wannier90: Tutorial

Version 1.2

15th January 2010

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://plasma-gate.weizmann.ac.il/Grace
- 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
- vmd can also be used to visualise crystal structures and MLWF.
  http://www.ks.uiuc.edu/Research/vmd

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. 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 www.quantum-espresso.org to download the package, and for installation instructions.

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

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

Contact us

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

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

1: Gallium Arsenide

QUANTUM-ESPRESSO

- 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.

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

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?

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.

SIESTA

- Outline: *Obtain and plot MLWF for the four valence bands of GaAs.*
- Generation details: *From* SIESTA*, using norm-conserving pseudopotentials and a 2×2×2 k-point grid. Starting guess: four bond-centred Gaussians.*
- Directory: `Wannier-examples/Example01-GaAs/`
- Input Files
  - `Ga.psf` *Pseudopotential file for Ga*
  - `As.psf` *Pseudopotential file for As*
  - `GaAs.fdf` *The input file to run bulk GaAs with* SIESTA
  - `GaAs.win` *The master input file for* WANNIER90

1. Run `wannier90` with the command line option `-pp` to generate the `GaAs.nnkp` file.

   ```
   wannier90.x -pp GaAs
   ```

2. Run SIESTA to construct the $\mathbf{M}^{(\mathbf{k},\mathbf{b})}$ overlap matrices, the $\mathbf{A}^{(\mathbf{k})}$ projection matrices, the file with the list of eigenvalues and the UNK files with the Bloch states in the real space unit cell.

   ```
   siesta < GaAs.fdf > GaAs.out
   ```

3. Rename the file that contains the eigenvalues The .EIG extension was used in SIESTA to store the eigenvalues in a different format. The eigenvalues in the format required by WANNIER90 are written in the .eigW file, to avoid name clashes with SIESTA's standard eigenvalue file in case-insensitive filesystems. Since WANNIER90 expects the .eig extension, we have to rename the file.

   ```
   mv GaAs.eigW GaAs.eig
   ```

4. 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 $\mathbf{U}^{(\mathbf{k})}$.

5. 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,[2] 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 0.10). Can you explain what you see?

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

2: Lead

QUANTUM-ESPRESSO

- 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
   ```

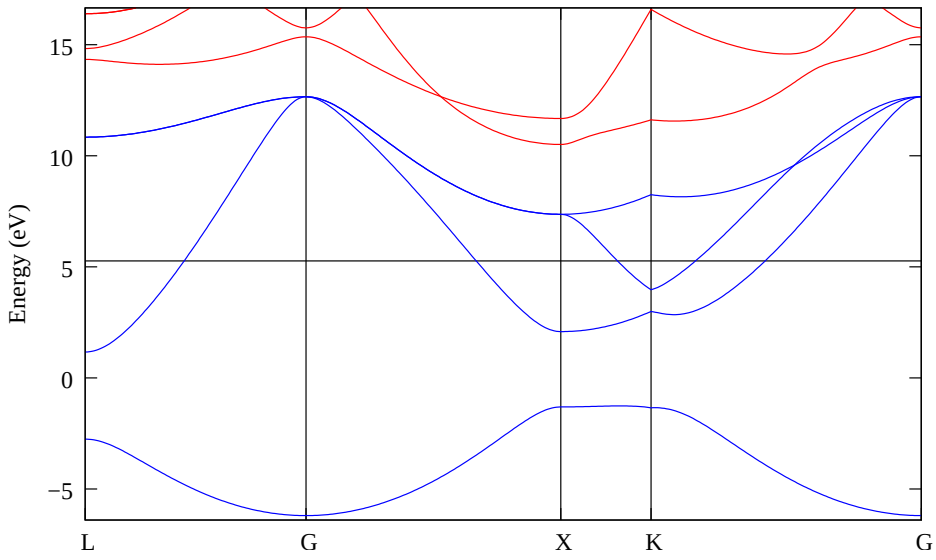

Figure 1: Bandstructure of lead showing the position of the Fermi level. Only the lowest four bands are included in the calculation.

SIESTA

- Outline: *Obtain MLWF for the four lowest states in lead. Use Wannier interpolation to plot the Fermi surface*
- Generation Details: *From* SIESTA*, using norm-conserving pseudopotentials and a 4×4×4 k-point grid. Starting guess: atom-centred sp^3 hybrid orbitals*
- Directory: `Wannier-examples/Example02-Pb/`
- Input Files
  - `Pb.psf` *Pseudopotential file for Pb*
  - `Pb.fdf` *The input file to run bulk Pb with* SIESTA
  - `Pb.win` *The master input file for* WANNIER90

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` with the command line option `-pp` to generate the `Pb.nnkp` file.

   ```
   wannier90.x -pp Pb
   ```

2. Run SIESTA to construct the $\mathbf{M}^{(\mathbf{k},\mathbf{b})}$ overlap matrices, the $\mathbf{A}^{(\mathbf{k})}$ projection matrices, the file with the list of eigenvalues at each $\mathbf{k}$-point.

   ```
   siesta < Pb.fdf > Pb.out
   ```

3. Rename the file that contains the eigenvalues The .EIG extension was used in SIESTA to store the eigenvalues in a different format. The eigenvalues in the format required by WANNIER90 are written in the .eigW file, to avoid name clashes with SIESTA's standard eigenvalue file in case-insensitive filesystems. Since WANNIER90 expects the .eig extension, we have to rename the file.

   ```
   mv Pb.eigW Pb.eig
   ```

4. Run `wannier90` to minimise the MLWF spread

   ```
   wannier90.x Pb
   ```

   Inspect the output file `Pb.wout`.

5. Use `wannier90`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 `Pb.win` file:

   ```
   restart = plot
   fermi_energy = -2.9449
   fermi_surface_plot = true
   ```

   and re-run `wannier90`. The value of the Fermi energy (-2.9449 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 `Pb.bxsf` can be viewed using XCrySDen, e.g.,

   ```
   xcrysden --bxsf Pb.bxsf
   ```

3: Silicon

QUANTUM-ESPRESSO

- 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.

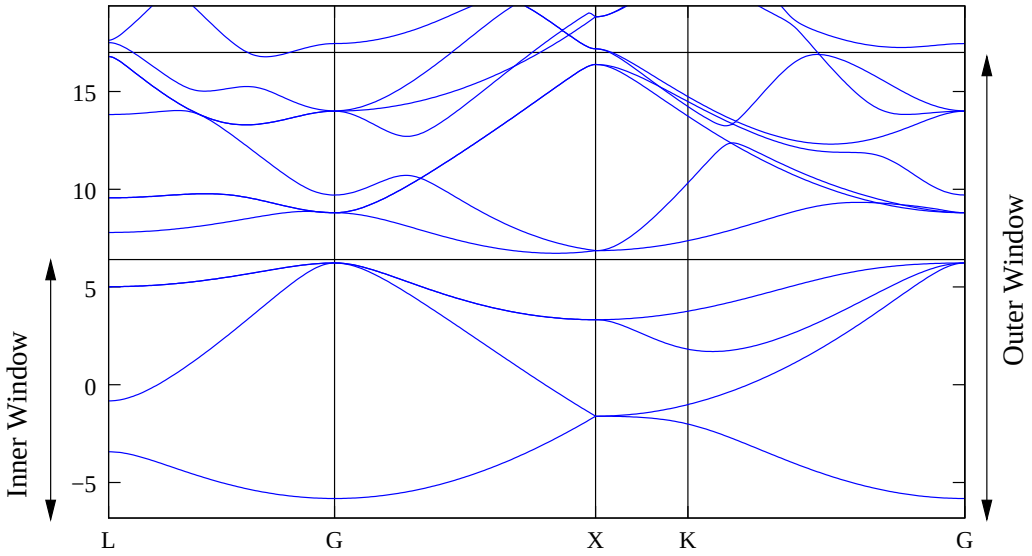

Figure 2: Bandstructure of silicon showing the position of the outer and inner energy windows.

SIESTA

- Outline: *Obtain MLWF for the valence and low-lying conduction states of Si. Plot the interpolated bandstructure*
- Generation Details: *From* SIESTA*, using norm-conserving pseudopotentials and a 4×4×4 k-point grid. Starting guess: atom-centred sp^3 hybrid orbitals*
- Directory: `Wannier-examples/Example03-Si/`
- Input Files
  - `Si.psf` *Pseudopotential file for Si*
  - `Si.DZP.fdf` *The input file to run bulk Si with* SIESTA *with a DZP basis set*
  - `Si.DZP.win` *The master input file for* WANNIER90*. Energy windows adapted to the DZP calculation*
  - `Si.TZTP.fdf` *The input file to run bulk Si with* SIESTA *with a triple-zeta, triple polarized plus f orbital basis set*
  - `Si.TZTPF.win` *The master input file for* WANNIER90*. Energy windows adapted to the TZTPF calculation*

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. 3.

1. Run `wannier90` with the command line option `-pp` to generate the `Si.DZP.nnkp` file.

```
wannier90.x -pp Si.DZP
```

2. Run SIESTA to construct the $\mathbf{M}^{(\mathbf{k},\mathbf{b})}$ overlap matrices, the $\mathbf{A}^{(\mathbf{k})}$ projection matrices, the file with the list of eigenvalues at each $\mathbf{k}$-point.

   ```
   siesta < Si.DZP.fdf > Si.DZP.out
   ```

3. Rename the file that contains the eigenvalues The .EIG extension was used in SIESTA to store the eigenvalues in a different format. The eigenvalues in the format required by WANNIER90 are written in the .eigW file, to avoid name clashes with SIESTA's standard eigenvalue file in case-insensitive filesystems. Since WANNIER90 expects the .eig extension, we have to rename the file.

   ```
   mv Si.DZP.eigW Si.DZP.eig
   ```

4. Run `wannier90`.

   ```
   wannier90.x Si.DZP
   ```

   Inspect the output file `Si.DZP.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 $\Omega_{\rm D} + \Omega_{\rm OD}$.

5. Plot the interpolated band structure by adding the following commands to the input file `Si.DZP.win`

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

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

   ```
   myshell> gnuplot
   gnuplot> load 'Si.DZP_band.gnu'
   ```

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

You can repeat the exercise with a much more complete basis set for Si (triple-zeta, triple polarized plus a shell of f orbitals. Just repeat the same steps as before replacing DZP by TZTPF. The agreements with the centers provided by WANNIER90 improves.

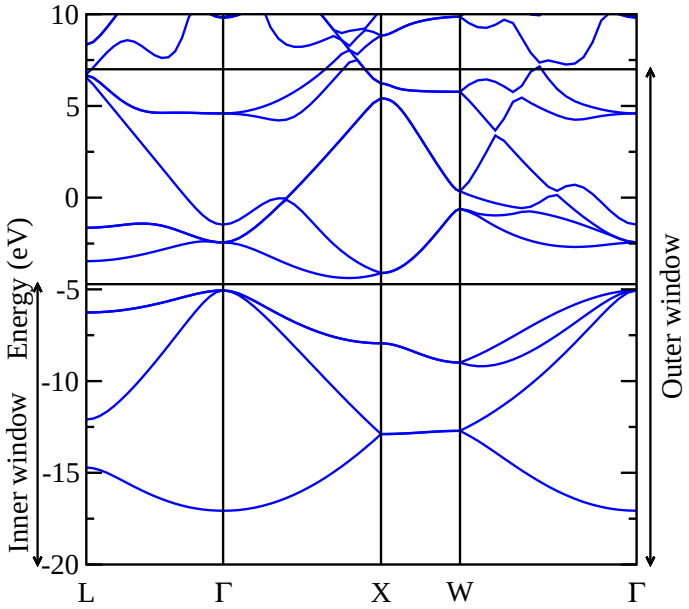

Figure 3: Bandstructure of silicon showing the position of the outer and inner energy windows as obtained with SIESTA with a double-zeta plus polarization basis set.

4: Copper

QUANTUM-ESPRESSO

- 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
   ```

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.

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

   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.

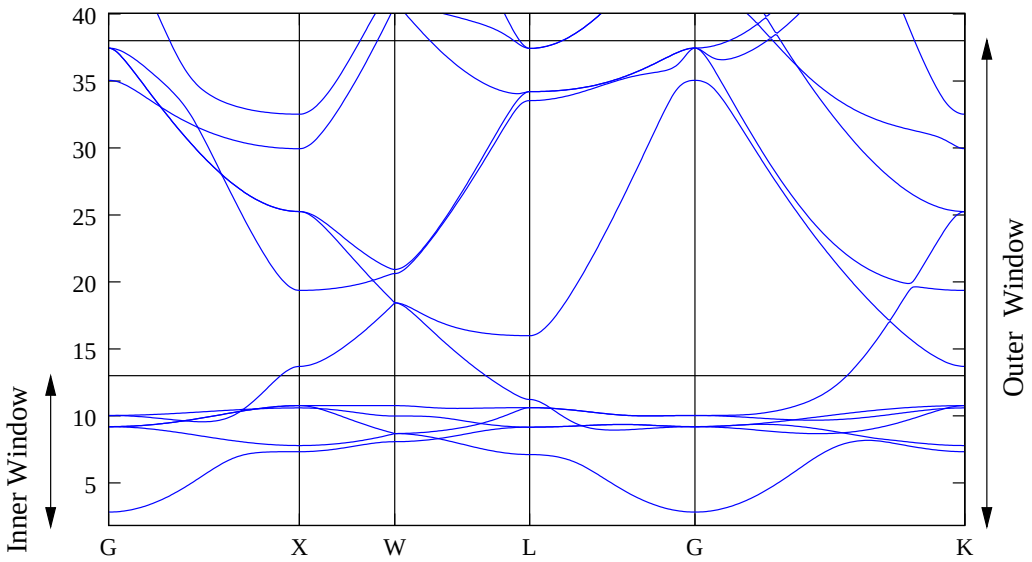

Figure 4: Bandstructure of copper showing the position of the outer and inner energy windows.

SIESTA

- Outline: *Obtain MLWF to describe the states around the Fermi-level in copper*
- Generation Details: *From* SIESTA*, using norm-conserving pseudopotentials 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: `Wannier-examples/Example04-Cu/`
- Input Files
  - `Cu.psf` *Pseudopotential file for Cu*
  - `Cu.fdf` *The input file to run bulk Cu with* SIESTA *with a DZP basis set*
  - `Cu.win` *The master input file for* WANNIER90*.*

1. Run `wannier90` with the command line option `-pp` to generate the `Cu.nnkp` file.

   ```
   wannier90.x -pp Cu
   ```

2. Run SIESTA to construct the $\mathbf{M}^{(\mathbf{k},\mathbf{b})}$ overlap matrices, the $\mathbf{A}^{(\mathbf{k})}$ projection matrices, the file with the list of eigenvalues at each $\mathbf{k}$-point.

   ```
   siesta < Cu.fdf > Cu.out
   ```

3. Rename the file that contains the eigenvalues The .EIG extension was used in SIESTA to store the eigenvalues in a different format. The eigenvalues in the format required by WANNIER90 are written in the .eigW file, to avoid name clashes with SIESTA's standard eigenvalue file in case-insensitive filesystems. Since WANNIER90 expects the .eig extension, we have to rename the file.

```
mv Cu.eigW Cu.eig
```

4. Run wannier90 to minimise the MLWF spread

   ```
   wannier90.x Cu
   ```

   Inspect the output file Cu.wout.

5. Plot the Fermi surface, it should look familiar! The Fermi energy is at -1.8718 eV. For this, use WANNIER90 interpolation. 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 Cu.win file:

   ```
   restart = plot
   fermi_energy = -1.8718
   fermi_surface_plot = true
   ```

   and re-run wannier90. The value of the Fermi energy (-1.8718 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 Cu.bxsf can be viewed using XCrySDen, e.g.,

   ```
   xcrysden --bxsf Cu.bxsf
   ```

6. Plot the interpolated bandstructure. A suitable path in k-space is For this add the following commands to the input file Cu.win

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

   and re-running wannier90. The files Cu_band.dat and Cu_band.gnu are created. 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
   ```

   To plot the bandstructure using gnuplot

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

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

7. Plot the contribution of the interstitial WF to the bandstructure. Add the following keyword to `Cu.win`

   ```
   bands_plot_project = 6,7
   ```

   The resulting file `Cu_band_proj.gnu` can be opened with gnuplot.

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

   Red lines correspond to a large contribution from the interstitial WF (blue line are a small contribution; i.e. a large '*d*' contribution).

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

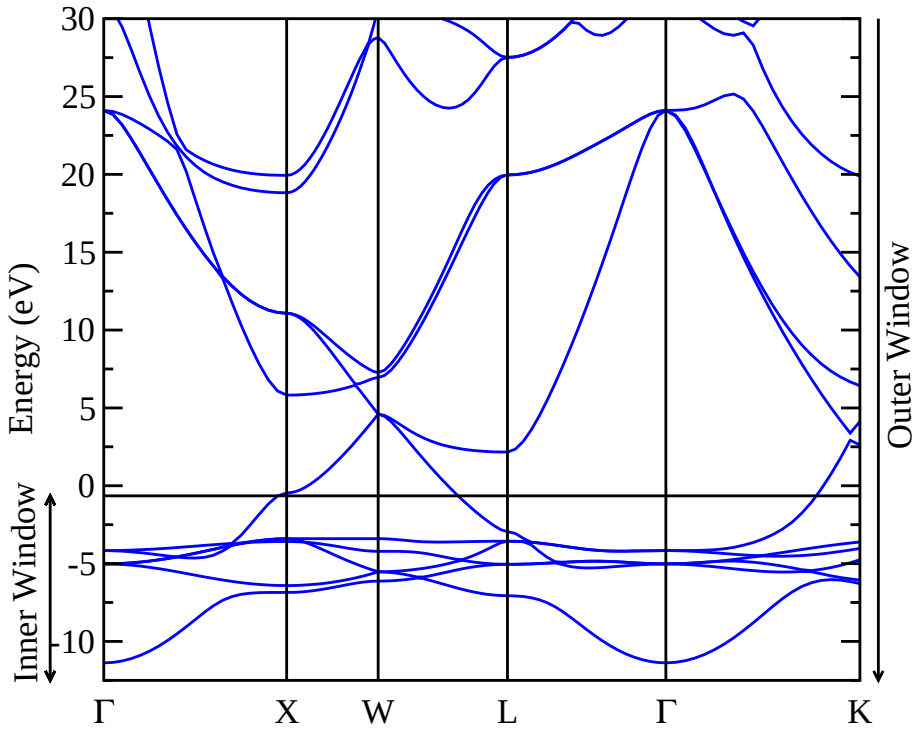

Figure 5: Bandstructure of copper showing the position of the outer and inner energy windows as obtained with SIESTA with a DZP basis set.

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. 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.

5: Diamond

QUANTUM-ESPRESSO

- 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`

SIESTA

- Outline: *Obtain MLWF for the valence bands of diamond*
- Generation Details: *From* SIESTA*, using norm-conserving pseudopotentials and a 4×4×4 k-point grid. Starting guess: atom-centred* sp^3 *hybrid orbitals*
- Directory: `Wannier-examples/Example05-C/`
- Input Files
  - `C.psf` *Pseudopotential file for C*
  - `C.fdf` *The input file to run diamond with* SIESTA
  - `C.win` *The master input file for* WANNIER90

1. Run `wannier90` to generate a list of the required overlaps (written into the `C.nnkp` file).

   ```
   wannier90.x -pp C
   ```

2. Run `siesta` to compute the overlap between Bloch states and the projections for the starting guess (written in the `C.mmn` and `C.amn` files).

   ```
   siesta < C.fdf > C.out
   ```

3. Rename the file that contains the eigenvalues The .EIG extension was used in SIESTA to store the eigenvalues in a different format. The eigenvalues in the format required by WANNIER90 are written in the .eigW file, to avoid name clashes with SIESTA's standard eigenvalue file in case-insensitive filesystems. Since WANNIER90 expects the .eig extension, we have to rename the file.

   ```
   mv C.eigW C.eig
   ```

4. Run `wannier90` to compute the MLWF.

   ```
   wannier90.x C
   ```

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

QUANTUM-ESPRESSO

- 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`

SIESTA

- Outline: *Obtain MLWF for the occupied states of molecular silane.* Γ*-point sampling*
- Generation Details: *From* SIESTA*, using norm-conserving pseudopotentials. Only* Γ *point. Starting guess: Si-centred* sp^3 *hybrid orbitals*
- Directory: `Wannier-examples/Example07-SiH4/`
- Input Files
  - `Si.psf` *Pseudopotential file for Si*
  - `H.psf` *Pseudopotential file for H*
  - `silane.fdf` *The input file to run silane with* SIESTA
  - `silane.win` *The master input file for* WANNIER90

1. Run `wannier90` to generate a list of the required overlaps (written into the `silane.nnkp` file).

   ```
   wannier90.x -pp silane
   ```

2. Run SIESTA to compute the overlap between Bloch states and the projections for the starting guess (written in the `silane.mmn` and `silane.amn` files).

   ```
   siesta < silane.fdf > silane.out
   ```

3. Rename the file that contains the eigenvalues The .EIG extension was used in SIESTA to store the eigenvalues in a different format. The eigenvalues in the format required by WANNIER90 are written in the .eigW file, to avoid name clashes with SIESTA's standard eigenvalue file in case-insensitive filesystems. Since WANNIER90 expects the .eig extension, we have to rename the file.

   ```
   mv silane.eigW silane.eig
   ```

4. Run `wannier90` to compute the MLWF.

   ```
   wannier90.x silane
   ```

8: Iron

QUANTUM-ESPRESSO

- 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`

SIESTA

- Outline: *Obtain MLWF to describe the states around the Fermi-level in iron.*
- Generation Details: *From* SIESTA*, using norm-conserving pseudopotentials. Starting guess: sp3d2; dxy; dxz; dyz*
- Directory: `Wannier-examples/Example08-Fe/`

- Input Files
  - `Fe.psf` *Pseudopotential file for Fe*
  - `Fe.fdf` *The input file to run bulk Fe with* SIESTA
  - `Fe_up.win` *The master input file for* WANNIER90*, spin up*
  - `Fe_dn.win` *The master input file for* WANNIER90*, spin down*

1. Run `wannier90` to generate a list of the required overlaps (written into the `Fe_up.nnkp` and `Fe_dn.nnkp` files).

   ```
   wannier90.x -pp Fe_up

   wannier90.x -pp Fe_dn
   ```

2. Run SIESTA to compute the overlap between Bloch states and the projections for the starting guess (written in the `Fe_up.mmn`, `Fe_dn.mmn`, `Fe_up.amn`, and `Fe_dn.amn` files).

   ```
   siesta < Fe.fdf > Fe.out
   ```

3. Rename the file that contains the eigenvalues The .EIG extension was used in SIESTA to store the eigenvalues in a different format. The eigenvalues in the format required by WANNIER90 are written in the .eigW file, to avoid name clashes with SIESTA's standard eigenvalue file in case-insensitive filesystems. Since WANNIER90 expects the .eig extension, we have to rename the file.

   ```
   mv Fe_up.eigW Fe_up.eig

   mv Fe_dn.eigW Fe_dn.eig
   ```

4. Run `wannier90` to compute the MLWF.

   ```
   wannier90.x Fe_up

   wannier90.x Fe_dn
   ```

9: Cubic $BaTiO_3$

QUANTUM-ESPRESSO

- 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.

SIESTA

- Outline: *Obtain MLWF for a perovskite*
- Generation Details: *From* SIESTA*, using norm-conserving pseudopotentials. To start with, we are going to obtain MLWF for the oxygen 2p*
- Directory: `Wannier-examples/Example09-BaTiO3/`
- Input Files
  - `Ba.psf` *Pseudopotential file for Ba*
  - `Ti.psf` *Pseudopotential file for Ti*
  - `O.psf` *Pseudopotential file for O*
  - `BaTiO3.fdf` *The input file to run bulk* $BaTiO_3$ *with* SIESTA
  - `BaTiO3.win` *The master input file for* WANNIER90

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 `wannier90` to generate a list of the required overlaps (written into the `BaTiO3.nnkp` file).

   ```
   wannier90.x -pp BaTiO3
   ```

2. Run SIESTA to compute the overlap between Bloch states and the projections for the starting guess (written in the `BaTiO3.mmn` and `BaTiO3.amn` files).

```
siesta < BaTiO3.fdf > BaTiO3.out
```

3. Rename the file that contains the eigenvalues The .EIG extension was used in SIESTA to store the eigenvalues in a different format. The eigenvalues in the format required by WANNIER90 are written in the .eigW file, to avoid name clashes with SIESTA's standard eigenvalue file in case-insensitive filesystems. Since WANNIER90 expects the .eig extension, we have to rename the file.

```
mv BaTiO3.eigW BaTiO3.eig
```

4. Run `wannier90` to compute the MLWF.

```
wannier90.x BaTiO3
```

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
```

Try values of the isosurface, for instance, of 0.2 and -0.2 [3]

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.

This requires the modification in the BaTiO3.fdf file of the block

```
%block AtomicCoordinatesAndAtomicSpecies
    0.0    0.0    0.0   1   137.3277    Ba
    0.505  0.5    0.5   2   47.8671     Ti
    0.0    0.5    0.5   3   15.9994     O
    0.5    0.5    0.0   3   15.9994     O
    0.5    0.0    0.5   3   15.9994     O
%endblock AtomicCoordinatesAndAtomicSpecies
```

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

and the modification of the BaTiO3.win file,

```
begin atoms_frac
Ba 0.0   0.0   0.0
Ti 0.505 0.5   0.5
O  0.0   0.5   0.5
O  0.5   0.5   0.0
O  0.5   0.0   0.5
end atoms_frac
```

and run again

```
siesta < BaTiO3.fdf > BaTiO3.out

mv BaTiO3.eigW BaTiO3.eig

wannier90.x BaTiO3
```

to compute the ground-state charge density and Bloch states using SIESTA, etc.). 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

QUANTUM-ESPRESSO

- 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`

SIESTA

- Outline: *Obtain MLWF for graphite (AB, Bernal)*
- Generation Details: *From* SIESTA*, using norm-conserving pseudopotentials. Starting guess: C1:sp2;pz C2:pz*
- Directory: `Wannier-examples/Example10-Graphite/`
- Input Files
  - `C.psf` *Pseudopotential file for C*
  - `graphite.fdf` *The input file to run graphite with* SIESTA
  - `graphite.win` *The master input file for* WANNIER90

1. Run `wannier90` to generate a list of the required overlaps (written into the `graphite.nnkp` file).

   ```
   wannier90.x -pp graphite
   ```

2. Run `siesta` to compute the overlap between Bloch states and the projections for the starting guess (written in the `graphite.mmn` and `graphite.amn` files).

   ```
   siesta < graphite.fdf > graphite.out
   ```

3. Rename the file that contains the eigenvalues The .EIG extension was used in SIESTA to store the eigenvalues in a different format. The eigenvalues in the format required by WANNIER90 are written in the .eigW file, to avoid name clashes with SIESTA's standard eigenvalue file in case-insensitive filesystems. Since WANNIER90 expects the .eig extension, we have to rename the file.

   ```
   mv graphite.eigW graphite.eig
   ```

4. 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 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 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. 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. 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]).

12: Benzene

Quantum-Espresso: Valence States

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

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

Plot the MLWF 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.

siesta: Valence States

- Outline: *Obtain MLWF for the valence states of benzene*
- Generation Details: *From* SIESTA*, using norm-conserving pseudopotentials.*
- Directory: `Wannier-examples/Example12-Benzene/`

- Input Files
  - `C.psf` *Pseudopotential file for C*
  - `H.psf` *Pseudopotential file for H*
  - `benzene-valence.fdf` *The input file to run benzene with* SIESTA
  - `benzene-valence.win` *The master input file for* WANNIER90

1. Run `wannier90` to generate a list of the required overlaps (written into the `benzene-valence.nnkp` file).

   ```
   wannier90.x -pp benzene-valence
   ```

2. Run SIESTA to compute the overlap between Bloch states and the projections for the starting guess (written in the `benzene-valence.mmn` and `benzene-valence.amn` files).

   ```
   siesta < benzene-valence.fdf > benzene-valence.out
   ```

3. Rename the file that contains the eigenvalues The .EIG extension was used in SIESTA to store the eigenvalues in a different format. The eigenvalues in the format required by WANNIER90 are written in the .eigW file, to avoid name clashes with SIESTA's standard eigenvalue file in case-insensitive filesystems. Since WANNIER90 expects the .eig extension, we have to rename the file.

   ```
   mv benzene-valence.eigW benzene-valence.eig
   ```

4. Run `wannier90` to compute the MLWF.

   ```
   wannier90.x benzene-valence
   ```

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

Plot the MLWF 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.

```
xcrysden --cube benzene-valence_00002.cube
```

Valence + Conduction States

- Outline: *Obtain MLWF 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 MLWF 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.

SIESTA: Valence + Conduction States

- Outline: *Obtain MLWF for the valence plus conduction states of benzene*
- Generation Details: *From* SIESTA*, using norm-conserving pseudopotentials.*
- Directory: `Wannier-examples/Example12-Benzene/`
- Input Files
  - `C.psf` *Pseudopotential file for C*
  - `H.psf` *Pseudopotential file for H*
  - `benzene-valence+cond.fdf` *The input file to run benzene with* SIESTA
  - `benzene-valence+cond.win` *The master input file for* WANNIER90

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 `wannier90` to generate a list of the required overlaps (written into the `benzene-valence+cond.nnk` file).

   ```
   wannier90.x -pp benzene-valence+cond
   ```

2. Run SIESTA to compute the overlap between Bloch states and the projections for the starting guess (written in the `benzene-valence.mmn` and `benzene-valence.amn` files).

   ```
   siesta < benzene-valence+cond.fdf > benzene-valence+cond.out
   ```

3. Rename the file that contains the eigenvalues The .EIG extension was used in SIESTA to store the eigenvalues in a different format. The eigenvalues in the format required by WANNIER90 are written in the .eigW file, to avoid name clashes with SIESTA's standard eigenvalue file in case-insensitive filesystems. Since WANNIER90 expects the .eig extension, we have to rename the file.

   ```
   mv benzene-valence+cond.eigW benzene-valence+cond.eig
   ```

4. Run `wannier90` to compute the MLWF.

   ```
   wannier90.x benzene-valence+cond
   ```

5. Plot the MLWF 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.

   ```
   xcrysden --cube benzene-valence+cond_00001.cube
   ```

13: (5,5) Carbon Nanotube

Transport properties

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

In order to form localised WF that describe both the occupied and unoccupied π and π^* manifolds, we use the disentanglement procedure to extract a smooth manifold of states that has dimension equal to 2.5 times the number of carbon atoms per unit cell [8]. The positions of the energy windows are shown in Fig. 6.

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. 7.

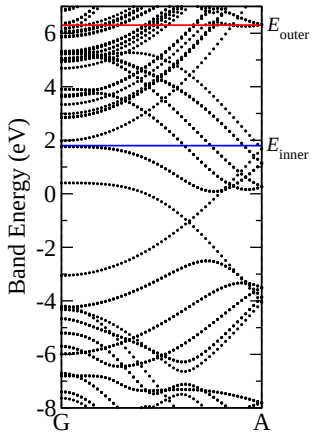

Figure 6: Bandstructure of (5,5) carbon nanotube showing the position of the outer and inner energy windows.

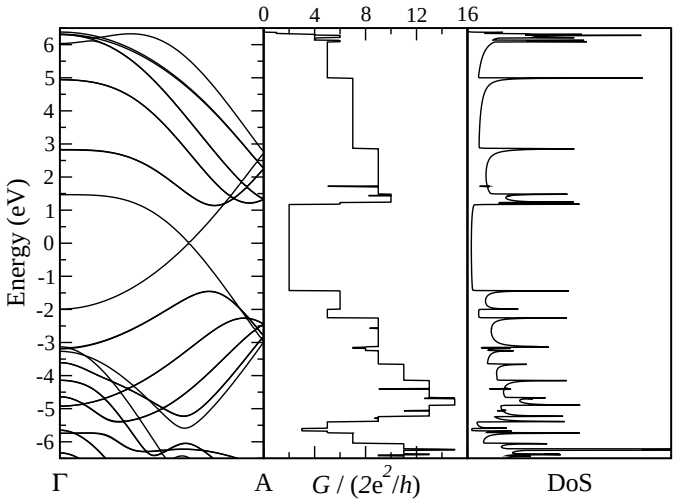

Figure 7: Wannier interpolated bandstructure, quantum conductance and density of states of (5,5) carbon nanotube. Note that the Fermi level has been shifted by 1.06eV with respect to Fig. 6.

14: Linear Sodium Chain

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

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

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

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

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

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

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

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

1. Run PWSCF and `wannier90` for the periodic system.
2. Run PWSCF and `wannier90` for the defected system.
3. The quantum conductance is written to the files `periodic/Na_chain_qc.dat` and `defected/Na_chain_dos.dat`, respectively. Compare the quantum conductance of the periodic (bulk) calculation with the defected (LCR) calculation. Your plot should look like Fig. 8.

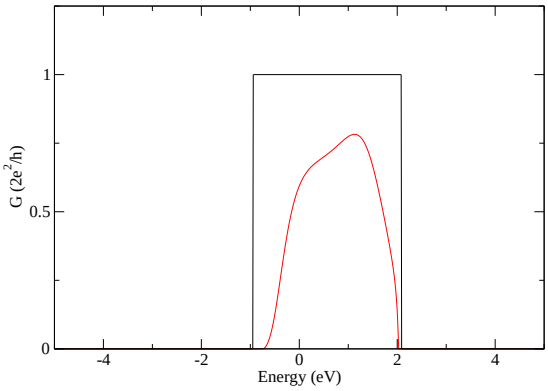

Figure 8: Quantum conductance of periodic Sodium chain (black) compared to that of the defected Sodium chain (red).

15: (5,0) Carbon Nanotube

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 a 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 2 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. 9.

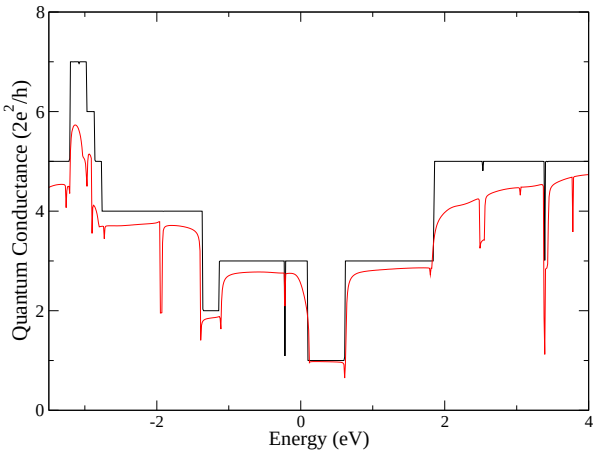

Figure 9: Quantum conductance of infinite pristine nanotube (black) compared to that of the infinite nanotube with the substitutional silicon defect (red).

Further ideas

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

References

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

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

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

[4] A. A. Mostofi, J. R. Yates, Y.-S. Lee, I. Souza, D. Vanderbilt and N. Marzari, wannier90: A Tool for Obtaining Maximally-Localized Wannier Functions, *Comput. Phys. Commun.*, accepted (2007) and http://arxiv.org/abs/0708.0650.

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

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

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

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