it is instructive to apply this method to $SrMnO_3$, where the t_{2g} bands are entangled with the above-lying e_g bands, and also with the deeper O-2p states. In the $SrMnO_3$ subfolder, you can find input files for building three different sets of Wannier functions: only t_{2g} states, only e_g states, or all V-3d-derived states ($t_{2g} + e_g$). In each case one needs to specify different disentanglement spheres, according to which region(s) in k-space show entanglement of the targeted bands. Also the index `dis_sphere_first_wan` needs to be adapted to the new disentanglement window, which here contains also states below the lowest-lying Wannier function (at variance with the $LaVO_3$ case).

21: Gallium Arsenide – Symmetry-adapted Wannier functions

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

- Outline: *Obtain symmetry-adapted Wannier functions out of four valence bands of GaAs. For the theoretical background of the symmetry-adapted Wannier functions, see R. Sakuma, Phys. Rev. B* **87***, 235109 (2013).*
- Directory: `examples/example21/atom_centered_As_sp/`
  `examples/example21/atom_centered_Ga_p/`
  `examples/example21/atom_centered_Ga_s/`
  `examples/example21/atom_centered_Ga_sp/`
  `examples/example21/bond_centered/`
- Input Files
  - `GaAs.scf` *The* PWSCF *input file for ground state calculation*

  - GaAs.nscf *The* PWSCF *input file to obtain Bloch states on a uniform grid*
  - GaAs.pw2wan *The input file for* pw2wannier90
  - GaAs.win *The* wannier90 *input file*

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

Each directory creates different kind of symmetry-adapted Wannier function. See more detail in examples/example21/README.

22: Copper – Symmetry-adapted Wannier functions

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

- Outline: *Obtain symmetry-adapted Wannier functions for Cu. By symmetry-adapted mode, for example, we can make atomic centered s-like Wannier function, which is not possible in the usual procedure to create maximally localized Wannier functions. For the theoretical background of the symmetry-adapted Wannier functions, see R. Sakuma, Phys. Rev. B* **87***, 235109 (2013).*
- Directory: examples/example22/s_at_0.00/
  examples/example22/s_at_0.25/
  examples/example22/s_at_0.50/
- Input Files
  - Cu.scf *The* PWSCF *input file for ground state calculation*
  - Cu.nscf *The* PWSCF *input file to obtain Bloch states on a uniform grid*
  - Cu.pw2wan *The input file for* pw2wannier90
  - Cu.sym *Used only in* examples/example22/s_at_0.25/. pw2wannier90 *reads this file when* ‘‘read_sym = .true.’’ *in* Cu.pw2wan. *By default,* ‘‘read_sym = .false.’’ and Cu.sym is the output of pw2wannier90*, in which the symmetry operations employed in the calculation are written for reference.*
  - Cu.win *The* wannier90 *input file*

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

Each directory creates *s*-like symmetry-adapted Wannier function centered at different position on top of atomic centered *d*-like Wannier functions. See more detail in `examples/example22/README`.

23: Silicon – G_0W_0 bands structure interpolation

Note: This example requires a recent version of the `ypp` post-processing code of `yambo`.

- Outline: *Interpolate the bands structure of silicon obtained from many-body perturbation theory at the G_0W_0 level. Using the* `yambo` *code, the quasi-particle corrections (QP) are summed to Kohn-Sham eigenvalues, while the wavefunctions remain the same.*
- Directory: `examples/example23/`
- Input Files
  - `silicon.scf` *The* PWSCF *input file for the ground state calculation*
  - `silicon.nscf` *The* PWSCF *input file to obtain Bloch states on a uniform grid*
  - `silicon.gw.nscf` *The* PWSCF *input file to obtain Bloch states on a reduced grid with many empty bands*
  - `silicon.pw2wan` *The input file for* `pw2wannier90`
  - `silicon.win` *The* `wannier90` *input file*
  - `silicon.gw.win` *The* `wannier90` *input file* (for the G_0W_0 step)
  - `yambo.in` *The* `yambo` *input file*
  - `ypp.in` *The* `ypp` *input file*

1. Copy the input files from the `INPUT directory` into a working directory (e.g. `WORK`)
2. Run PWSCF to obtain the ground state charge of silicon
   `pw.x < silicon.scf > scf.out`

3. Run PWSCF to obtain the Bloch states reduced grid. We use a 8x8x8 with many bands (many empty bands are needed to perform a G_0W_0 with `yambo`)
   `pw.x < silicon.gw.nscf > nscf.gw.out`

4. Use the `k_mapper.py` utility to find the indexes of a 4x4x4 uniform grid into the 8x8x8 reduced grid
   `./k_mapper.py 4 4 4 "../examples/example23/WORK/nscf.gw.out"`
   Use the output to complete the `yambo.in` input file (you also need to specify the on how many bands you want to compute the QP corrections, here you can use all the bands from 1 to 14). Then, you should have obtained something like:
   ```
   1| 1| 1|14|
   3| 3| 1|14|
   5| 5| 1|14|
   13| 13| 1|14|
   ...
   ```

5. Enter the `si.save` directory and run `p2y`. A `SAVE` folder is created, you can move it up in the `/WORK/` directory.

6. Run a G_0W_0 calculation from the `/WORK/` directory (remember, we are using a 8x8x8 grid but computing QP corrections only on a 4x4x4 grid)
   `yambo`

7. Run PWSCF to obtain the Bloch states on a uniform k-point grid
   `pw.x < silicon.nscf > nscf.out`

8. Run `wannier90` to generate a list of the required overlaps (written into the `silicon.nnkp` file).
   `wannier90.x -pp silicon`

9. Run `pw2wannier90` to compute the overlap between Bloch states, the projections for the starting guess (written in the `silicon.mmn` and `silicon.amn` respectively).
   `pw2wannier90.x < silicon.pw2wan > pw2wan.out`

10. Run `wannier90` to compute the MLWFs.
    `wannier90.x silicon`
    At this point, you should have obtained the interpolated valence bands for silicon at the DFT level.

11. Run a `ypp` calculation (just type `ypp`)
    You should obtain a file `silicon.gw.unsorted.eig` which contains the QP corrections on a uniform 4x4x4 grid.

12. Run the gw2wannier90.py script to reorder, align and correct all matrices and files using the QP corrections
    `../../../utility/gw2wannier90.py silicon mmn amn`

13. Run `wannier90` to compute the MLWFs.
    `wannier90.x silicon.gw`
    At this point, you should have obtained the interpolated valence bands for silicon at the G_0W_0 level.

After you completed the tutorial for the valence bands only, you can repeat the final steps to interpolate also some conduction bands using disentanglement (the code is already present as comments in the input files).

24: Tellurium – gyrotropic effects

- Outline: *Calculate the gyrotropic effects in trigonal right-handed Te* Similar to the calculations of [14]
- Directory: `examples/example24/`
- Input files
  - `Te.scf` *The* PWSCF *input file for ground state calculation*
  - `Te.nscf` *The* PWSCF *input file to obtain Bloch states on a uniform grid*
  - `Te.pw2wan` *The input file for* `pw2wannier90`
  - `Te.win` *The* `wannier90` *input file*

To make things easy, the example treats Te without spin-orbit

1. Run PWSCF to obtain the ground state of tellurium
   `pw.x < Te.scf > scf.out`
2. Run PWSCF to obtain the Bloch states on a uniform `3x3x4` k-point grid
   `pw.x < Te.nscf > nscf.out`
3. Run `wannier90` to generate a list of the required overlaps (written into the `Te.nnkp` file).
   `wannier90.x -pp Te`
4. Run `pw2wannier90` to compute:
   - The overlaps $\langle u_{n\mathbf{k}} | u_{m\mathbf{k}+\mathbf{b}} \rangle$ (written in the `Te.mmn` file)
   - The projections for the starting guess (written in the `Te.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 `Te.uHu` file)
   - The spin matrix elements $\langle \psi_{n\mathbf{k}} | \sigma_i | \psi_{m\mathbf{k}} \rangle$ (would be written in the `Te.spn` file, but only if spin-orbit is included, which is not the case for the present example)

   `pw2wannier90.x < Te.pw2wan > pw2wan.out`
5. Run `wannier90` to compute the MLWFs.
   `wannier90.x Te`
6. Add the following lines to the `wannier90.win` file:
   ```
   gyrotropic=true
   gyrotropic_task=-C-dos-D0-Dw-K
   fermi_energy_step=0.0025
   fermi_energy_min=5.8
   fermi_energy_max=6.2
   gyrotropic_freq_step=0.0025
   ```

```
gyrotropic_freq_min=0.0
gyrotropic_freq_max=0.1
gyrotropic_smr_fixed_en_width=0.01
gyrotropic_smr_max_arg=5
gyrotropic_degen_thresh=0.001
gyrotropic_box_b1=0.2 0.0 0.0
gyrotropic_box_b2=0.0 0.2 0.0
gyrotropic_box_b3=0.0 0.0 0.2
gyrotropic_box_center=0.33333 0.33333 0.5
gyrotropic_kmesh=50 50 50
```

7. Run `postw90`
   to compute the gyrotropic properties: tensors D, $\widetilde{D}$, K, C (See the User Guide):.
   `postw90.x Te` (serial execution)
   `mpirun -np 8 postw90.x Te` (example of parallel execution with 8 MPI processes)

   The integration in the k-space is limited to a small area around the H point. Thus it is valid only for Fermi levels near the band gap. And one needs to multiply the results by 2, to account for the H' point. To integrate over the entire Brillouin zone, one needs to remove the `gyrotropic_box_`... parameters

8. Now change the above lines to

```
gyrotropic=true
gyrotropic_task=-NOA
fermi_energy=5.95
gyrotropic_freq_step=0.0025
gyrotropic_freq_min=0.0
gyrotropic_freq_max=0.3
gyrotropic_smr_fixed_en_width=0.01
gyrotropic_smr_max_arg=5
gyrotropic_band_list=4-9
gyrotropic_kmesh=50 50 50
```

   and compute the interband natural optical activity

   `postw90.x Te` (serial execution)
   `mpirun -np 8 postw90.x Te` (example of parallel execution with 8 MPI processes)

25: Gallium Arsenide – Nonlinear shift current

- Outline: *Calculate the nonlinear shift current of inversion asymmetric fcc Gallium Arsenide. In preparation for this example it may be useful to read Ref. [15]*
- Directory: `examples/example25/`
- Input files:
  - `GaAs.scf` *The* `PWSCF` *input file for ground state calculation*

- `GaAs.nscf` *The* `PWSCF` *input file to obtain Bloch states on a uniform grid*
- `GaAs.pw2wan` *The input file for* `pw2wannier90`
- `GaAs.win` *The* `wannier90` *and* `postw90` *input file*

1. Run `PWSCF` to obtain the ground state of Gallium Arsenide
   `pw.x < GaAs.scf > scf.out`
2. Run `PWSCF` to obtain the ground state of Gallium Arsenide
   `pw.x < GaAs.nscf > nscf.out`
3. Run `Wannier90` to generate a list of the required overlaps (written into the `GaAs.nnkp` file)
   `wannier90.x -pp GaAs`
4. Run `pw2wannier90` to compute:
   - The overlaps $\langle u_{n\mathbf{k}} | u_{n\mathbf{k}+\mathbf{b}} \rangle$ between spinor Bloch states (written in the `GaAs.mmn` file)
   - The projections for the starting guess (written in the `GaAs.amn` file)

   `pw2wannier90.x < GaAs.pw2wan > pw2wan.out`
5. Run `wannier90` to compute MLWFs
   `wannier90.x GaAs`
6. Run `postw90` to compute nonlinear shift current
   `postw90.x GaAs` (serial execution)
   `mpirun -np 8 postw90.x GaAs` (example of parallel execution with 8 MPI processes)

Shift current σ^{abc}

The shift current tensor of GaAs has only one independent component that is finite, namely σ^{xyz}. For its computation, set

```
berry = true
berry_task = sc
```

Like the linear optical conductivity, the shift current is a frequency-dependent quantity. The frequency window and step is controlled by `kubo_freq_min`, `kubo_freq_max` and `kubo_freq_step`, as explained in the users guide.

The shift current requires an integral over the Brillouin zone. The interpolated k-mesh is controlled by `berry_kmesh`, which has been set to

```
berry_kmesh = 100 100 100
```

We also need to input the value of the Fermi level in eV:

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

Due to the sum over intermediate states involved in the calculation of the shift current, one needs to consider a small broadening parameter to avoid numerical problems due to possible degeneracies (see parameter η in Eq. (36) of Ref. [15] and related discussion). This parameter is controlled by `sc_eta`.

It is normally found that values between 0.01 eV and 0.1 eV yield an stable spectrum. The default value is set to 0.04 eV.

Finally, `sc_phase_conv` controls the phase convention used for the Bloch sums. `sc_phase_conv=1` uses the so-called tight-binding convention, whereby the Wannier centres are included into the phase, while `sc_phase_conv=2` leaves the Wannier centres out of the phase. These two possible conventions are explained in Ref. [16]. Note that the overall shift-current spectrum does not depend on the chosen convention, but the individual terms that compose it do.

On output, the program generates a set of 18 files named `SEED-sc_***.dat`, which correspond to the different tensor components of the shift current (note that the 9 remaining components until totaling $3 \times 3 \times 3 = 27$ can be obtained from the 18 outputed by taking into account that σ^{abc} is symmetric under $b \leftrightarrow c$ index exchange). For plotting the only finite shift-current component of GaAs σ^{xyz} (units of $\mathrm{A/V^2}$) as in the upper panel of Fig. 3 in Ref. [15],

```
myshell> gnuplot
gnuplot> plot 'GaAs-sc_xyz.dat' u 1:2 w l
```

26: Gallium Arsenide – Selective localization and constrained centres

- Outline: *Application of the selectively localised Wannier function (SLWF) method to gallium arsenide (GaAs), following the example in Ref. [17], which is essential reading for this tutorial example.*
- Directory: `examples/example26/`
- Input files:
  - `GaAs.scf` *The* `PWSCF` *input file for ground state calculation*
  - `GaAs.nscf` *The* `PWSCF` *input file to obtain Bloch states on a uniform grid*
  - `GaAs.pw2wan` *The input file for* `pw2wannier90`
  - `GaAs.win` *The* `wannier90` *and* `postw90` *input file*

1. Run `PWSCF` to obtain the ground state of Gallium Arsenide
   `pw.x < GaAs.scf > scf.out`
2. Run `PWSCF` to obtain the ground state of Gallium Arsenide
   `pw.x < GaAs.nscf > nscf.out`
3. Run `Wannier90` to generate a list of the required overlaps (written into the `GaAs.nnkp` file)
   `wannier90.x -pp GaAs`
4. Run `pw2wannier90` to compute:
   - The overlaps $\langle u_{n\mathbf{k}} | u_{n\mathbf{k}+\mathbf{b}} \rangle$ between Bloch states (written in the `GaAs.mmn` file)
   - The projections for the starting guess (written in the `GaAs.amn` file)

   `pw2wannier90.x < GaAs.pw2wan > pw2wan.out`
5. Inspect the `.win` file.
   - Make sure you understand the new keywords corresponding to the selective localisation algorithm.

– Run `wannier90` to compute the SLWFs, in this case using one objective Wannier function.

```
wannier90.x GaAs
```

To constrain the centre of the SLWF you need to add `slwf_constrain = true` and `slwf_lambda = 1` to the input file and uncomment the `slwf_centres` block. This will add a penalty functional to the total spread, which will try to constrain the centre of the SLWF to be on the As atom (as explained in Ref. [17], particularly from Eq. 24 to Eq. 35).

Look at the value of the penalty functional, is this what you would expect at convergence? Does the chosen value of the Lagrange multiplier `slwf_lambda` give a SLWF function centred on the As atom?

Alternatively, you can modify the `slwf_centres` block to constrain the centre of the SLWF to be on the Ga atom. Do you need a different value of `slwf_lambda` in this case to converge? Take a look at the result in Vesta and explain what you see. Do these functions transform like the identity under the action of the T_d group?

27: Silicon – Selected columns of density matrix algorithm for automated MLWFs

Note: This example requires a recent version of the `pw2wannier90.x` post-processing code of `Quantum ESPRESSO` (v6.4 or above).

- Outline: *For bulk crystalline Silicon, generate the A_{mn} matrices via the selected columns of density matrix (SCDM) algorithm and the corresponding MLWFs for 1) Valence bands 2) Valence bands and 4 low-lying conduction bands 3) Conduction bands only. To better understand the input files and the results of these calculations, it is crucial that the Reader has familiarized with the concepts and methods explained in Ref. [18]. More info on the keywords related to the SCDM method may be found in the user_ guide.*
- Directory: `examples/example27/`
- Input Files: `input_files`, and in the three subfolders `isolated, erfc` and `gaussian`. The `input_files` folder contains:
  - `si.scf` *The* PWSCF *input file for the ground state calculation*
  - `si_4bands.nscf` *The* PWSCF *input file to obtain Bloch states on a uniform grid for 4 bands.*
  - `si_12bands.nscf` *The* PWSCF *input file to obtain Bloch states on a uniform grid for 12 bands.*
- Whereas the three subfolders `isolated, erfc` and `gaussian` contain the `si.win wannier90` input files and `si.pw2wan pw2wannier90` input files each corresponding to one of the scenarios listed in the outline.

1 Valence bands: In this case we will compute 4 localized WFs corresponding to the 4 valence bands of Silicon. These 4 bands constitute a manifold that is separated in energy from other bands. In this case the columns of the density matrix are already localized in real space and no extra parameter is required.

1. Copy the input files `si.scf` and `si_4bands.nscf` from the `input_files` directory into the `isolated` folder
2. Run PWSCF to obtain the ground state charge of bulk Silicon.
   `pw.x < si.scf > scf.out`
3. Run PWSCF to obtain the Bloch states on a uniform k-point grid of 4x4x4 for 4 bands.
   `pw.x < si_4bands.nscf > nscf.out`
4. Inspect the `si.win` input file and make sure that the `auto_projections` flag is set to `.true.`. Also, make sure that no projections block is present.
5. Run `wannier90` to generate a list of the required overlaps and also info on the SCDM method (written into the `si.nnkp` file).
   `wannier90.x -pp si`
6. Inspect the `si.nnkp` file and make sure you find the `auto_projections` block and that no projections have been written in the `projections` block.
7. Inspect the `.pw2wan` input file. You will find two new keywords, i.e. `scdm_proj` and `scdm_entanglement`. The former, will instruct `pw2wannier90.x` to use the SCDM method when generating the A_{mn} matrix. The latter, defines which formula to adopt for the function $f(\varepsilon_{n\mathbf{k}})$ (see [18] and point below).
8. Run `pw2wannier90` to compute the overlap between Bloch states and the projections via the SCDM method (written in the `si.mmn` and `si.amn` respectively).
   `pw2wannier90.x < si.pw2wan > pw2wan.out`
9. Run `wannier90` to compute the MLWFs.
   `wannier90.x si`
   At this point, you should have obtained 4 Wannier functions and the interpolated valence bands for Silicon. Inspect the output file `si.wout`. In particular, look at the geometric centres of each WF, do they lie at the centre of the Si-Si bond as for the MLWFs computed from user-defined initial s-like projections (see Example11)? Plot these WFs using Vesta. Do they show the σ character one would expect from chemical arguments?

2 Valence bands + conduction bands: In this case we will compute 8 localized WFs corresponding to the 4 valence bands and 4 low-lying conduction bands. Here, we don't have a separate manifold, since the conduction bands are entangled with other high-energy bands and the columns of the density matrix are not exponentially localized by construction. A modified density matrix is required in this case[18], and it is defined as:

$$P(\mathbf{r},\mathbf{r}') = \sum_{n,\mathbf{k}} \psi_{n\mathbf{k}}(\mathbf{r}) f(\varepsilon_{n,\mathbf{k}}) \psi^*_{n\mathbf{k}}(\mathbf{r}'),$$

where $\psi_{n\mathbf{k}}$ and $\varepsilon_{n,\mathbf{k}}$ are the energy eigestates and eigenvalues from the first-principle calculation respectively. The function $f(\varepsilon_{n,\mathbf{k}})$ contains two free parameters μ and σ and is defined as a complementary error function:

$$f(\varepsilon_{n,\mathbf{k}}) = \frac{1}{2}\mathrm{erfc}\left(\frac{\varepsilon_{n,\mathbf{k}} - \mu}{\sigma}\right).$$

1. Copy the input files `si.scf` and `si_12bands.nscf` from the `input_files` folder into the `erfc` folder
2. Run PWSCF to obtain the ground state charge of bulk Silicon.
   `pw.x < si.scf > scf.out`

3. Run PWSCF to obtain the Bloch states on a uniform k-point grid of 4x4x4 for 12 bands this time.
   `pw.x < si_12bands.nscf > nscf.out`
4. Inspect the `si.win` input file and make sure that the `auto_projections` flag is set to `.true.`. Also, make sure that no projection block is present.
5. Run `wannier90` to generate a list of the required overlaps and also info on the SCDM method (written into the `si.nnkp` file).
   `wannier90.x -pp si`
6. Inspect the `si.nnkp` file and make sure you find the `auto_projections` block and that no projections have been written in the `projections` block.
7. Inspect the `.pw2wan` input file. You will find other two new keywords, i.e. `scdm_mu` and `scdm_sigma`. These are the values in eV of μ and σ in $f(\varepsilon_{n,\mathbf{k}})$, respectively.
8. Run `pw2wannier90` to compute the overlap between Bloch states and the projections via the SCDM method (written in the `si.mmn` and `si.amn` respectively).
   `pw2wannier90.x < si.pw2wan > pw2wan.out`
9. Run `wannier90` to compute the MLWFs.
   `wannier90.x si`
   At this point, you should have obtained 8 localized Wannier functions and the interpolated valence and conduction bands for Silicon. Again, compare the results for the geometric centres and the individual spreads with the ones from Example11. Is the final value of total spread bigger or smaller than the one from Example11? Look at the WFs with Vesta. Can you explain what you see? Where do the major lobes of the $sp3$-like WFs point in this case?

3 Conduction bands only: In this case we will compute 4 localized WFs corresponding to the 4 low-lying conduction bands only. As for the previous point, we need to define a modified density matrix[18]. Since we are only interested in a subset of the conduction states, within a bounded energy region, a good choice for $f(\varepsilon_{n,\mathbf{k}})$ is:

$$f(\varepsilon_{n,\mathbf{k}}) = \exp\left(-\frac{(\varepsilon_{n,\mathbf{k}} - \mu)^2}{\sigma^2}\right).$$

1. Copy the input files `si.scf` and `si_12bands.nscf` from the `input_files` directory into the `gaussian` folder
2. Run PWSCF to obtain the ground state charge of bulk Silicon.
   `pw.x < si.scf > scf.out`
3. Run PWSCF to obtain the Bloch states on a uniform k-point grid of 4x4x4 for 12 bands this time.
   `pw.x < si_12bands.nscf > nscf.out`
4. Inspect the `si.win` input file and make sure that the `auto_projections` flag is set to `.true.`. Also, make sure that no projections block is present.
5. Run `wannier90` to generate a list of the required overlaps and also info on the SCDM method (written into the `si.nnkp` file).
   `wannier90.x -pp si`
6. Inspect the `si.nnkp` file and make sure you find the `auto_projections` block and that no projections have been written in the `projections` block.
7. Run `pw2wannier90` to compute the overlap between Bloch states, the projections for the starting guess via the SCDM method (written in the `si.mmn` and `si.amn` respectively).
   `pw2wannier90.x < si.pw2wan > pw2wan.out`

8. Run `wannier90` to compute the MLWFs.
   `wannier90.x si`
   At this point, you should have obtained 4 localized Wannier functions and the interpolated conduction bands for Silicon. From chemical intuition, we would expect these functions to be similar to anti-bonding orbitals of molecules with tetrahedral symmetry. Plot the WFs and check if this is confirmed.

28: Diamond – plotting of MLWFs using Gaussian cube format and VESTA

- Outline: *Obtain MLWFs for the valence bands of diamond and output them in Gaussian cube format*
- Directory: `examples/example28/` The input files for this examples are the same as the ones in example05
- 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. When the lattice vectors are non-orthogonal, not all the visualisation programs are capable to plot volumetric data in the Gaussian cube format. One program that can read volumetric data for these systems is VESTA. To instruct `wannier90` to output the MLWFs data in Gaussian cube format you need to add the following lines to the `.win` file

   ```
   wannier_plot           = .true.
   wannier_plot_supercell = 3
   wannier_plot_format    = cube
   wannier_plot_mode      = crystal
   wannier_plot_radius    = 2.5
   wannier_plot_scale     = 1.0
   ```

   Run `wannier90` to compute the MLWFs and output them in the Gaussian cube file.
   `wannier90.x diamond`

6. Plot the first MLWF with VESTA `vesta diamond_00001.cube`

Extra: Instead of using `wannier_plot_mode = crystal` try to use the molecule mode as `wannier_plot_mode = molecule` (see the user guide for the definition of this keyword). Add the following line to the `.win` file:

```
restart = plot
```

and re-run `wannier90`. Use VESTA to plot the resulting MLWFs, do you see any difference from the `crystal` mode case? Can you explain why? Try to change the size of the supercell from 3 to 5, do you expect the results to be different? (*Hint:* When using the Gaussian cube format the code outputs the WF on a grid that is smaller than the super unit-cell. The size of the grid is specified by `wannier_plot_scale` and `wannier_plot_radius`.)

29: Platinum – Spin Hall conductivity

- Outline: *Calculate spin Hall conductivity (SHC) and plot Berry curvature-like term of fcc Pt considering spin-orbit coupling. To gain a better understanding of this example, it is suggested to read Ref. [19] for a detailed description of the theory and Ch. 12.5 of the User Guide.*
- Directory: `examples/example29/`
- Input files
  - `Pt.scf` *The* PWSCF *input file for ground state calculation*
  - `Pt.nscf` *The* PWSCF *input file to obtain Bloch states on a uniform grid*
  - `Pt.pw2wan` *The input file for* `pw2wannier90`
  - `Pt.win` *The* `wannier90` *and* `postw90` *input file*

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

Spin Hall conductivity

The intrinsic spin Hall conductivity $\sigma_{\alpha\beta}^{\mathrm{spin}\gamma}$ is proportional to the BZ integral of the Berry curvature-like term. To evaluate the SHC using a $25 \times 25 \times 25$ k-point mesh, set the following lines in `Pt.win`,

```
berry = true
berry_task = shc
berry_kmesh = 25 25 25
```

When calculating SHC, adaptive smearing can be used by commenting the following two lines,

```
#kubo_adpt_smr = false
#kubo_smr_fixed_en_width = 1
```

Then set the Fermi energy ε_F to a specific value

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

or invoke Fermi energy scan by setting

```
fermi_energy_min = [insert here your lower range]
fermi_energy_max = [insert here your upper range]
fermi_energy_step = [insert here your step]
```

and re-run `postw90`. The SHC is written in the output file `Pt-shc-fermiscan.dat`. If only `fermi_energy` is set, the output file will contain SHC at this specific energy; if a list of Fermi energies are set, the output file will contain SHC calculated at each energy point in the list: we call this the "Fermi energy scan" of SHC.

To plot the Fermi energy scan of SHC $\sigma_{xy}^{\mathrm{spin}z}$ versus ε_F, issue

```
myshell> gnuplot
gnuplot> plot 'Pt-shc-fermiscan.dat' u 2:3 w lp
```

As a result of the strong and rapid variations of the Berry curvature-like term across the BZ, the SHC converges rather slowly with k-point sampling, and a $25\times25\times25$ kmesh does not yield a well-converged value.

- Increase the kmesh density by changing `berry_kmesh`.
- To accelerate the convergence, adaptively refine the kmesh around spikes in the Berry curvature-like term, by adding to `Pt.win` the lines

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

  This adds a $5\times5\times5$ fine mesh around those points where $|\Omega_{\alpha\beta}^{\mathrm{spin}\gamma}(\boldsymbol{k})|$ exceeds 100 `berry_curv_unit`. The percentage of points triggering adaptive refinement is reported in `Pt.wpout`.

Compare the converged SHC value with those obtained in Refs. [19] and [20].

Note some rough estimations of computation progress and time are reported in `Pt.wpout` (see the SHC part of the Solution Booklet). These may be helpful if the computation time is very long.

Notes

- Since the Kubo formula of SHC involves unoccupied bands, we need to include some unoccupied bands and construct more MLWF. Thus the following parameters should be increased accordingly:

  ```
  dis_froz_max
  dis_win_max
  projections
  ```

- Normally we calculate the SHC $\sigma_{xy}^{\mathrm{spin}z}$, i.e. $\alpha = x, \beta = y, \gamma = z$. To calculate other components, the following parameters can be set as `1, 2, 3`

  ```
  shc_alpha = [insert here the α direction]
  shc_beta = [insert here the β direction]
  shc_gamma = [insert here the γ direction]
  ```

  with `1, 2, 3` standing for `x, y, z` respectively.

Berry curvature-like term plots

The band-projected Berry curvature-like term $\Omega_{n,\alpha\beta}^{\mathrm{spin}\gamma}(\boldsymbol{k})$ is defined in Eq. (12.22) of the User Guide. The following lines in `Pt.win` are used to calculate the energy bands colored by the band-projected Berry curvature-like term $\Omega_{n,\alpha\beta}^{\mathrm{spin}\gamma}(\boldsymbol{k})$ (in Å^2), as well as the k-resolved Berry curvature-like term $\Omega_{\alpha\beta}^{\mathrm{spin}\gamma}(\boldsymbol{k})$ along high-symmetry lines in k-space, i.e. the `kpath` plot. First comment the line `berry = true` and then set

```
kpath = true
kpath_task = bands+shc
kpath_bands_colour = shc
kpath_num_points = 400
kubo_adpt_smr = false
kubo_smr_fixed_en_width = 1
fermi_energy = [insert your value here]
berry_curv_unit = ang2
```

After executing `postw90`, four files are generated: `Pt-bands.dat`, `Pt-path.kpt`, `Pt-shc.dat` and `Pt-bands+shc.py`. Then plot the band-projected Berry curvature-like term $\Omega_{n,\alpha\beta}^{\mathrm{spin}\gamma}(\boldsymbol{k})$ using the script generated at runtime,

```
myshell> python Pt-bands+shc.py
```

and compare with Fig. 2 of Ref. [19]. Note a large fixed smearing of 1 eV is used to recover the result in Ref. [19]. You can adjust the `kubo_smr_fixed_en_width` as you like to draw a visually appealing figure. A `kpath` plot of 0.05 eV smearing is shown in the Solution Booklet.

Besides, you can set `kpath_task = shc` to only draw k-resolved term $\Omega_{\alpha\beta}^{\mathrm{spin}\gamma}(\boldsymbol{k})$ (the lower panel of the figure), or set `kpath_task = bands` and `kpath_bands_colour = shc` to only draw energy bands colored by the band-projected term $\Omega_{n,\alpha\beta}^{\mathrm{spin}\gamma}(\boldsymbol{k})$ (the upper panel of the figure).

Similar to that of AHC, we can get a heatmap plot of the k-resolved Berry curvature-like term $\Omega_{\alpha\beta}^{\mathrm{spin}\gamma}(\boldsymbol{k})$, i.e. the `kslice` plot. To move forward, set `kpath = false` and uncomment the following lines in `Pt.win`,

```
kslice = true
kslice_task = shc+fermi_lines
kslice_corner = 0.0 0.0 0.0
kslice_b1 = 1.0 0.0 0.0
kslice_b2 = 0.3535533905932738 1.0606601717798214 0.00
kslice_2dkmesh = 200 200
```

Note the `kslice_b2` is actually $(\frac{\sqrt{2}}{4}, \frac{3\sqrt{2}}{4}, 0.0)$ which leads to a square slice in the BZ, making it easier to plot in the generated `python` script. Re-run `postw90`, and issue

```
myshell> python Pt-kslice-shc+fermi_lines.py
```

Compare the generated figure with Fig. 3 in Ref. [19], or the Solution Booklet.

Notes

- Adaptive smearing depends on a uniform kmesh, so when running `kpath` and `kslice` plots adaptive smearing should not be used. A fixed smearing is needed to avoid near zero number in the denominator of the Kubo formula, Eq. (12.22) in the User Guide. To add a fixed smearing of 0.05 eV, add the following keywords in the `Pt.win`,

  ```
  kubo_adpt_smr = .false.
  kubo_smr_fixed_en_width = 0.05
  ```

Input parameters for SHC

Finally, we provide a complete list of input parameters that can be used to control the SHC calculation, including the calculation of alternating current (ac) SHC which will be introduced in the next tutorial.

- general controls for SHC

  ```
  shc_freq_scan, shc_alpha, shc_beta, shc_gamma,
  kubo_eigval_max, exclude_bands, berry_curv_unit
  ```

- kmesh

  ```
  berry_task, berry_kmesh,
  berry_curv_adpt_kmesh, berry_curv_adpt_kmesh_thresh
  ```

- ac SHC

  ```
  kubo_freq_min, kubo_freq_max, kubo_freq_step,
  shc_bandshift, shc_bandshift_firstband, shc_bandshift_energyshift,
  scissors_shift, num_valence_bands
  ```

- smearing

  ```
  [kubo_]adpt_smr, [kubo_]adpt_smr_fac, [kubo_]adpt_smr_max,
  [kubo_]smr_fixed_en_width
  ```

- Fermi energy

  ```
  fermi_energy, fermi_energy_min, fermi_energy_max, fermi_energy_step
  ```

- kpath

  ```
  kpath, kpath_task, kpath_num_points, kpath_bands_colour
  ```

- kslice

  ```
  kslice, kslice_task, kslice_corner, kslice_b1, kslice_b2, kslice_2dkmesh,
  kslice_fermi_level, kslice_fermi_lines_colour
  ```

Their meanings and usages can be found in Ch. 11.5 of the User Guide.

30: Gallium Arsenide – Frequency-dependent spin Hall conductivity

- Outline: *Calculate the alternating current (ac) spin Hall conductivity of gallium arsenide considering spin-orbit coupling. To gain a better understanding of this example, it is suggested to read Ref. [19] for a detailed description of the theory and Ch. 12.5 of the User Guide.*
- Directory: `examples/example30/`
- Input files
  - `GaAs.scf` *The* PWSCF *input file for ground state calculation*
  - `GaAs.nscf` *The* PWSCF *input file to obtain Bloch states on a uniform grid*
  - `GaAs.pw2wan` *The input file for* `pw2wannier90`
  - `GaAs.win` *The* `wannier90` *and* `postw90` *input file*

1. Run PWSCF to obtain the ground state of gallium arsenide
   `pw.x < GaAs.scf > scf.out`

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

ac spin Hall conductivity

The spin Hall conductivity is also dependent on the frequency ω in the Eq. (12.22) of the User Guide. The direct current (dc) SHC calculated in the previous example corresponds to $\sigma_{\alpha\beta}^{\mathrm{spin}\gamma}$ in the limit $\omega \to 0$ and it is a real number. At finite frequency $\sigma_{\alpha\beta}^{\mathrm{spin}\gamma}$ acquires an imaginary part.

To compute the ac spin Hall conductivity for $\hbar\omega$ up to 8 eV, add the lines

```
shc_freq_scan = true
kubo_freq_min = 0.0
kubo_freq_max = 8.0
kubo_freq_step = 0.01
```

and re-run `postw90`. The file `GaAs-shc-freqscan.dat` contains the calculated ac SHC. Reasonably converged spectra can be obtained with a $250 \times 250 \times 250$ k-point mesh. To plot the ac SHC, issue the following commands

```
myshell> gnuplot
gnuplot> plot 'GaAs-shc-freqscan.dat' u 2:3 w l title 'Re', 'GaAs-shc-freqscan.dat'
u 2:4 w l title 'Im'
```

and then compare the result with Fig. 4 in Ref. [19] or the Solution Booklet.

Notes

- When calculating ac SHC, adaptive smearing can be used by add the following keywords in the `GaAs.win`,

```
kubo_adpt_smr = true
kubo_adpt_smr_fac = [insert here your smearing factor]
kubo_adpt_smr_max = [insert here your maximum smearing]
```

- Adaptive kmesh refinement is not implemented for ac SHC calculation.
- The first 10 semi-core states are excluded from the calculation by using the following keywords

  ```
  exclude_bands = 1-10
  ```

  and in the case of GaAs disentanglement is not adopted so

  ```
  num_bands = 16
  num_wann = 16
  ```

- Since the band gap is often under estimated by LDA/GGA calculations, a scissors shift is applied to recover the experimental band gap by using the following keywords

  ```
  shc_bandshift = true
  shc_bandshift_firstband = 9
  shc_bandshift_energyshift = 1.117
  ```

  or by

  ```
  num_valence_bands = 8
  scissors_shift = 1.117
  ```

31: Platinum – Selected columns of density matrix algorithm for spinor wavefunctions

Note: This example requires a recent version of the `pw2wannier90.x` post-processing code of `Quantum ESPRESSO` (v6.3 or above).

- Outline: *For bulk crystalline platinum with spin-orbit coupling, generate the A_{mn} matrices via the selected columns of density matrix (SCDM) algorithm and the corresponding spinor-MLWFs. To better understand the input files and the results of these calculations, it is crucial that the Reader has familiarized with the concepts and methods explained in Ref. [18]. More info on the keywords related to the SCDM method may be found in the user_guide.*

  This example focuses on the use of the SCDM method for spin-noncollinear systems. For the overview of the use of SCDM method to spinless systems, please refer to example27.

- Directory: `examples/example31/`

  The input files for this examples are similar to the ones in example 29, except that a coarser k-point grid is used and that the keywords related to `postw90.x` are removed.

- Input Files:
  - `Pt.scf` *The* PWSCF *input file for the ground state calculation*
  - `Pt.nscf` *The* PWSCF *input file to obtain Bloch states on a uniform grid*
  - `Pt.pw2wan` *The input file for* `pw2wannier90` *with keywords related to the SCDM method*
  - `Pt.win` *The* `wannier90` *input file*

We will compute 18 localized WFs. Since the band structure of platinum is metallic, the low-lying bands are entangled with other high-energy bands, and the columns of the density matrix are not exponentially localized by construction. Thus, we use a modified density matrix [18], with the function $f(\varepsilon_{n,\mathbf{k}})$ defined as a complementary error function. Refer to example 27 for the definition of the modified density matrix and the functional form of $f(\varepsilon_{n,\mathbf{k}})$.

1. Run PWSCF to obtain the ground state of platinum
   `pw.x < Pt.scf > scf.out`
2. Run PWSCF to obtain the Bloch states on a uniform $7 \times 7 \times 7$ k-point grid
   `pw.x < Pt.nscf > nscf.out`
3. Inspect the `Pt.win` input file and make sure that the `auto_projections` flag is set to `.true.`. Also, make sure that no projection block is present.
4. Run `wannier90` to generate a list of the required overlaps (written into the `Pt.nnkp` file)
   `wannier90.x -pp Pt`
5. Inspect the `Pt.nnkp` file and make sure you find the `auto_projections` block and that no projections have been written in the `projections` block.
6. Inspect the `Pt.pw2wan` input file. You will find four SCDM-related keywords: `scdm_proj`, `scdm_entanglement`, `scdm_mu` and `scdm_sigma`. In particular, the keyword `scdm_proj` will instruct `pw2wannier90.x` to use the SCDM method when generating the A_{mn} matrix. The remaining three keywords defines the formula and parameters to define the function $f(\varepsilon_{n\mathbf{k}})$ (see Ref. [18] and example 27).
7. Run `pw2wannier90` to compute the overlap between Bloch states and the projections via the SCDM method (written in the `Pt.mmn` and `Pt.amn` respectively).
   `pw2wannier90.x < Pt.pw2wan > pw2wan.out`
8. Inspect the `pw2wan.out` output file. Compared to the spinless case, you will find the following two additional lines.

   ```
   Number of pivot points with spin up  :     9
   Number of pivot points with spin down:     9
   ```

   These lines give information on the pivots obtained by the QR decomposition with column pivoting (QRCP) in the SCDM algorithm. Each pivot determines a point in the real-space grid and a spin state. The basis of the spin state is determined by the basis used in the electronic structure code. In PWSCF, the basis states are spin up and down states along the Cartesian z-axis.
9. Run `wannier90` to compute the MLWFs
   `wannier90.x Pt`

32: Tungsten — SCDM parameters from projectability

- Outline: *Compute the Wannier interpolated band structure of tungsten (W) using the SCDM method to calculate the initial guess (see Example 27 for more details). The free parameters in*

the SCDM method, i.e., μ and σ, are obtained by fitting a complementary error function to the projectabilities. The number of MLWFs is given by the number of pseudo-atomic orbitals (PAOs) in the pseudopotential, 21 *in this case. All the steps shown in this example have been automated in the AiiDA[21] workflow that can be downloaded from the MaterialsCloud website[22].*

- Directory: `examples/example31/`
- Input files
  - `W.scf` *The* PWSCF *input file for ground state calculation*
  - `W.nscf` *The* PWSCF *input file to obtain Bloch states on a uniform grid*
  - `W.pw2wan` *The input file for* `pw2wannier90`
  - `W.proj` *The input file for* `projwfc`
  - `generate_weights.sh` *The bash script to extract the projectabilities from the output of* `projwfc`
  - `W.win` *The* `wannier90` *input file*

1. Run PWSCF to obtain the ground state of tungsten
   `pw.x -in W.scf > scf.out`
2. Run PWSCF to obtain the Bloch states on a $10 \times 10 \times 10$ uniform k-point grid
   `pw.x -in W.nscf > nscf.out`
3. Run `wannier90` to generate a list of the required overlaps (written into the `W.nnkp` file)
   `wannier90.x -pp W`
4. Run `projwfc` to compute the projectabilities of the Bloch states onto the Bloch sums obtained from the PAOs in the pseudopotential
   `projwfc.x -in W.proj > proj.out`
5. Run `generate_weights` to extract the projectabilitites from `proj.out` in a format suitable to be read by Xmgrace or gnuplot
   `./generate_weights.sh`
6. Plot the projectabilities and fit the data with the complementary error function

$$f(\epsilon;\mu,\sigma) = \frac{1}{2}\mathrm{erfc}(-\frac{\mu-\epsilon}{\sigma}).$$

   We are going Xmgrace to plot the projectabilities and perform the fitting. Open Xmgrace
   `xmgrace`

   To Import the `p_vs_e.dat` file, click on `Data` from the top bar and then `Import -> ASCII...`. At this point a new window `Grace: Read sets` should pop up. Select `p_vs_e.dat` in the `Files` section, click `Ok` at the bottom and close the window. You should now be able to see a quite noisy function that is bounded between 1 and 0. You can modify the appearence of the plot by clicking on `Plot` in the top bar and then `Set appearance...`. In the `Main` section of the pop-up window change the symbol type from `None` to `Circle`. Change the line type from straight to none, since the lines added by default by Xmgrace are not meaningful. For the fitting, go to `Data -> Transformations -> Non-linear curve fitting`. In this window, select the source from the `Set` box and in the `Formula` box insert the following

```
y = 0.5 * erfc( ( x - A0 ) / A1 )
```

Select 2 as number of parameters, give 40 as initial condition for `A0` and 7 for `A1`. Click `Apply`. A new window should pop up with the stats of the fitting. In particular you should find a `Correlation coefficient` of 0.96 and a value of 39.9756 for `A0` and 6.6529 for `A1`. These are the value of μ_{fit} and σ_{fit} we are going to use for the SCDM method. In particular, $\mu_{SCDM} = \mu_{fit} - 3\sigma_{fit} = 20.0169$ eV and $\sigma_{SCDM} = \sigma_{fit} = 6.6529$ eV. The motivation for this specific choice of μ_{fit} and σ_{fit} may be found in Ref. [23], where the authors also show validation of this approach on a dataset of 200 materials. You should now see the fitting function, as well as the projectabilities, in the graph (see Fig. 10-(a)).

7. Open `W.pw2wan` and append the following lines

   ```
   scdm_entanglement = 'erfc'
   scdm_mu = 20.0169
   scdm_proj = .true.
   scdm_sigma = 6.6529
   /
   ```

8. Run `pw2wannier90` to compute the overlaps between Bloch states and the projections for the starting guess (written in the `W.mmn` and `W.amn` files)
   `pw2wannier90.x -in W.pw2wan > pw2wan.out`

9. Run `wannier90` to obtain the interpolated bandstructure (see Fig. 10-(b)).
   `wannier90.x W`

   Please cite Ref. [23] in any publication employing the procedure outlined in this example to obtain μ and σ.

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).

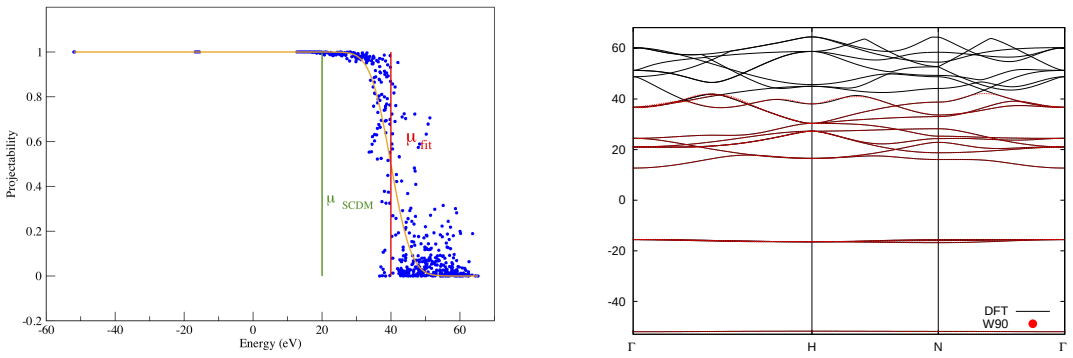
[5] A. A. Mostofi, J. R. Yates, Y.-S. Lee, I. Souza, D. Vanderbilt, and N. Marzari, Comput. Phys. Commun. **178**, 685 (2008).

[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).



(a) (b)

Figure 10: a) Each blue dot represents the projectability as defined in Eq. (22) of Ref. [23] of the state $|n\mathbf{k}\rangle$ as a function of the corresponding energy $\epsilon_{n\mathbf{k}}$ for tungsten. The yellow line shows the fitted complementary error function. The vertical red line represents the value of σ_{fit} while the vertical green line represents the optimal value of μ_{SCDM}, i.e. $\mu_{SCDM} = \mu_{fit} - 3\sigma_{fit}$. b) Band structure of tungsten on the Γ-H-N-Γ path from DFT calculations (solid black) and Wannier interpolation using the SCDM method to construct the initial guess (red dots).

[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).

[14] S. S. Tsirkin, P. Aguado Puente, and I. Souza, ArXiv e-prints (2017), arXiv:1710.03204 [cond-mat.mtrl-sci] .

[15] J. Ibañez-Azpiroz, S. S. Tsirkin, and I. Souza, ArXiv e-prints (2018), arXiv:1804.04030 .

[16] T. Yusufaly, D. Vanderbilt, and S. Coh, "Tight-Binding Formalism in the Context of the PythTB Package," `http://physics.rutgers.edu/pythtb/formalism.html`.

[17] R. Wang, E. A. Lazar, H. Park, A. J. Millis, and C. A. Marianetti, Physical Review B **90** (2014), 10.1103/PhysRevB.90.165125.

[18] A. Damle and L. Lin, ArXiv e-prints (2017), 1703.06958 .

[19] J. Qiao, J. Zhou, Z. Yuan, and W. Zhao, Phys. Rev. B **98**, 214402 (2018).

[20] G. Y. Guo, S. Murakami, T.-W. Chen, and N. Nagaosa, Phys. Rev. Lett. **100**, 096401 (2008).

[21] G. Pizzi, A. Cepellotti, R. Sabatini, N. Marzari, and B. Kozinsky, Computational Materials Science **111**, 218 (2016).

[22] V. Vitale, G. Pizzi, A. Marrazzo, J. R. Yates, N. Marzari, and A. A. Mostofi, Materials Cloud Archive (2019), doi:10.24435/materialscloud:2019.0044/v2.

[23] V. Vitale, G. Pizzi, A. Marrazzo, J. Yates, N. Marzari, and A. Mostofi, "Automated high-throughput wannierisation," (2019), arXiv:1909.00433 .