Wannier90: Tutorial

Version 1.0

April 26, 2006

Preliminaries Install the Wannier90 code following the instructions in the README file. Refer to the overview of the code given in Chapter 1 of the Wannier90 User guide and the two papers, MV[1] nd SMV[2].

The following additional programs should be installed to visualise the output of Wannier90

- `gnuplot`; a graph plotting program is used to plot band structures. It is available for many operating systems and is often installed by default on unix/Linux distributions
  `http://www.gnuplot.info`
- `XCrysden` is used to visualise crystal structures, Wannier functions 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 the tutorial comprises 4 examples taken from the MV and SMV papers. All of the 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 to Wannier90, however these can be easily modified for other electronic structure codes.

[1] *Maximally localized generalized Wannier functions for composite energy bands*
N. Marzari and D. Vanderbilt, Phys. Rev. B 56, 12847 (1997)

[2] *Maximally localized Wannier functions for entangled energy bands*
I. Souza, N. Marzari and D. Vanderbilt, Phys. Rev. B 65, 035109 (2002)

1: Gallium Arsenide

- Outline: *Obtain and plot Wannier functions for the 4 valence states of GaAs.*
- Generation details: *From pwscf using norm-conserving pseudopotentials and a 2x2x2 kpoint grid. Starting guess: 4 bond centred Gaussians.*
- Directory: `examples/example1/`
- Input Files
  - `gaas.win` *The master input file*
  - `gaas.mmn` *The overlap matrices* $M_{\mathbf{k}}$
  - `gaas.amn` *Projection of the Bloch states onto a set of trial localised orbitals* $A_{\mathbf{k}}$
  - `UNK00001.1` *The Bloch states in the real space unit cell. For plotting only.*

1. Run the `Wannier90` code to minimise the Wannier function 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 Wannier function 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 Wannier functions are defined at each kpoint by just the 4x4 unitary matrix, $U_{\mathbf{k}}$.

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

   ```
   wannier_plot = true
   ```

   and re-running Wannier90. To visualise the Wannier functions we must explicitly represent them on a real space grid (see User Guide Eq 2). As a consequence plotting the Wannier functions is slower and uses more memory than the minimisation of the spread. The 4 files that are created, `gaas_00001.xsf` etc can be viewed using XCrysden

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

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

   ```
   wannier_plot_list = 1 2 7
   ```

   The Wannier functions 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 8 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 Wannier function 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 kpoint mesh the Wannier functions are in fact periodic, where the period is related to spacing of the kpoint mesh. For an kpoint mesh with n divisions in the ith direction in the Brillouin zone the Wannier functions live in a supercell n times the unit cell.

2: Lead

- Outline: *Obtain Wannier Functions for the 4 lowest states in Lead. Use Wannier Interpolation to plot the Fermi Surface*
- Generation Details: *From pwscf using norm-conserving pseudopotentials and a 4x4x4 kpoint 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* $M_{\mathbf{k}}$
  - `lead.amn` *Projection of the Bloch states onto a set of trial localised orbitals* $A_{\mathbf{k}}$
  - `lead.eig` *The Bloch eigenvalues at each kpoint. For interpolation only*

The four lowest valence bands in lead are separated in energy from the higher conduction states, see Figure 1. The Wannier functions of these states have partial occupancy. Wannier functions describing just the occupied states would exhibit very poor localisation properties.

1. Run the `Wannier90` code to minimise the Wannier function spread

   ```
   wannier90.x lead
   ```

   Inspect the output file `lead.wout`.

2. Use Wannier interpolation to obtain the Fermi surface of lead. Rather than rerunning the whole calculation we can use the unitary transformations obtained in the first calculation, restarting 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. The Wannier90 code calculates the band energies, through wannier interpolation, on a dense mesh of kpoints in the Brillouin zone. The density of this grid is controlled by the keyword `fermi_surface_num_points`. The default value is 50 (ie 50^3 points). The Fermi surface file `lead.bxsf` can be viewed using XCrysden

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

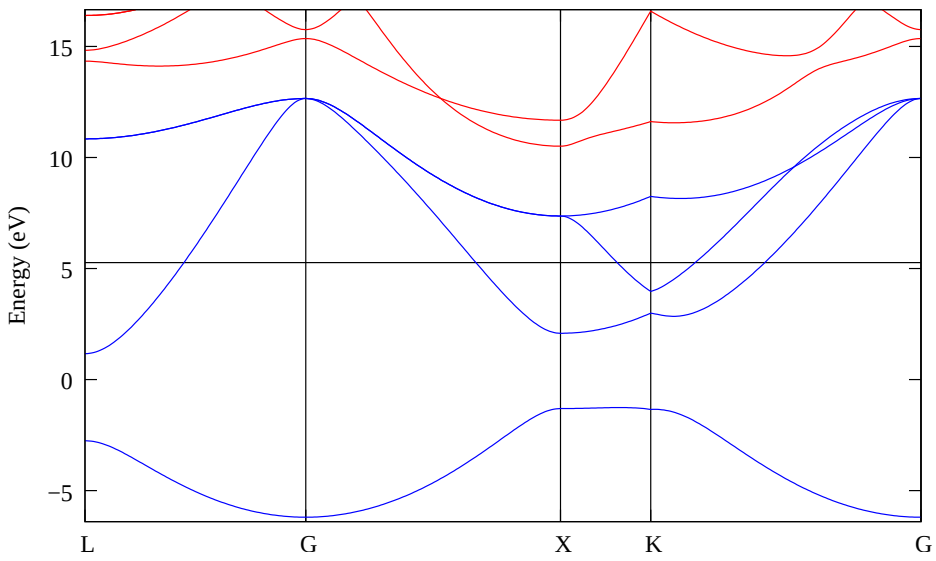

Figure 1: Band Structure of Lead showing the position of the Fermi level. Only the lowest 4 bands are included in the calculation.

3: Silicon

- Outline: *Obtain MLWF for the valence and lower conduction states of Si. Plot the interpolated band structure*
- Generation Details: *From pwscf using norm-conserving pseudopotentials and a 4x4x4 kpoint 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* $M_{\mathbf{k}}$
  - `silicon.amn` *Projection of the Bloch states onto a set of trial localised orbitals* $A_{\mathbf{k}}$
  - `silicon.eig` *The Bloch eigenvalues at each kpoint*

The valence and lower conduction states can be represented by Wannier Functions 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 Wannier function we use the disentanglement procedure introduced in SMV. The position of the inner and outer energy windows are shown in Figure 2.

1. Run the `Wannier90` code.

   ```
   wannier90.x silicon
   ```

   Inspect the output file `silicon.wout`. The minimisation of the spread occurs in a two step procedure. We first minimise Ω_{I}, this is the extraction of the optimal subspace in the disentanglement procedure. We then minimise $\Omega_{\mathrm{O}} + \Omega_{\mathrm{OD}}$

2. Plot the Wannier Functions 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 band-structure using gnuplot

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

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

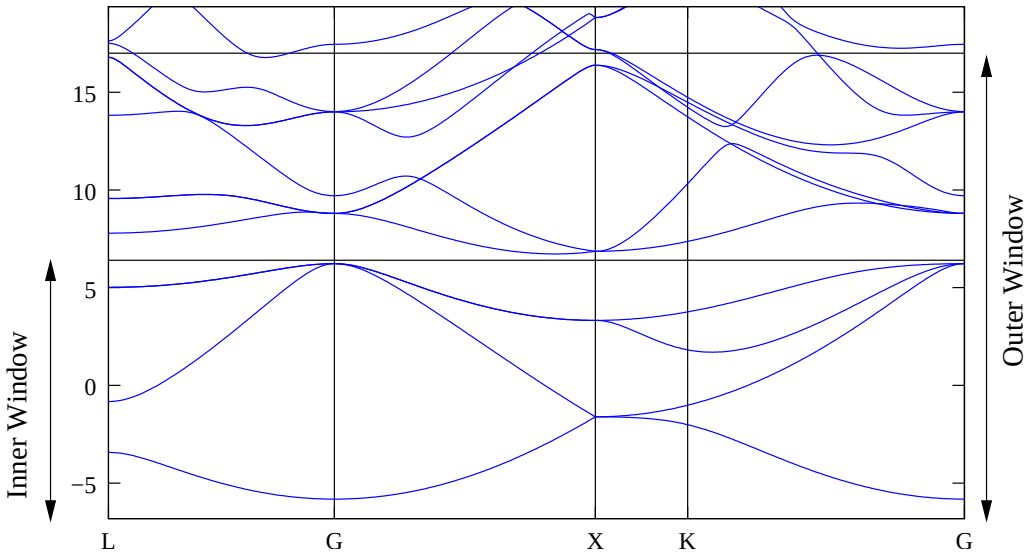

Figure 2: Band Structure of Silicon showing the position of the outer and inner energy windows.

4: Copper

- Outline: *Obtain MLWF for the 4 valence states of Copper. Plot the MLWF*
- Generation Details: *From pwscf using "ultrasoft" pseudopotentials and a 4x4x4 kpoint grid. Starting guess: 5 atom centred 'd' orbital 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* $M_{\mathbf{k}}$
  - `copper.amn` *Projection of the Bloch states onto a set of trial localised orbitals* $A_{\mathbf{k}}$
  - `copper.eig` *The Bloch eigenvalues at each kpoint*

1. Run the `Wannier90` code to minimise the Wannier function spread

   ```
   wannier90.x copper
   ```

   Inspect the output file `copper.wout`.

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

3. Plot the interpolated band structure. A suitable path in kspace 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.

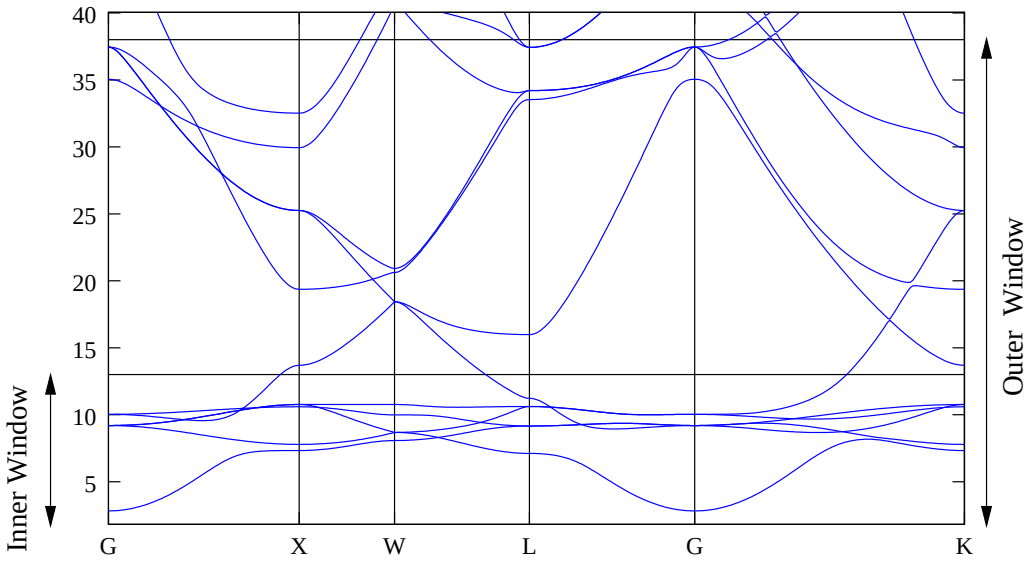

Figure 3: Band Structure of Copper showing the position of the outer and inner energy windows.

PWSCF Examples

A fully featured interface between Wannier90 and Pwscf will be available as part of the next Quantum Espresso release. Before that time it can be downloaded as part of the pwscf development distribution. Use the following commands to download a copy from the pwscf cvs server (bash shell):

```
export CVS_RSH=ssh
export CVSROOT=:pserver:cvsanon@democritos.sissa.it:/home/cvs
cvs login  (The password is: cvsanon)
cvs checkout pwscf
```

Diamond

- Outline: *Obtain MLWF for the occupied state of Diamond*
- Directory: `examples/example5/`
- Input Files
  - `diamond.scf` *Pwscf input file for ground state calculation*
  - `diamond.nscf` *Pwscf input file to obtain Bloch states on a uniform grid*
  - `diamond.pw2wan` *Input file for pw2wannier90*
  - `diamond.win` *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 kpoint 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 pw2wannier 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 Wannier functions.
   `wannier90.x diamond`

0.1 Still to write up

1. Silane
2. Iron (spin up channel)
3. Maybe MnO