wannier90: Tutorial

Version 1.0.3

31st July 2007

Preliminaries

Welcome to wannier90! The examples contained in this tutorial are designed to help you to become familiar with the procedure of generating, analysing and using maximally-localised Wannier functions (MLWF). As a first step, install wannier90 following the instructions in the README file of the wannier90 distribution. For an introduction to the theory underlying MLWF, 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], and to a recent paper [4] describing wannier90.

The following additional programs should be installed in order to visualise the output of wannier90

- 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://www.xmgrace.org
- XCrySDen is used to visualise crystal structures, MLWF, 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

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 (www.quantum-espresso.org) interface to wannier90. At the time of writing, interfaces to other electronic structure codes, such as CASTEP (www.castep.org), ABINIT (www.abinit.org), and FLEUR (www.flapw.de), are in progress.

Contact us

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

- Outline: *Obtain and plot MLWF 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 MLWF 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 MLWF are defined at each k-point by just the 4×4 unitary matrices $\mathbf{U}^{(\mathbf{k})}$.

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

   ```
   wannier_plot = true
   ```

   and re-running `wannier90`. To visualise the MLWF we must represent them explicitly on a real space grid (see Ref. [1]). As a consequence, plotting the MLWF 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 MLWF may be time consuming and require a lot of memory. Use the keyword `wannier_plot_list` to plot a subset of the MLWF. E.g., to plot the 1st, 2nd and 7th MLWF use

   ```
   wannier_plot_list = 1 2 7
   ```

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

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

Hint: For a finite k-point mesh, the MLWF 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 MLWF "live" in a supercell n times the unit cell.

2: Lead

- Outline: *Obtain MLWF 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 MLWF of these states have partial occupancy. MLWF describing only the occupied states would be poorly localised.

1. Run `wannier90` to minimise the MLWF spread

   ```
   wannier90.x lead
   ```

   Inspect the output file `lead.wout`.

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

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

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

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

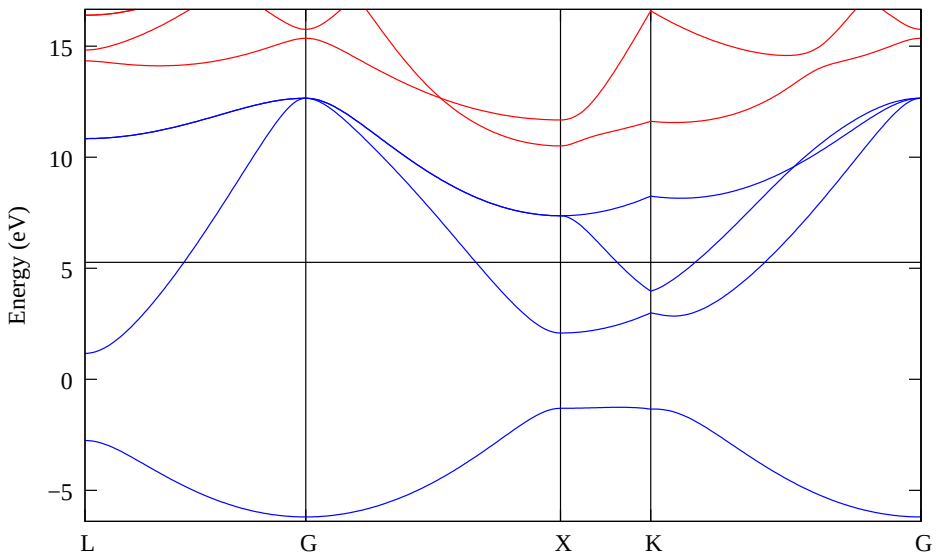

Figure 1: Bandstructure of Lead showing the position of the Fermi level. Only the lowest four bands are included in the calculation.

3: Silicon

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

The valence and lower conduction states can be represented by MLWF 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 MLWF 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.

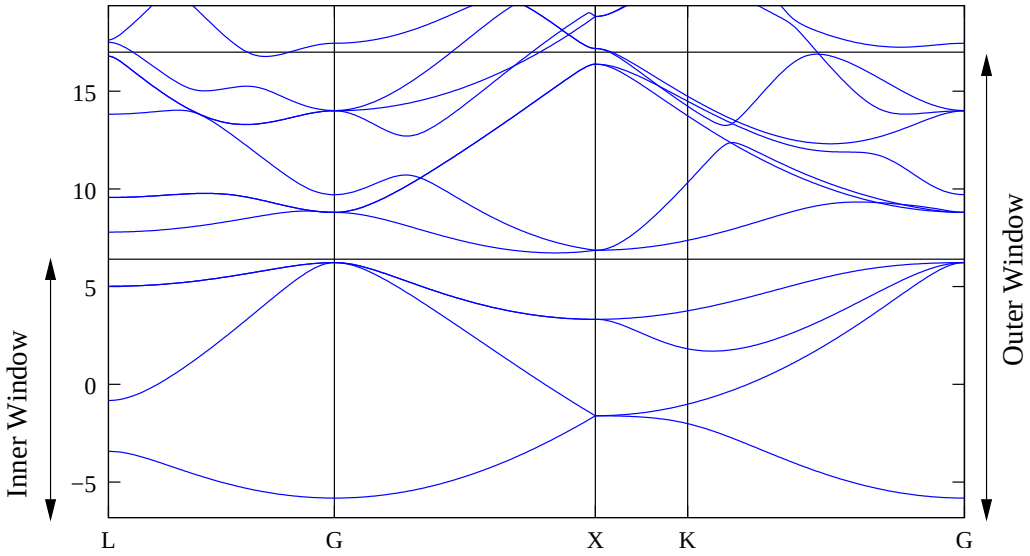

Figure 2: Bandstructure of silicon showing the position of the outer and inner energy windows.

4: Copper

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

1. Run `wannier90` to minimise the MLWF 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
   ```

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

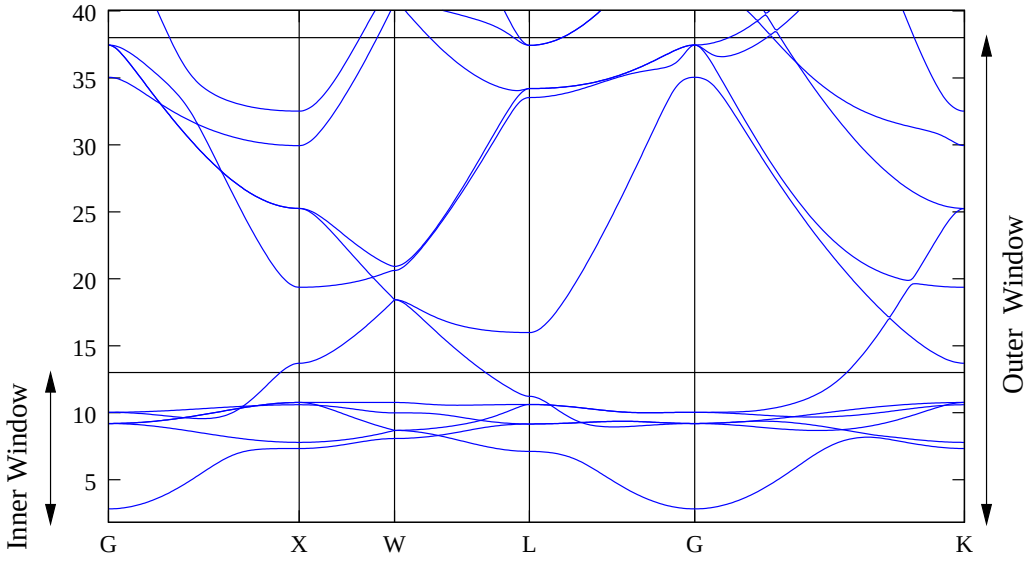

Figure 3: Bandstructure of copper showing the position of the outer and inner energy windows.

Examples Using the PWSCF Interface

The PWSCF plane-wave, density-functional theory code, which is available as part of the QUANTUM-ESPRESSO distribution (`www.quantum-espresso.org`), is fully interfaced to `wannier90` via the `pw2wannier90` post-processing code that is also available as part of QUANTUM-ESPRESSO.

5: Diamond

- Outline: *Obtain MLWF 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 MLWF.
   `wannier90.x diamond`

6: Copper

- Outline: *Obtain MLWF 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 MLWF.
   `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 MLWF using a more dense k-point grid.
- Investigate the effect of the outer and inner energy window 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 projected bandstructures.

7: Silane - SiH_4

- Outline: *Obtain MLWF 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 MLWF.
   `wannier90.x silane`

8: Iron

- Outline: *Obtain MLWF to describe the states around the Fermi-level in iron.*
- Directory: `examples/example8/`

- Input Files
  - `iron.scf` *The* PWSCF *input file for ground state calculation*
  - `iron.nscf` *The* PWSCF *input file to obtain Bloch states on a uniform grid*
  - `iron_up.pw2wan` *Input file for* `pw2wannier90` - *spin up*
  - `iron_up.win` *The* `wannier90` *input file - spin up*
  - `iron_dn.pw2wan` *Input file for* `pw2wannier90` - *spin down*
  - `iron_dn.win` *The* `wannier90` *input file - spin down*
- Note that we obtain the MLWF for spin up and spin down in separate calculations.

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

9: Cubic $BaTiO_3$

- Outline: *Obtain MLWF 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 MLWF for the oxygen 2p states. From the bandstructure [6], these form an isolated group of bands. We use the wannier90 keyword exclude_bands to remove all but the 2p bands from the calculation of the overlap and projection matrices (we don't have to do this, but it saves time).

1. Run PWSCF to obtain the ground state of $BaTiO_3$
   `pw.x < BaTiO3.scf > scf.out`

2. Run PWSCF to obtain the Bloch states on a uniform k-point grid
   `pw.x < BaTiO3.nscf > nscf.out`

3. Run wannier90 to generate a list of the required overlaps (written into the BaTiO3.nnkp file).
   `wannier90.x -pp BaTiO3`

4. Run pw2wannier90 to compute the overlap between Bloch states and the projections for the starting guess (written in the BaTiO3.mmn and BaTiO3.amn files).
   `pw2wannier90.x < BaTiO3.pw2wan > pw2wan.out`

5. Run wannier90 to compute the MLWF.
   `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 MLWF (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 MLWF for other groups of bands. What happens if you form MLWF for the whole valence manifold?
- Following Ref. [6], compute the Born effective charges from the change in Wannier centres under an atomic displacement.

10: Graphite

- Outline: *Obtain MLWF 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 MLWF.
   `wannier90.x graphite`

11: Silicon

Valence Bands

- Outline: *Obtain MLWF 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 rquest 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 MLWF.
   `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 MLWF are defined by just the 4×4 unitary matrices $\mathbf{U}^{(\mathbf{k})}$.

Plot the MLWF 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 State

- Outline: *Obtain MLWF for the valence and low-lying conduction states of Si. Plot the interpolated bandstructure*
- 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 MLWF 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. 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. Compute MLWF using a finer k-point grid (e.g., 6×6×6 or 8×8×8) and note how the accuracy of the interpolation increases [7].
- Compute MLWF for four conduction states (see Ref. [3]).

References

[1] A. A. Mostofi and J. R. Yates, `wannier90`: User Guide v1.0.3, available at `http://www.wannier.org/user_guide.html`.

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

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

[4] A. A. Mostofi, J. R. Yates, Y.-S. Lee, I. Souza, D. Vanderbilt and N. Marzari, `wannier90`: A Tool for Obtaining Maximally-Localized Wannier Functions, *Comput. Phys. Commun.*, submitted (2007) and `http://arxiv.org/abs/cond-mat/xxxxxxx`.

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

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

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