Wannier90 User Guide

Version 1.0

30th April 2006

Chapter 1

Overview

1.1 Methodology

Wannier90 computes Maximally Localised Wannier Functions following the method of Marzari and Vanderbilt (MV).[1] For entangled energy bands the method of Souza, Marzari and Vanderbilt (SMV)[2] is used. We briefly introduce the methods and key definitions, full details can be found in the original papers.

First principles codes typically solve the electronic structure of periodic materials in terms of Bloch states, $\psi_{n\mathbf{k}}$. These extended states are characterised by a band index, n and crystal momentum, $\mathbf{k}$. An alternative representation can be given in terms of spatially localised functions known as Wannier functions (WF). The Wannier function centred on a lattice site $\mathbf{R}$, $w_{n\mathbf{R}}(\mathbf{r})$, is written in terms of the set of Bloch states as

$$w_{n\mathbf{R}}(\mathbf{r}) = \frac{V}{(2\pi)^3} \int_{BZ} \left[\sum_m U^{(\mathbf{k})}_{mn} \psi_{m\mathbf{k}}(\mathbf{r}) \right] e^{-\mathbf{k}.\mathbf{R}} d\mathbf{k}, \tag{1.1}$$

where $U^{(\mathbf{k})}$ is a unitary matrix which mixes the Bloch states at each $\mathbf{k}$. $U^{(\mathbf{k})}$ is not uniquely defined and different choices will lead to WF with varying spatial localisations. We define the spread of the Wannier functions, Ω

$$\Omega = \sum_n \left[\langle w_{n\mathbf{0}}(\mathbf{r}) | r^2 | w_{n\mathbf{0}}(\mathbf{r}) \rangle - |\langle w_{n\mathbf{0}}(\mathbf{r}) | \mathbf{r} | w_{n\mathbf{0}}(\mathbf{r}) \rangle|^2 \right]. \tag{1.2}$$

The total spread can be decomposed into a gauge invariant term, Ω_{I} and a term, $\tilde{\Omega}$ which is dependant on the gauge choice $U^{(\mathbf{k})}$. $\tilde{\Omega}$ can be further divided into terms diagonal and off-diagonal in the WF basis, Ω_{D} and Ω_{OD}.

$$\Omega = \Omega_{\mathrm{I}} + \tilde{\Omega} = \Omega_{\mathrm{I}} + \Omega_{\mathrm{D}} + \Omega_{\mathrm{OD}} \tag{1.3}$$

where

$$\Omega_{\mathrm{I}} = \sum_n \left[\langle w_{n\mathbf{0}}(\mathbf{r}) | r^2 | w_{n\mathbf{0}}(\mathbf{r}) \rangle - \sum_{\mathbf{R}m} |\langle w_{n\mathbf{R}}(\mathbf{r}) | \mathbf{r} | w_{n\mathbf{0}}(\mathbf{r}) \rangle|^2 \right] \tag{1.4}$$

[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)

$$\Omega_{\rm D} = \sum_n \sum_{\mathbf{R}\neq\mathbf{0}} |\langle w_{n\mathbf{R}}(\mathbf{r})|\mathbf{r}|w_{n\mathbf{0}}(\mathbf{r})\rangle|^2 \tag{1.5}$$

$$\Omega_{\rm OD} = \sum_{m\neq n} \sum_{\mathbf{R}} |\langle w_{m\mathbf{R}}(\mathbf{r})|\mathbf{r}|w_{n\mathbf{0}}(\mathbf{r})\rangle|^2 \tag{1.6}$$

The MV method minimises the gauge dependent spread $\tilde{\Omega}$, with respect the set of $U^{(\mathbf{k})}$ to obtain Maximally Localised Wannier Functions (MLWF).

The Wannier90 code requires two ingredients from an initial electronic structure calculation.

1. The overlaps between the cell periodic part of the bloch states $|u_{n\mathbf{k}}\rangle$

$$M_{mn\mathbf{k}} = \langle u_{m\mathbf{k}}|u_{n\mathbf{k}+\mathbf{b}}\rangle, \tag{1.7}$$

   where the vectors $\mathbf{b}$, which connect a given kpoint with its neighbours, are determined by the Wannier90 code according to prescription outlined in MV.

2. As a starting guess the projection of the bloch states onto trial localised orbitals

$$M_{mn\mathbf{k}} = \langle \psi_{m\mathbf{k}}|g_n\rangle, \tag{1.8}$$

Note that $M_{\mathbf{k}}$, $A_{\mathbf{k}}$ and $U^{(\mathbf{k})}$ are all small, $N_{\rm wann}^2$ matrices, independent of the basis set used to obtain the original Bloch states. To date Wannier90 has been used in combination with first principles codes based on planewaves and pseudopotentials (norm-conserving and "ultrasoft") as well as mixed basis set techniques such as FLAPW.

1.1.1 Entangled Energy Bands

The above description is sufficient to obtain Wannier functions for isolated set of bands, such as the valence states in an insulator. In order to obtain MLWF for entangled energy bands we use the "disentanglement" procedure introduced in SMV.

We define an energy window (the "outer window"). At a given kpoint $N_{\rm win}^{\mathbf{k}}$ states lie within this energy window. We obtain a set of $N_{\rm wann}$ Bloch states by performing a unitary[3] transformation amongst the Bloch states which fall within the energy window at each kpoint.

$$|u_{n\mathbf{k}}^{\rm opt}\rangle = \sum_{m\in N_{\rm win}^{\mathbf{k}}} U_{mn\mathbf{k}}^{\rm dis}|u_{m\mathbf{k}}\rangle \tag{1.9}$$

where $U_{\mathbf{k}}^{\rm dis}$ is a rectangular $N_{\rm wann}\times N_{\rm win}^{\mathbf{k}}$ matrix. The set of $U_{\mathbf{k}}^{\rm dis}$ are obtained by minimising the gauge invariant spread, $\Omega_{\rm I}$ within the outer energy window. The MV procedure can then be used to minimise $\tilde{\Omega}$ and hence obtain MLWF for this optimal subspace.

It should be noted that the energy bands of this optimally subspace may not correspond to any of the original energy bands (due to mixing between states). In order to preserve exactly the properties of a system in a given energy range (eg around the Fermi level) we introduce a second energy window. States lying within this inner, or "frozen" energy window are included unchanged in the optimal subspace.

[3] As $U_{\mathbf{k}}^{\rm dis}$ is a rectangular matrix this is a unitary operation in the sense that $(U_{\mathbf{k}}^{\rm dis})^\dagger U_{\mathbf{k}}^{\rm dis} = 1$.

1.1.2 Citation

We ask that you acknowledge the use of Wannier90 in any publications arising from the use of this code through the following reference

> [ref] A. A. Mostofi, J. R. Yates, N. Marzari, I. Souza and D. Vanderbilt,
> http://www.wannier.org/

It will also be appropriate to cite the original articles:

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

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

1.1.3 Credits

The present release of Wannier90 was written by Arash Mostofi (Marzari group @ MIT) and Jonathan Yates (Souza Group @ UCB/LBNL). Wannier90 is based on routines written in 1996-7 for occupied bands by Nicola Marzari and David Vanderbilt and for entangled bands by Ivo Souza, Nicola Marzari, and David Vanderbilt in 2000-1.

Acknowledgements: ?

1.1.4 Licence

Chapter 2

Parameters

2.1 Usage

```
wannier90.x [-pp] [seedname]
```

- `seedname` If a seedname string is given the code will read its input from a file `seedname.win`. The default value is `wannier`.
- `-pp` This optional flag tells the code to generate a list of the required overlaps and then exit. This information is written to the file `seedname.nnkp`.

2.2 win File

The Wannier90 input file `seedname.win` has a flexible free-form structure.

The ordering of the keywords is not significant. Case is ignored (so `num_bands` is the same as `Num_Bands`). Characters after !, or # are treated as comments. Most keywords have a default value which is used unless the keyword is given in the win file. Keywords can be set in any of the following ways

```
num_wann 4

num_wann = 4

num_wann :  4
```

A logical keyword can be set to `.true.` using any of the following strings: `T`, `true`, `.true.`.

For futher examples see Chapter 8.1 and the the Wannier90 Tutorial.

2.3 Keyword List

Keyword	Type	Description
System Parameters		
NUM_WANN	I	Number of Wannier Functions
NUM_BANDS	I	Number of bands passed to the code
UNIT_CELL_CART	P	Unit cell vectors
ATOMS_CART *	P	Positions of atoms in Cartesian coordinates
ATOMS_FRAC *	R	Positions of atoms in lattice vectors
MP_GRID	I	Dimensions of the Monkhorst-Pack grid
KPOINTS	R	List of kpoints in the Monkhorst-Pack grid
NUM_SHELLS	I	Number of shells in finite difference formula
SHELL_LIST	I	Which shells to use in finite difference formula

Table 2.1: win file keywords defining the system. Argument types are represented by, I for a integer, R for a real number, P for a physical value, L for a logical value and S for a text string.

* ATOMS_CART and ATOMS_FRAC may not both be defined in the same input file.

Keyword	Type	Description
Job Control		
POSTPROC_SETUP	L	To output the nnkp file
CP_PP	L	CP code post-processing
CALC_ONLY_A	L	Only recalculate the projections
EXCLUDE_BANDS	I	List of bands to exclude from the calculation
RESTART	C	Restart from checkpoint file
IPRINT	I	Output verbosity level
LENGTH_UNIT	S	System of units to output lengths
WVFN_FORMATED	L	Read the wavefunctions from a (un)formatted file
SPIN	S	Which spin channel to read
DEVEL_FLAG	S	Flag for development use

Table 2.2: win file keywords defining the system. Argument types are represented by, I for a integer, R for a real number, P for a physical value, L for a logical value and S for a text string.

Keyword	Type	Description
Disentanglement Parameters		
DIS_WIN_MIN	P	Bottom of the outer energy window
DIS_WIN_MAX	P	Top of the outer energy window
DIS_FROZ_MIN	P	Bottom of the inner (frozen) energy window
DIS_FROZ_MAX	P	Top of the inner (frozen) energy window
DIS_NUM_ITER	I	Number of iterations for the minimisation of Ω_I
DIS_MIX_RATIO	R	Mixing ratio during the minimisation of Ω_I
DIS_CONV_TOL	R	The convergence tolerance for finding Ω_I
DIS_CONV_WINDOW	I	The number of iterations over which convergence of Ω_I is assessed.

Table 2.3: win file keywords controlling the disentanglement. Argument types are represented by, I for a integer, R for a real number, P for a physical value, L for a logical value and S for a text string.

Keyword	Type	Description
Wannierise Parameters		
NUM_ITER	I	Number of iterations for the minimisation of Ω
NUM_CG_STEPS	I	During the minimisation of Ω the number of Conjugate Gradient steps before resetting to Steepest Descents
CONV_TOL	P	The convergence tolerance for finding Ω
CONV_WINDOW	I	The number of iterations over which convergence of Ω is assessed
NUM_DUMP_CYCLES	I	Control frequency of check-pointing
NUM_PRINT_CYCLES	I	Control frequency of printing
WRITE_R2MN	L	Write matrix elements of r^2 between Wannier functions to file
GUIDING_CENTRES	L	Use guiding centres
NUM_GUIDE_CYCLES	I	Frequency of guiding centres
NUM_NO_GUIDE_ITER	I	The number of iterations after which guiding centres are used

Table 2.4: win file keywords controlling the wannierisation. Argument types are represented by, I for a integer, R for a real number, P for a physical value, L for a logical value and S for a text string.

Keyword	Type	Description
Plot Parameters		
WANNIER_PLOT	L	Plot the Wannier Functions
WANNIER_PLOT_LIST	I	List of Wannier Functions to plot
WANNIER_PLOT_SUPERCELL	I	Size of the supercell for plotting the Wannier Functions
WANNIER_PLOT_FORMAT	S	File format in which to plot the Wannier Functions
WANNIER_PLOT_MODE	S	Mode in which to plot the Wannier Functions, molecule or crystal
BANDS_PLOT	L	Plot and interpolated band structure
KPOINT_PATH	P	K-point path for the interpolated band structure
BANDS_NUM_POINTS	I	Number of points along the first section of the k-point path
BANDS_PLOT_FORMAT	S	File format in which to plot the interpolated bands
FERMI_SURFACE_PLOT	L	Plot the Fermi surface
FERMI_SURFACE_NUM_POINTS	I	Number of points in the Fermi surface plot
FERMI_ENERGY	P	The Fermi energy
FERMI_SURFACE_PLOT_FORMAT	S	File format for the Fermi surface plot

Table 2.5: win file keywords controlling the plotting. Argument types are represented by, I for a integer, R for a real number, P for a physical value, L for a logical value and S for a text string.

2.4 System

2.4.1 integer :: num_wann

Number of wannier functions to be found.

No default.

2.4.2 integer :: num_bands

Total number of bands passed to the code in the `<seedname>.mmn` file.

Default `num_bands=num_wann`

2.4.3 Cell Lattice Vectors

The cell lattice vectors should be specified in Cartesian coordinates.

`begin unit_cell_cart`
`[units]`

$$\begin{array}{ccc} R_{1x} & R_{1y} & R_{1z} \\ R_{2x} & R_{2y} & R_{2z} \\ R_{3x} & R_{3y} & R_{3z} \end{array}$$

`end unit_cell_cart`

Here R_{1x} is the x-component of the first lattice vector, R_{2y} is the y-component of the second lattice vector etc.

`[units]` specifies the units in which the lattice vectors are defined either `bohr` or `ang`. If not present, the default is Å.

There is no default.

2.4.4 Ionic Positions

The ionic positions may be specified in fractional coordinates relative to the lattice vectors of the unit cell, or in absolute cartesian coordinates. Only one of `atoms_cart` and `atoms_frac` may be given in the input file.

atoms_cart

`begin atoms_cart`
`[units]`

$$\begin{array}{cccc} X & R_{1i} & R_{1j} & R_{1k} \\ Y & R_{2i} & R_{2j} & R_{2k} \\ \vdots & & & \end{array}$$

`end atoms_cart`

The first entry on a line is the atomic symbol. The next three entries are the atom's position in Cartesian coordinates in units specified by `length_unit`.

`[units]` specifies the units in which the lattice vectors are defined either `bohr` or `ang`. If not present, the default is Å.

atoms_frac

`begin atoms_frac`

$$\begin{array}{cccc} X & R_{1i} & R_{1j} & R_{1k} \\ Y & R_{2i} & R_{2j} & R_{2k} \\ \vdots & & & \end{array}$$

`end atoms_frac`

The first entry on a line is the Atomic symbol. The next three entries are the atom's position in fractional coordinates.

2.4.5 `integer, dimension :: mp_grid(3)`

Dimensions of the regular (Monkhorst-Pack) kpoint mesh.

No default.

2.4.6 Kpoints

Each line gives the coordinate of a k-point in relative units, i.e. in units of the reciprocal lattice vectors. The position of each k-point in this list assigns its numbering; the first k-point is k-point 1, the second is k-point 2, and so on.

`begin kpoints`

$$\begin{array}{ccc} R_{1i} & R_{1j} & R_{1k} \\ R_{2i} & R_{2j} & R_{2k} \\ \vdots & & \end{array}$$

`end kpoints`

There is no default.

2.4.7 Shells

The Marzari-Vanderbilt scheme requires a finite difference expression for ∇_k defined on a uniform Monkhorst-Pack mesh of kpoints. One choice (the 'B1' condition of MV) is to choose shells of kpoint neighbours to satisfy the equation

$$\sum_s w_s \sum_i b^s_{i\alpha} b^s_{i\beta} = \delta_{\alpha\beta} \tag{2.1}$$

's' is sum over shells, 'i' is a sum over kpoints in that shell, w_s is a weight factor for the shell s, $\mathbf{b}_i^s$ is a vector connecting a kpoint to one of it nearest neighbours, α and β are Cartesian coordinates.

2.4.8 `integer :: num_shells`

If `num_shells`>0 the number of shells to include in the finite difference expression. If `num_shells`$=0$ the code will choose the shells automatically.

The default value is 0.

2.4.9 `integer :: shell_list(num_shells)`

If `num_shells`>0 `shell_list` is vector listing the shells to include in the finite difference expression.

2.5 Projection

The projections block defines a set of localised functions used to generate an initial guess for the unitary transformations. This data will be written in the `<seedname>.nnkp` file.

```
begin projections

end projections
```

For details see section 3.1.

2.6 Job Control

2.6.1 `logical :: postproc_setup`

If `postproc_setup`=`TRUE` then the wannier code will write `<seedname>.nnkp` file and exit. If Wannier90 is called with the option `-pp`; `postproc_setup` is set to `TRUE`, over-riding its value in the `<seedname>.win` file.

The default value is `FALSE`.

2.6.2 `logical :: cp_pp`

If `cp_pp`=`TRUE` we are using input files from the CP code.

The default value is `FALSE`.

2.6.3 `integer :: iprint`

This indicates the level of verbosity of the output from 0, the bare minimum to 3, which corresponds to full debugging output.

The default value is 1.

2.6.4 `character(len=20) :: length_unit`

The length unit to be used for output.

The valid options for this parameter are:

– Ang (default)

– Bohr

2.6.5 `character(len=50) :: devel_flag`

Not a regular keyword. Its purpose is to allow a developer to pass a string into the code to be used inside a new routine as it is developed.

No default.

2.6.6 `logical :: calc_only_A`

Not yet implemented

If `calc_only_A` = `.true.`, then the *ab initio* code, eg PWSCF, calculates only $A_{mn}^{(\mathbf{k})}$. Otherwise, both $M_{mn}^{(\mathbf{k},\mathbf{b})}$ and $A_{mn}^{(\mathbf{k})}$ are calculated.

The default value of this parameter is `FALSE`.

2.6.7 `integer :: exclude_bands(:)`

A kpoint independent list of states to excluded from the calculation of the overlap matrices; for example to select only valence states, or ignore semi-core states. This keyword is passed to the first-principles code via the `<seedname>.nnkp` file.

2.6.8 `character(len=20) :: restart`

If `restart` is present the code will attempt to restart the calculation from the `<seedname>.chk` file. The value of the parameter determines the position of the restart

The valid options for this parameter are:

– `default`. Restart from the point at which the check file was written

– `wannierise`. Restart from the beginning of the wannierise routine

– `plot`. Go directly to the plotting phase

2.6.9 `character(len=20) :: wvfn_formatted`

If `wvfn_formatted`=`TRUE` the wavefunctions will be read from disk as formatted (ie ASCII) files. Otherwise they will be read as unformatted files. Unformatted in generally preferable as the files will take less disk space I/O is significantly faster. However such files will not be transferable between all machine architectures and formatted files should be used if transferability is required (ie for test cases).

The default value of this parameter is `FALSE`.

2.6.10 `character(len=20) :: spin`

For bands from a spin polarised calculation determines which set of bands to read in, either 'up' or 'down'.

The default value of this parameter is 'up'.

2.7 Disentanglement

These keywords control the disentanglement routine of SMV. This routine will be activated if `num_wann`<`num_bands`.

2.7.1 `real(kind=dp) :: dis_win_min`

The lower bound of the outer energy window for the disentanglement procedure.

The default is the lowest eigenvalue in the system.

2.7.2 `real(kind=dp) :: dis_win_max`

The upper bound of the outer energy window for the disentanglement procedure.

The default is the highest eigenvalue in the given states (ie all states are included in the disentanglement procedure).

2.7.3 `real(kind=dp) :: dis_froz_min`

The lower bound of the inner energy window for the disentanglement procedure.

If `dis_froz_max` is given the default for `dis_froz_min` is `dis_win_min`.

2.7.4 `real(kind=dp) :: dis_froz_max`

The upper bound of the inner energy window for the disentanglement procedure. If `dis_froz_max` is not specified then there are no frozen states.

No default.

2.7.5 `integer :: dis_num_iter`

In the disentanglement procedure, the number of iterations used to extract the most connected subspace.

The default value is 100.

2.7.6 `real(kind=dp) :: dis_mix_ratio`

In the disentanglement procedure the mixing parameter to use for convergence.

The default value is 1.0

2.7.7 `real(kind=dp) :: dis_conv_tol`

In the disentanglement procedure the minimisation is said to to converged if the fractional change in the spread between successive iterations is less than `dis_conv_tol` for `dis_conv_window` iterations.

The default value is 1.0E-10

2.7.8 `integer :: dis_conv_window`

In the disentanglement procedure the minimisation is said to to converged if the fractional change in the spread between successive iterations is less than `dis_conv_tol` for `dis_conv_window` iterations.

The default value of this parameter is 3.

2.8 Wannierise

Minimise the non-invariant part of the spread functional.

2.8.1 `integer :: num_iter`

Total number of iterations in the minimisation procedure.

The default value is 100.

2.8.2 `integer :: num_cg_steps`

Number of conjugate gradient steps to take before resetting to steepest descents.

The default value is 5.

2.8.3 `integer :: num_dump_cycles`

Write sufficient information to do a restart every `num_dump_cycles` iterations.

The default is 0 (ie don't write out any restart information).

2.8.4 `integer :: num_print_cycles`

Write data to the `<seedname>.wout` file every `num_print_cycles` iterations.

The default is 1.

2.8.5 `logical :: write_r2mn`

If `write_r2mn` = `true`, then the matrix elements $< m|r^2|n >$ (where m and n refer to Wannier functions) are written to file `seedname.r2mn` at the end of the wannierisation procedure.

The default value of this parameter is `FALSE`.

2.8.6 `logical :: guiding_centres`

Use guiding centres during the minimisation, in order to avoid local minima.

The default value is `FALSE`.

2.8.7 `integer :: num_guide_cycles`

If `guiding_centres` is set to true the guiding centres are used only every `num_guide_cycles`.

The default value is 1.

2.8.8 `integer :: num_no_guide_iter`

If `guiding_centres` is set to true the guiding centres are used only after `num_no_guide_iter` minimisation iterations have been completed.

The default value is 0.

2.9 Post-Processing

Capabilities:

- Plot the Wannier functions
- Plot the interpolated band structure
- Plot the Fermi surface

2.9.1 `logical :: wannier_plot`

If `wannier_plot` = `TRUE` the code will write out the wannier functions in a super-cell `wannier_plot_supercell` times the original unit cell in a format specified by `wannier_plot_format`

The default value of this parameter is `FALSE`.

2.9.2 `integer :: wannier_plot_supercell`

Dimension of the "super-unit-cell" in which the Wannier Functions are plotted. The super-unit-cell is `wannier_plot_supercell` times the unit cell along all three linear dimensions (the 'home' unit cell is kept approximately in the middle)

The default value is 2.

2.9.3 `character(len=20) :: wannier_plot_format`

The valid options for this parameter are:

- xcrysden (default)

2.9.4 `integer :: wannier_plot_list(:)`

A list of Wannier Functions to plot. The Wannier Functions numbered as per the `<seedname>.wout` file after the minimisation of the spread.

The default behaviour is to plot all Wannier Functions.

2.9.5 `character(len=20) :: wannier_plot_mode`

Choose the mode in which to plot the Wannier functions, either as a molecule or as a crystal.

The valid options for this parameter are:

- crystal (default)
- molecule

2.9.6 `logical :: bands_plot`

If `bands_plot` = `TRUE` the code will calculate the band structure, through Wannier interpolation, along the path in k-space defined by `bands_kpath` using `bands_num_points` along the first section of the path and write out an output file in a format specified by `bands_plot_format`.

The default value is `FALSE`.

2.9.7 kpoint_path

Defines the path in kspace along which to calculate the bandstructure. Each lines gives the start and end points (with labels) for a section of the path.

`begin kpoint_path`

$$\begin{array}{cccccccc} G & 0.0 & 0.0 & 0.0 & L & 0.0 & 0.0 & 1.0 \\ L & 0.0 & 0.0 & 1.0 & N & 0.0 & 1.0 & 1.0 \\ \vdots & & & & & & & \end{array}$$

`end kpoint_path`

There is no default

2.9.8 `integer :: bands_num_points`

If `bands_plot` = `TRUE` the number of points along the first section of the bandstructure plot given by `kpoint_path`. Other sections will have the same density of kpoints.

The default value for `bands_num_points` is 100.

2.9.9 `character(len=20) :: bands_plot_format`

Format in which to plot the interpolated band structure The valid options for this parameter are:

- gnuplot (default)

2.9.10 `logical :: fermi_surface_plot`

If `fermi_surface_plot` = `TRUE` the code will calculate, through Wannier interpolation, the eigenvalues on a regular grid with `fermi_surface_num_points` in each direction. The code will write a file in bxsf format which can be read with Xcrysden and used to plot the Fermi surface.

The default value is `FALSE`.

2.9.11 integer :: fermi_surface_num_points

If fermi_surface_plot = TRUE the number of divisions in the regular kpoint grid used to calculate the Fermi surface.

The default value for fermi_surface_num_points is 50.

2.9.12 real(kind=dp) :: fermi_energy

The Fermi energy eV. Whilst this is not directly used by the Wannier code is a useful parameter to set for Fermi surface plots as it will be written into the bxsf file.

The default value is 0.0eV.

2.9.13 character(len=20) :: fermi_surface_plot_format

Format in which to plot the Fermi surface. The valid options for this parameter are:

- xcrysden (default)

Chapter 3

Projections

3.1 Specification of projections in seedname.win

Here we describe the projection functions used to construct the initial guess $A_{mn}^{(\mathbf{k})}$ for the unitary transformations.

Each projection is associated with a site and an angular momentum state defining the projection function. Optionally, one may define, for each projection, the spatial orientation, the radial part, the diffusivity, and the volume over which real-space overlaps A_{mn} are calculated.

We would like to be able to

1. project onto s,p,d and f angular momentum states, plus the hybrids sp, sp^2, sp^3, sp^{3}d, sp^{3}d^2.
2. control the radial part of the projection functions to allow higher angular momentum states, e.g., both 3s and 4s in silicon.

The atomic orbitals of the hydrogen atom provide a good basis to use for constructing the projection functions: analytical mathematical forms exist in terms of the good quantum numbers n, l and m; hybrid orbitals (sp, sp^2, sp^3, sp^{3}d etc.) can be constructed by simple linear combination $|\phi\rangle = \sum_{nlm} C_{nlm}|nlm\rangle$ for some coefficients C_{nlm}.

The angular functions that we want to use as a basis for the projections are not the canonical spherical harmonics Y_{lm} of the hydrogenic Schrödinger equation but rather the *real* (in the sense of non-imaginary) states $\Theta_{lm_{\mathrm{r}}}$, obtained by a unitary transformation. For example, the canonical eigenstates associated with $l = 1$, $m = \{-1, 0, 1\}$ are not the real p$_x$, p$_y$ and p$_z$ that we want. See Section 3.2 for our mathematical conventions regarding projection orbitals for different n, l and m_{r}.

We use the following format to specify projections in `<seedname>.win`:

```
Begin Projections
units
site:ang_mtm:zaxis:xaxis:radial:zona:box-size
    ⋮
```

`End Projections`

Notes:

`units`:
Optional. Either `Ang` or `Bohr` to specify whether the projection centres specified in this block (if given in Cartesian co-ordinates) are in units of Angstrom or Bohr, respectively. Default is `Ang`.

`site`:
`C`, `Al`, etc. applies to all atoms of that type
`f=0,0.50,0` – centre on (0.0,0.5,0.0) in **f**ractional coordinates (crystallographic units) relative to the direct lattice vectors
`c=0.0,0.805,0.0` – centre on (0.0,0.805,0.0) in **C**artesian coordinates in units specified by the optional string `units` in the first line of the projections block (default is Angstrom).

`ang_mtm`:
Angular momentum states may be specified by `l` and `mr`, or by the appropriate character string. See Tables 3.1 and 3.2. Examples:
`l=2,mr=1` or `dz2` – a single projection with $l = 2$, $m_{\rm r} = 1$ (i.e., d_{z^2})
`l=2,mr=1,4` or `dz2,dx2-y2` – two functions: d_{z^2} and d_{xz}
`l=-3` or `sp3` – four sp^3 hybrids
Specific hybrid orbitals may be specified as follows:
`l=-3,mr=1,3` or `sp3-1,sp3-3` – two specific sp^3 hybrids
Multiple states may be specified by separating with '`;`', e.g.,
`sp3;l=0` or `l=-3;l=0` – four sp^3 hybrids and one s orbital

`zaxis` (optional):
`z=1,1,1` – set the z-axis to be in the (1,1,1) direction. Default is `z=0,0,1`

`xaxis` (optional):
`x=1,1,1` – set the x-axis to be in the (1,1,1) direction. Default is `x=1,0,0`

`radial` (optional):
`r=2` – use a radial function with one node (ie second highest pseudostate with that angular momentum). Default is `r=1`. Radial functions associated with different values of `r` should be orthogonal to each other.

`zona` (optional):
`zona=2.0` – the value of $\frac{Z}{a}$ for the radial part of the atomic orbital (controls the diffusivity of the radial function). Units always in reciprocal Angstrom. Default is `zona=1.0`.

`box-size` (optional):
`b=2.0` – the linear dimension of the real-space box (or sphere) for calculating the overlap $\langle\psi_{m\mathbf{k}}|\phi_n\rangle$ of a wavefunction with the localised projection function. Units are always in Angstrom. Default is `b=1.0`

Examples

1. CuO, s,p and d on all Cu; sp^3 hybrids on O:

`Cu:l=0;l=1;l=2`

`O:l=-3` or `O:sp3`

2. A single projection onto a p_z orbital orientated in the (1,1,1) direction:

`c=0,0,0:l=1,mr=1:z=1,1,1` or `c=0,0,0:pz:z=1,1,1`

3. Project onto s, p and d (with no radial nodes), and s and p (with one radial node) in silicon:

`Si:l=0;l=1;l=2`

`Si:l=0;l=1:r=2`

3.2 Orbital Definitions

The angular functions $\Theta_{l m_{\mathrm{r}}}(\theta, \varphi)$ associated with particular values of `l` and `mr` are given in Tables 3.1 and 3.2.

The radial functions $R_{\mathrm{r}}(r)$ associated with different values of `r` should be orthogonal. One choice would be to take the set of solutions to the radial part of the hydrogenic Schrödinger equation for $l = 0$, i.e., the radial parts of the 1s, 2s, 3s… orbitals, which are given in Table 3.3.

l	mr	Name	$\Theta_{lm_\mathrm{r}}(\theta,\varphi)$
0	1	s	$\frac{1}{\sqrt{4\pi}}$
1	1	pz	$\sqrt{\frac{3}{4\pi}}\cos\theta$
1	2	px	$\sqrt{\frac{3}{4\pi}}\sin\theta\cos\varphi$
1	3	py	$\sqrt{\frac{3}{4\pi}}\sin\theta\sin\varphi$
2	1	dz2	$\sqrt{\frac{5}{16\pi}}(3\cos^2\theta-1)$
2	2	dxz	$\sqrt{\frac{15}{4\pi}}\sin\theta\cos\theta\cos\varphi$
2	3	dyz	$\sqrt{\frac{15}{4\pi}}\sin\theta\cos\theta\sin\varphi$
2	4	dx2-y2	$\sqrt{\frac{15}{16\pi}}\sin^2\theta\cos 2\varphi$
2	5	dxy	$\sqrt{\frac{15}{16\pi}}\sin^2\theta\sin 2\varphi$
3	1	fz3	$\frac{\sqrt{7}}{4\sqrt{\pi}}(5\cos^3\theta-3\cos\theta)$
3	2	fxz2	$\frac{\sqrt{21}}{4\sqrt{2\pi}}(5\cos^2\theta-1)\sin\theta\cos\varphi$
3	3	fyz2	$\frac{\sqrt{21}}{4\sqrt{2\pi}}(5\cos^2\theta-1)\sin\theta\sin\varphi$
3	4	fz(x2-y2)	$\frac{\sqrt{105}}{4\sqrt{\pi}}\sin^2\theta\cos\theta\cos 2\varphi$
3	5	fxyz	$\frac{\sqrt{105}}{4\sqrt{\pi}}\sin^2\theta\cos\theta\sin 2\varphi$
3	6	fx(x2-3y2)	$\frac{\sqrt{35}}{4\sqrt{2\pi}}\sin^3\theta(\cos^2\varphi-3\sin^2\varphi)\cos\varphi$
3	7	fy(3x2-y2)	$\frac{\sqrt{35}}{4\sqrt{2\pi}}\sin^3\theta(3\cos^2\varphi-\sin^2\varphi)\sin\varphi$

Table 3.1: Angular functions $\Theta_{lm_\mathrm{r}}(\theta,\varphi)$ associated with particular values of l and mr for l≥ 0.

l	mr	Name	$\Theta_{lm_r}(\theta,\varphi)$
-1	1	sp-1	$\frac{1}{\sqrt{2}}\texttt{s} + \frac{1}{\sqrt{2}}\texttt{px}$
-1	2	sp-2	$\frac{1}{\sqrt{2}}\texttt{s} - \frac{1}{\sqrt{2}}\texttt{px}$
-2	1	sp2-1	$\frac{1}{\sqrt{3}}\texttt{s} - \frac{1}{\sqrt{6}}\texttt{px} + \frac{1}{\sqrt{2}}\texttt{py}$
-2	2	sp2-2	$\frac{1}{\sqrt{3}}\texttt{s} - \frac{1}{\sqrt{6}}\texttt{px} - \frac{1}{\sqrt{2}}\texttt{py}$
-2	3	sp2-3	$\frac{1}{\sqrt{3}}\texttt{s} + \frac{2}{\sqrt{6}}\texttt{px}$
-3	1	sp3-1	$\frac{1}{2}(\texttt{s} + \texttt{px} + \texttt{py} + \texttt{pz})$
-3	2	sp3-2	$\frac{1}{2}(\texttt{s} + \texttt{px} - \texttt{py} - \texttt{pz})$
-3	3	sp3-3	$\frac{1}{2}(\texttt{s} - \texttt{px} + \texttt{py} - \texttt{pz})$
-3	4	sp3-4	$\frac{1}{2}(\texttt{s} - \texttt{px} - \texttt{py} + \texttt{pz})$
-4	1	sp3d-1	$\frac{1}{\sqrt{3}}\texttt{s} - \frac{1}{\sqrt{6}}\texttt{px} + \frac{1}{\sqrt{2}}\texttt{py}$
-4	2	sp3d-2	$\frac{1}{\sqrt{3}}\texttt{s} - \frac{1}{\sqrt{6}}\texttt{px} - \frac{1}{\sqrt{2}}\texttt{py}$
-4	3	sp3d-3	$\frac{1}{\sqrt{3}}\texttt{s} + \frac{2}{\sqrt{6}}\texttt{px}$
-4	4	sp3d-4	$\frac{1}{\sqrt{2}}\texttt{pz} + \frac{1}{\sqrt{2}}\texttt{dz2}$
-4	5	sp3d-5	$-\frac{1}{\sqrt{2}}\texttt{pz} + \frac{1}{\sqrt{2}}\texttt{dz2}$
-5	1	sp3d2-1	$\frac{1}{\sqrt{6}}\texttt{s} - \frac{1}{\sqrt{2}}\texttt{px} - \frac{1}{\sqrt{12}}\texttt{dz2} + \frac{1}{2}\texttt{dx2-y2}$
-5	2	sp3d2-2	$\frac{1}{\sqrt{6}}\texttt{s} + \frac{1}{\sqrt{2}}\texttt{px} - \frac{1}{\sqrt{12}}\texttt{dz2} + \frac{1}{2}\texttt{dx2-y2}$
-5	3	sp3d2-3	$\frac{1}{\sqrt{6}}\texttt{s} - \frac{1}{\sqrt{2}}\texttt{py} - \frac{1}{\sqrt{12}}\texttt{dz2} - \frac{1}{2}\texttt{dx2-y2}$
-5	4	sp3d2-4	$\frac{1}{\sqrt{6}}\texttt{s} + \frac{1}{\sqrt{2}}\texttt{py} - \frac{1}{\sqrt{12}}\texttt{dz2} - \frac{1}{2}\texttt{dx2-y2}$
-5	5	sp3d2-5	$\frac{1}{\sqrt{6}}\texttt{s} - \frac{1}{\sqrt{2}}\texttt{pz} + \frac{1}{\sqrt{3}}\texttt{dz2}$
-5	6	sp3d2-6	$\frac{1}{\sqrt{6}}\texttt{s} + \frac{1}{\sqrt{2}}\texttt{pz} + \frac{1}{\sqrt{3}}\texttt{dz2}$

Table 3.2: Angular functions $\Theta_{lm_r}(\theta,\varphi)$ associated with particular values of l and mr for l<0, in terms of the orbitals defined in Table 3.1.

r	$R_{\mathrm{r}}(r)$
1	$2\alpha^{3/2}\exp(-\alpha r)$
2	$\frac{1}{2\sqrt{2}}\alpha^{3/2}(2-\alpha r)\exp(-\alpha r/2)$
3	$\sqrt{\frac{4}{27}}\alpha^{3/2}(1-2\alpha r/3+2\alpha^2 r^2/27)\exp(-\alpha r/3)$

Table 3.3: One possible choice for the radial functions $R_{\mathrm{r}}(r)$ associated with different values of r: the set of solutions to the radial part of the hydrogenic Schrödinger equation for $l=0$, i.e., the radial parts of the 1s, 2s, 3s... orbitals, where $\alpha = Z/a =$ zona.

Chapter 4

Overview

The wannier90 code can operate in two modes

(1) Read in the overlaps and projections from file as computed inside an ab-initio code. We expect this to be the most common route to using the Wannier code.

(2) As a set of library routines to be called from within an ab-initio code. The ab-initio code passes the overlaps and projections to the wannier library routines and in return gets the unitary transformation corresponding to maximally localised wannier functions. This route should be used if the Wannier functions are needed within the ab-initio code, for example in post-LDA methods such as LDA+U or SIC. (this mode is under development)

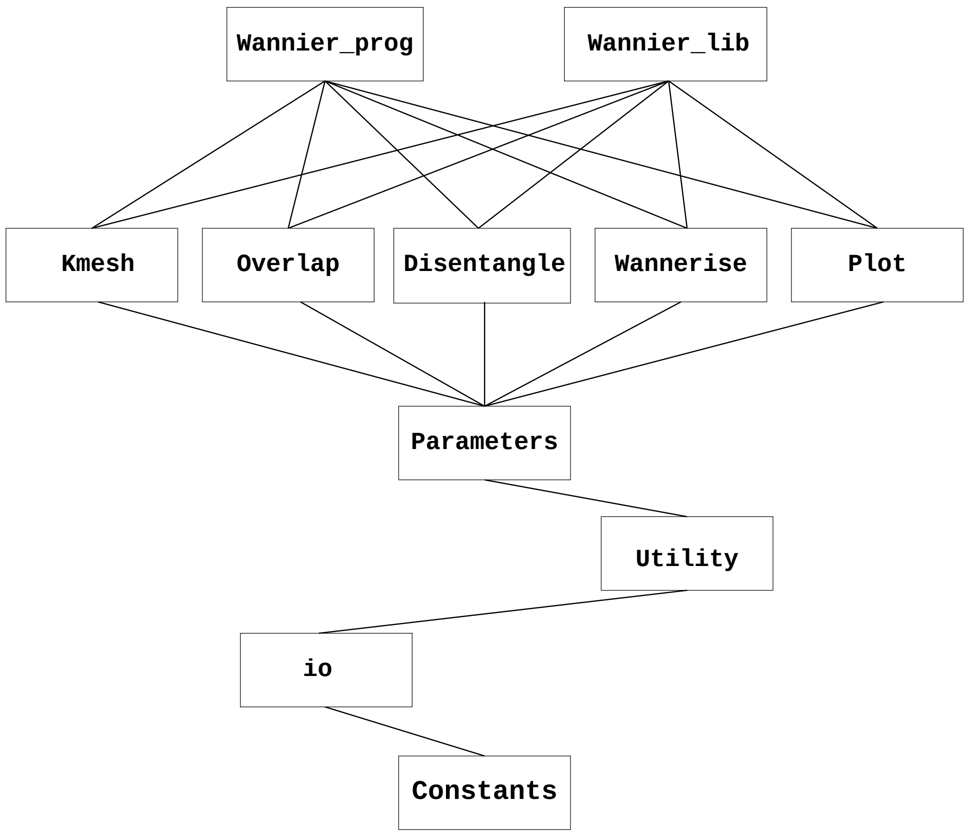

Figure 4.1: Schematic overview of the module structure of the wannier90 code. Modules may only use data and subroutines from lower modules.

Chapter 5

Wannier as a post-processing tool

This is a description of how to use Wannier90 code as a post-processing tool.

The code must be run twice. On the first pass the logical keyword `postproc_setup` must be set to `.true.` in the input file `seedname.win`. Running the code then generates the file `seedname.nnkp` which provides the information required to construct the $M_{mn}^{(\mathbf{k},\mathbf{b})}$ overlaps (MV[1] Eq. (25)) and $A_{mn}^{(\mathbf{k})} = \langle\psi_{m\mathbf{k}}|g_n\rangle$ projections (MV Eq. (62), SMV[2] Eq. (22)).

Once the overlaps and projection have been computed and written to files `seedname.mmn` and `seedname.amn`, respectively, set `postproc_setup` to `.false.` and run the code. Output is written to the file `seedname.wout`.

5.1 nnkp file

OUTPUT, if `postproc_setup` = `.true.`

The file `seedname.nnkp` provides the information needed to determine the required overlap elements $M_{mn}^{(\mathbf{k},\mathbf{b})}$ (MV Eq. (25)) and projections $A_{mn}^{(\mathbf{k})}$ (MV Eq. (62), SMV Eq. (22)), ie, `M_matrix_orig` and `A_matrix`, described in Section 6.1.2.

Much of the information in `seedname.nnkp` is arranged in blocks delimited by the strings `begin block_name` ... `end block_name`, as described below.

5.1.1 Keywords

The first line of the file is a user comment, e.g., the date and time:

```
File written on 12Feb2006 at 15:13:12
```

The only logical keyword is `calc_only_A`, eg,

```
calc_only_A  :  F
```

[1]Marzari and Vanderbilt, *Phys. Rev. B* **56**, 12847 (1997)
[2]Souza, Marzari and Vanderbilt, *Phys. Rev. B* **65**, 035109 (2001)

5.1.2 Real_lattice block

```
begin real_lattice
 2.250000   0.000000   0.000000
 0.000000   2.250000   0.000000
 0.000000   0.000000   2.250000
end real_lattice
```

The real lattice vectors in units of Angstrom.

5.1.3 Recip_lattice block

```
begin recip_lattice
 2.792527   0.000000   0.000000
 0.000000   2.792527   0.000000
 0.000000   0.000000   2.792527
end recip_lattice
```

The reciprocal lattice vectors in units of inverse Angstrom.

5.1.4 Kpoints block

```
begin kpoints
  8
  0.00000   0.00000   0.00000
  0.00000   0.50000   0.00000
  .
  .
  .
  0.50000   0.50000   0.50000
end kpoints
```

The first line in the block is the total number of k-points `num_kpts`. The subsequent `num_kpts` lines specify the k-points in crystallographic co-ordinates relative to the reciprocal lattice vectors.

5.1.5 Projections block

```
begin projections
   n_proj
   centre   l  mr  r
     z-axis   x-axis   zona   box-size
   centre   l  mr  r
     z-axis   x-axis   zona   box-size
   .
```

```
    .
end projections
```

Notes:

`n_proj`: integer; the number of projection centres, equal to the number of Wannier functions `num_wann`.

`centre`: three real numbers; projection function centre in crystallographic co-ordinates relative to the direct lattice vectors.

`l mr r`: three integers; l and m_r specify the angular part $\Theta_{lm_\mathrm{r}}(\theta,\varphi)$, and r specifies the radial part $R_\mathrm{r}(r)$ of the projection function (see Tables 3.1, 3.2 and 3.3).

`z-axis`: three real numbers; default is `0.0 0.0 1.0`; defines the axis from which the polar angle θ in spherical polar coordinates is measured.

`x-axis`: three real numbers; must be orthogonal to `z-axis`; default is `1.0 0.0 0.0` or a vector perpendicular to `z-axis` if `z-axis` is given; defines the axis from with the azimuthal angle φ in spherical polar coordinates is measured.

`zona`: real number; the value of $\frac{Z}{a}$ associated with the radial part of the atomic orbital. Units are in reciprocal Angstrom.

`box-size`: real number; the linear dimension of the real-space box (or sphere) for calculating the overlap $\langle\psi_{m\mathbf{k}}|\phi_n\rangle$ of a wavefunction with the localised projection function. Units are in Angstrom.

5.1.6 nnkpts block

```
begin nnkpts
  10
  1   2   0  0  0
  .
  .
end nnkpts
```

First line: `nntot`, the number of nearest neighbours belonging to each k-point of the Monkhorst-Pack mesh

Subsequent lines: `nntot`×`num_kpts` lines, ie, `nntot` lines of data for each k-point of the mesh.

Each line of consists of 5 integers. The first is the k-point number `nkp`. The second to the fifth specify it's nearest neighbours $\mathbf{k}+\mathbf{b}$: the second integer points to the k-point that is the periodic image of the $\mathbf{k}+\mathbf{b}$ that we want; the last three integers give the G-vector, in reciprocal lattice units, that brings the k-point specified by the second integer (which is in the first BZ) to the actual $\mathbf{k}+\mathbf{b}$ that we need.

5.1.7 exclude_bands block

```
begin exclude_bands
```

```
  8
  1
  2
  .
  .
end exclude_bands
```

To exclude bands (independent of kpoint) from the calculation of the overlap and projection matricies, for example to ignore shallow-core states. The first line is the number of states to exclude, the following lines give the states for be excluded.

5.1.8 An example of projections

As a concrete example: one wishes to have a set of four sp^3 projection orbitals on, say, a carbon atom at (0.5,0.5,0.5) in fractional co-ordinates relative to the direct lattice vectors. In this case `seedname.win` will contain the following lines:

```
begin projections
 C:l=-1
end projections
```

and `seedname.nnkp`, generated on the first pass of `wannier90` (with `postproc_setup=T`), will contain:

```
begin projections
   4
   0.50000    0.50000    0.50000    -1  1  1
     0.000  0.000  1.000   1.000  0.000  0.000   2.00  2.00
   0.50000    0.50000    0.50000    -1  2  1
     0.000  0.000  1.000   1.000  0.000  0.000   2.00  2.00
   0.50000    0.50000    0.50000    -1  3  1
     0.000  0.000  1.000   1.000  0.000  0.000   2.00  2.00
   0.50000    0.50000    0.50000    -1  4  1
     0.000  0.000  1.000   1.000  0.000  0.000   2.00  2.00
end projections
```

where the first line tells us that in total four projections are specified, and the subsequent lines provide the projection centre, the angular and radial parts of the orbital (see Section 3.2 for definitions), the z and x axes, and the diffusivity and cut-off radius for the projection orbital.

PWSCF, or any other *ab initio* electronic structure code, then reads `seedname.nnkp` file, calculates the projections and writes them to `seedname.amn`.

5.2 Mmn file

INPUT.

The file seedname.mmn contains the overlaps $M_{mn}^{(\mathbf{k},\mathbf{b})}$.

First line: a user comment, e.g., the date and time

Second line: 3 integers: num_bands, num_kpts, nntot

Then: num_kpts $\times$ nntot blocks of data:

First line of each block: 5 integers. The first specifies the $\mathbf{k}$ (i.e., gives the ordinal corresponding to its position in the list of k-points in seedname.win). The 2nd to 5th integers specify $\mathbf{k}+\mathbf{b}$. The 2nd integer, in particular, points to the k-point on the list that is a periodic image of $\mathbf{k}+\mathbf{b}$, and in particular is the image that is actually mentioned in the list. The last three integers specify the $\mathbf{G}$ vector, in reciprocal lattice units, that brings the k-point specified by the fourth integer, and that thus lives inside the first BZ zone, to the actual $\mathbf{k}+\mathbf{b}$ that we need.

Subsequent num_bands $\times$ num_bands lines of each block: two real numbers per line. These are the real and imaginary parts, respectively, of the actual scalar product $M_{mn}^{(\mathbf{k},\mathbf{b})}$ for $m, n \in$ [1 : num_bands]. The order of these elements is such that the first index m is fastest.

5.3 Amn file

INPUT.

The file seedname.amn contains the projection $A_{mn}^{(\mathbf{k})}$.

First line: a user comment, e.g., the date and time

Second line: 3 integers: num_bands, num_kpts, num_wann

Subsequently num_bands $\times$ num_wann $\times$ num_kpts lines: 3 integers and 2 real numbers on each line. The first two integers are the band indices m and n. The third integer specifies the $\mathbf{k}$ by giving the ordinal corresponding to its position in the list of k-points in the master input file. The real numbers are the real and imaginary parts, respectively, of the actual $A_{mn}^{(\mathbf{k})}$.

5.4 eig file

INPUT.

Required if any of disentanglement, plot_bands, plot_fermi_surface or plot_dos are .true.

The file seedname.eig contains the Kohn-Sham eigenvalues $\varepsilon_{n\mathbf{k}}$ (in eV) at each point in the Monkhorst-Pack mesh.

Each line consist of two integers and a real number. The first integer is the band index, the second integer gives the ordinal corresponding to the k-point in the list of k-points in the master input file, and the real number is the eigenvalue.

E.g.,

```
           1           1  -6.43858831271328
```

```
2        1   19.3977795287297
3        1   19.3977795287297
4        1   19.3977795287298
```

5.5 Interface with PWSCF

1. Run an 'scf' calculation with `pw`
2. Run `wannier90` with `postproc_setup` = `.true.` to generate `seedname.nnkp`
3. Run `pw2wannier90`. First it reads `pw2wannier90.in`, which defines `prefix` and `outdir` for the underlying 'scf' calculation, as well as the name of the file `seedname.nnkp`, and does a consistency check between the direct and reciprocal lattice vectors read from `seedname.nnkp` and those defined in the files specified by `prefix`. Then generate `seedname.mmn`, `seedname.amn`, `seedname.eig`
4. Run `wannier90` with `postproc_setup` = `.false.` to disentangle and localise Wannier functions

Chapter 6

Wannier as a library

This is a description of the interface between any external program and the wannier code. There are two subroutines: `wannier_setup` and `wannier_run`. Calling `wannier_setup` will return information required to construct the $M_{mn}^{(\mathbf{k},\mathbf{b})}$ overlaps (MV[1] Eq. (25)) and $A_{mn}^{(\mathbf{k})} = \langle\psi_{m\mathbf{k}}|g_n\rangle$ projections (MV Eq. (62), SMV[2] Eq. (22)). Once the overlaps and projection have been computed, calling `wannier_run` activates the main wannier code.

6.1 Subroutines

6.1.1 wannier_setup

```
wannier_setup(mp_grid,real_lattice,kpt_latt, nntot,nnlist,nncell)
```

- `integer, dimension(3), intent(in) :: mp_grid`
  The dimensions of the Monkhorst-Pack k-point grid.
- `real(kind=dp), dimension(3,3), intent(in) :: real_lattice`
  The lattice vectors in Cartesian co-ordinates in units of Angstrom.
- `real(kind=dp), dimension(3,num_kpts), intent(in) :: kpt_latt`
  The positions of the k-points in fractional co-ordinates relative to the reciprocal lattice vectors.
- `integer, intent(out) :: nntot`
  The total number of nearest neighbours for each k-point.
- `integer, dimension(num_kpts,num_nnmax), intent(out) :: nnlist`
  The list of nearest neighbours for each k-point.
- `integer,dimension(3,num_kpts,num_nnmax), intent(out) :: nncell`
  The vector, in fractional reciprocal lattice co-ordinates, that brings the `nn`$^{\mathrm{th}}$ nearest

[1] Marzari and Vanderbilt, *Phys. Rev. B* **56**, 12847 (1997)
[2] Souza, Marzari and Vanderbilt, *Phys. Rev. B* **65**, 035109 (2001)

neighbour of k-point `nkp` to its periodic image that is needed for computing the overlap $M_{mn}^{(\mathbf{k},\mathbf{b})}$.

Conditions:

- ⋆ `num_kpts` = `mp_grid(1)` × `mp_grid(2)` × `mp_grid(3)`.
- ⋆ `num_nnmax` = 12

This subroutine returns the information required to determine the required overlap elements $M_{mn}^{(\mathbf{k},\mathbf{b})}$ (MV Eq. (25)) and projections $A_{mn}^{(\mathbf{k})}$ (MV Eq. (62), SMV Eq. (22)), ie, `M_matrix_orig` and `A_matrix`, described in Section 6.1.2.

For the avoidance of doubt, `real_lattice(1,2)` is the $y-$component of the first lattice vector $\mathbf{A}_1$, etc.

The list of nearest neighbours of a particular k-point `nkp` is given by `nnlist(nkp,1:nntot)`.

Additionally, the parameters `num_shells` and `shell_list` may be specified in the wannier input file.

6.1.2 wannier_run

```
wannier_run(mp_grid,real_lattice,kpt_latt, num_bands,num_wann,num_kpts,
            nntot,num_atoms,atom_symbols,atoms_cart,M_matrix_orig,
            A_matrix,eigenvalues,U_matrix,U_matrix_opt,wann_centres,
            wann_spreads,omega_bits)
```

- `integer, dimension(3), intent(in) :: mp_grid`
  The dimensions of the Monkhorst-Pack k-point grid.
- `real(kind=dp), dimension(3,3), intent(in) :: real_lattice`
  The lattice vectors in Cartesian co-ordinates in units of Angstrom.
- `real(kind=dp), dimension(3,num_kpts), intent(in) :: kpt_latt`
  The positions of the k-points in fractional co-ordinates relative to the reciprocal lattice vectors.
- `integer, intent(in) :: num_bands`
  The total number of bands to be processed.
- `integer, intent(in) :: num_wann`
  The number of Wannier functions to be extracted.
- `integer, intent(in) :: nntot`
  The number of nearest neighbours for each k-point.
- `integer, intent(in) :: num_atoms`
  The total number of atoms in the system.

- `character(len=2), dimension(num_atoms), intent(in) :: atom_symbols`
  The elemental symbols of the atoms.

- `real(kind=dp), dimension(3,num_atoms), intent(in) :: atoms_cart`
  The positions of the atoms in Cartesian co-ordinates in Angstrom.

- `complex(kind=dp), dimension(num_bands,num_bands,num_kpts,nntot),`
  `intent(in) :: M_matrix_orig`
  The matrices of overlaps between neighbouring periodic parts of the Bloch eigenstates at each k-point, $M_{mn}^{(\mathbf{k},\mathbf{b})}$ (MV Eq. (25)).

- `complex(kind=dp), dimension(num_bands,num_wann,num_kpts),`
  `intent(in) :: A_matrix`
  The matrices describing the projection of `num_wann` trial orbitals on `num_bands` Bloch states at each k-point, $A_{mn}^{(\mathbf{k})}$ (MV Eq. (62), SMV Eq. (22)).

- `real(kind=dp), dimension(num_bands,num_kpts), intent(in) :: eigenvalues`
  The eigenvalues $\varepsilon_{n\mathbf{k}}$ corresponding to the eigenstates, in eV.

- `complex(kind=dp), dimension(num_wann,num_wann,num_kpts),`
  `intent(out) :: U_matrix`
  The unitary matrices at each k-point (MV Eq. (59))

- `complex(kind=dp), dimension(num_bands,num_wann,num_kpts),`
  `optional, intent(out) :: U_matrix_opt`
  The unitary matrices that describe the optimal sub-space at each k-point (see SMV Section IIIA).

- `real(kind=dp), dimension(3,num_wann), optional,`
  `intent(out) :: wann_centres`
  The centres of the wannier functions in Cartesian co-ordinates in Angstrom.

- `real(kind=dp), dimension(num_wann), optional,`
  `intent(out) :: wann_spreads`
  The spread of each wannier function in Å^2.

- `real(kind=dp), dimension(3), optional, intent(out) :: omega_bits`
  The values of Ω, Ω_{I} and $\tilde{\Omega}$ (MV Eq. (13)).

Conditions:

⋆ `num_wann` $\leq$ `num_bands`

⋆ `num_kpts` $=$ `mp_grid(1)` $\times$ `mp_grid(2)` $\times$ `mp_grid(3)`.

If `num_bands` $=$ `num_wann` then `U_matrix_opt` is redundant.

For the avoidance of doubt, `real_lattice(1,2)` is the $y-$component of the first lattice vector $\mathbf{A}_1$, etc.

$$\begin{aligned}
\texttt{M_matrix_orig(m,n,nkp,nn)} &= \langle u_{m\mathbf{k}}|u_{n\mathbf{k}+\mathbf{b}}\rangle \\
\texttt{A_matrix(m,n,nkp)} &= \langle \psi_{m\mathbf{k}}|g_n\rangle \\
\texttt{eigenvalues(n,nkp)} &= \varepsilon_{n\mathbf{k}}
\end{aligned}$$

where

$$\begin{aligned}
\mathbf{k} &= \texttt{kpt_latt(1:3,nkp)} \\
\mathbf{k}+\mathbf{b} &= \texttt{kpt_latt(1:3,nnlist(nkp,nn))} + \texttt{nncell(1:3,nkp,nn)}
\end{aligned}$$

and $\{|g_n\rangle\}$ are a set of initial trial orbitals. These are typically atom- or bond-centred Gaussians that are modulated by appropriate spherical harmonics.

Additional parameters should be specified in the wannier input file.

Chapter 7

Files

7.1 seedname.win

INPUT. The master input file; contains the specification of the system and any parameters for the run.

7.1.1 Units

The following are the dimensional quantities that are specified in the master input file:

- Direct lattice vectors
- Positions (of atomic or projection) centres in real space
- Energy windows
- Positions of **k**-points in reciprocal space
- `zona` and `box-size` (see Section 3.1)

Notes:

- The units (either `ang` (default) or `bohr`) can be set in the first line of the blocks `unit_cell_cart` and `atoms_cart`
- Energy is always in eV.
- Positions of **k**-points are always in crystallographic coordinates relative to the reciprocal lattice vectors.
- `box-size` and `zona` always in Angstrom and reciprocal Angstrom, respectively
- The keyword `length_unit` may be set to `ang` (default) or `bohr`, in order to set the units in which the quantities in the output file are written.

The reciprocal lattice vectors $\{\mathbf{B}_1, \mathbf{B}_2, \mathbf{B}_3\}$ are defined in terms of the direct lattice vectors $\{\mathbf{A}_1, \mathbf{A}_2, \mathbf{A}_3\}$ by the equation

$$\mathbf{B}_1 = \frac{2\pi}{\Omega}\mathbf{A}_2 \times \mathbf{A}_3 \quad \text{etc.,} \tag{7.1}$$

where the cell volume is $\Omega = \mathbf{A}_1 \cdot (\mathbf{A}_2 \times \mathbf{A}_3)$.

7.2 seedname.mmn

INPUT. See Chapter 5.

7.3 seedname.amn

INPUT. See Chapter 5.

7.4 seedname.eig

INPUT. See Chapter 5.

7.5 seedname.nnkp

OUTPUT. See Chapter 5.

7.6 seedname.wout

OUTPUT. The master output file.

7.7 seedname.chk

INPUT/OUTPUT. Sufficient information to restart the calculation or enter the plotting phase.

7.8 UNKp.s

INPUT. Read if `wannier_plot=TRUE` and used to plot the Wannier functions.

The periodic part of the bloch states represented on a regular real space grid, indexed by k-point `p` (from 1 to `num_kpts`) and spin `s` ('1' for 'up', '2' for 'down').

The name of the wavefunction file is assumed to have the form:

```
    write(wfnname,200) p,spin
200 format ('UNK',i5.5,'.',i1)
```

The first line of each file should contain 5 integers: the number of grid points in each direction (`ngx`, `ngy` and `ngz`), the k-point number `ik` and the total number of bands `num\_band` in the file. The full file will be read by Wannier90 as:

```
read(file_unit) ngx,ngy,ngz,ik,nbnd
do loop_b=1,num_bands
  read(file_unit) (r_wvfn(nx,loop_b),nx=1,ngx*ngy*ngz)
end do
```

The file can be in formatted or unformatted style, this is controlled by the logical keyword `wvfn_formatted`.

Chapter 8

Sample files

8.1 Input file

8.1.1 seedname.win

```
num_wann          : 4
mp_grid           : 4 4 4
num_iter          : 100
postproc_setup    : true

begin unit_cell_cart
ang
-1.61 0.00 1.61
 0.00 1.61 1.61
-1.61 1.61 0.00
end unit_cell_cart

begin atoms_frac
C   -0.125  -0.125  -0.125
C    0.125   0.125   0.125
end atoms_frac

bands_plot        : true
bands_num_points  : 100
bands_plot_format : gnuplot

begin kpoint_path
L 0.50000 0.50000 0.50000 G 0.00000 0.00000 0.00000
G 0.00000 0.00000 0.00000 X 0.50000 0.00000 0.50000
X 0.50000 0.00000 0.50000 K 0.62500 0.25000 0.62500
end kpoint_path

begin projections
```

```
C:l=0,l=1
end projections

begin kpoints
0.00 0.00 0.00
0.00 0.00 0.25
0.00 0.50 0.50
 .
 .
 .
0.75 0.75 0.50
0.75 0.75 0.75
end kpoints
```

8.1.2 seedname.nnkp

Running `wannier90` on the above input file would generate the following `nnkp` file:

```
File written on  9Feb2006 at 15:13: 9

calc_only_A   :  F

begin real_lattice
  -1.612340   0.000000   1.612340
   0.000000   1.612340   1.612340
  -1.612340   1.612340   0.000000
end real_lattice

begin recip_lattice
  -1.951300  -1.951300   1.951300
   1.951300   1.951300   1.951300
  -1.951300   1.951300  -1.951300
end recip_lattice

begin kpoints
     64
  0.00000   0.00000   0.00000
  0.00000   0.25000   0.00000
  0.00000   0.50000   0.00000
  0.00000   0.75000   0.00000
  0.25000   0.00000   0.00000
  .
  .
  .
  0.50000   0.75000   0.75000
```

```
  0.75000   0.00000   0.75000
  0.75000   0.25000   0.75000
  0.75000   0.50000   0.75000
  0.75000   0.75000   0.75000
end kpoints

begin projections
   8
  -0.12500   -0.12500   -0.12500     0  1  1
     0.000  0.000  1.000   1.000  0.000  0.000   2.00  2.00
  -0.12500   -0.12500   -0.12500     1  1  1
     0.000  0.000  1.000   1.000  0.000  0.000   2.00  2.00
  -0.12500   -0.12500   -0.12500     1  2  1
     0.000  0.000  1.000   1.000  0.000  0.000   2.00  2.00
  -0.12500   -0.12500   -0.12500     1  3  1
     0.000  0.000  1.000   1.000  0.000  0.000   2.00  2.00
   0.12500    0.12500    0.12500     0  1  1
     0.000  0.000  1.000   1.000  0.000  0.000   2.00  2.00
   0.12500    0.12500    0.12500     1  1  1
     0.000  0.000  1.000   1.000  0.000  0.000   2.00  2.00
   0.12500    0.12500    0.12500     1  2  1
     0.000  0.000  1.000   1.000  0.000  0.000   2.00  2.00
   0.12500    0.12500    0.12500     1  3  1
     0.000  0.000  1.000   1.000  0.000  0.000   2.00  2.00
end projections

begin nnkpts
    8
  1     2      0   0   0
  1     4      0  -1   0
  1     5      0   0   0
  1    13     -1   0   0
  1    17      0   0   0
  1    22      0   0   0
  1    49      0   0  -1
  1    64     -1  -1  -1
  2     1      0   0   0
  2     3      0   0   0
  2     6      0   0   0
  2    14     -1   0   0
  2    18      0   0   0
  2    23      0   0   0
  2    50      0   0  -1
  2    61     -1   0  -1
  .
  .
  .
```

```
 64     1       1   1   1
 64    16       0   0   1
 64    43       0   0   0
 64    48       0   0   0
 64    52       1   0   0
 64    60       0   0   0
 64    61       0   1   0
 64    63       0   0   0
end nnkpts

begin exclude_bands
   4
   1
   2
   3
   4
end exclude_bands
```