`wannier90`: User Guide

Version 1.0.3

31st July 2007

Chapter 1

Introduction

1.1 Methodology

`wannier90` computes maximally-localised Wannier functions (MLWF) 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 introduce briefly the methods and key definitions here, but full details can be found in the original papers and in Ref. [4].

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 WF centred on a lattice site $\mathbf{R}$, $w_{n\mathbf{R}}(\mathbf{r})$, is written in terms of the set of Bloch states as

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

where V is the unit cell volume, the integral is over the Brillouin zone (BZ), and $\mathbf{U}^{(\mathbf{k})}$ is a unitary matrix that mixes the Bloch states at each $\mathbf{k}$. $\mathbf{U}^{(\mathbf{k})}$ is not uniquely defined and different choices will lead to WF with varying spatial localisations. We define the spread Ω of the WF as

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

The total spread can be decomposed into a gauge invariant term Ω_{I} plus a term $\tilde{\Omega}$ that is dependant on the gauge choice $\mathbf{U}^{(\mathbf{k})}$. $\tilde{\Omega}$ 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

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

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

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

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

`wannier90` requires two ingredients from an initial electronic structure calculation.

1. The overlaps between the cell periodic part of the Bloch states $|u_{n\mathbf{k}}\rangle$
$$M_{mn}^{(\mathbf{k},\mathbf{b})} = \langle u_{m\mathbf{k}}|u_{n\mathbf{k}+\mathbf{b}}\rangle, \tag{1.7}$$
where the vectors $\mathbf{b}$, which connect a given k-point with its neighbours, are determined by `wannier90` according to the prescription outlined in Ref. [1].

2. As a starting guess the projection of the Bloch states $|\psi_{n\mathbf{k}}\rangle$ onto trial localised orbitals $|g_n\rangle$
$$A_{mn}^{(\mathbf{k})} = \langle \psi_{m\mathbf{k}}|g_n\rangle, \tag{1.8}$$

Note that $\mathbf{M}^{(\mathbf{k},\mathbf{b})}$, $\mathbf{A}^{(\mathbf{k})}$ and $\mathbf{U}^{(\mathbf{k})}$ are all small, $N \times N$ matrices[1] that are independent of the basis set used to obtain the original Bloch states.

To date, `wannier90` has been used in combination with electronic codes based on plane-waves and pseudopotentials (norm-conserving and ultrasoft [5]) as well as mixed basis set techniques such as FLAPW [6].

1.1.1 Entangled Energy Bands

The above description is sufficient to obtain MLWF for an 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 Ref. [2].

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

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

where $\mathbf{U}^{{\rm dis}(\mathbf{k})}$ is a rectangular $N \times N_{\rm win}^{(\mathbf{k})}$ matrix[2]. The set of $\mathbf{U}^{{\rm dis}(\mathbf{k})}$ 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 optimal 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 (e.g., 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.

[1] Technically, this is true for the case of an isolated group of N bands from which we obtain N MLWF. When using the disentanglement procedure of Ref. [2], $\mathbf{A}^{(\mathbf{k})}$, for example, is a rectangular matrix. See Section 1.1.1.

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

1.1.2 Getting Help

The latest version of wannier90 and documentation can always be found at

http://www.wannier.org

There is a wannier90 mailing list for discussing issues in the development, theory, coding and algorithms pertinent to MLWF. You can register for this mailing list by following the links at

http://www.wannier.org/forum.html

Finally, many frequently asked questions are answered in Chapter 8.

1.1.3 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, Y.-S. Lee, I. Souza, D. Vanderbilt and N. Marzari,
> wannier90: A Tool for Obtaining Maximally-Localised Wannier Functions,
> *Comput. Phys. Commun.*, submitted (2007); http://arxiv.org/abs/0708.0650

It would 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 (2001)

1.1.4 Credits

The present release of wannier90 was written by Arash A. Mostofi (Imperial College London, UK), Jonathan R. Yates (University of Cambridge, UK), and Young-Su Lee (KIST, S. Korea). wannier90 is based on routines written in 1996-7 for isolated bands by Nicola Marzari and David Vanderbilt, and for entangled bands by Ivo Souza, Nicola Marzari, and David Vanderbilt in 2000-1.

Acknowledgements: Stefano de Gironcoli (SISSA, Trieste, Italy) for the PWSCF interface; Michel Posternak (EPFL, Switzerland) for the original plotting routines.

1.1.5 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 `seedname.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 that is used unless the keyword is given in `seedname.win`. 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 further examples see Section 9.1 and the the `wannier90` Tutorial.

2.3 Keyword List

Keyword	Type	Description
System Parameters		
NUM_WANN	I	Number of WF
NUM_BANDS	I	Number of bands passed to the code
UNIT_CELL_CART	P	Unit cell vectors in Cartesian coordinates
ATOMS_CART *	P	Positions of atoms in Cartesian coordinates
ATOMS_FRAC *	R	Positions of atoms in fractional coordinates with respect to the lattice vectors
MP_GRID	I	Dimensions of the Monkhorst-Pack grid of k-points
KPOINTS	R	List of k-points in the Monkhorst-Pack grid
GAMMA_ONLY	L	Wavefunctions from underlying ab initio calculation are manifestly real
NUM_SHELLS	I	Number of shells in finite difference formula
SHELL_LIST	I	Which shells to use in finite difference formula
SEARCH_SHELLS	I	The number of shells to search when determining finite difference formula

Table 2.1: `seedname.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 `seedname.nnkp` file
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_FORMATTED	L	Read the wavefunctions from a (un)formatted file
SPIN	S	Which spin channel to read
DEVEL_FLAG	S	Flag for development use
TIMING_LEVEL	I	Determines amount of timing information written to output
TRANSLATE_HOME_CELL	L	To translate final Wannier centres to home unit cell when writing xyz file
WRITE_XYZ	L	To write final centres in xyz file format

Table 2.2: `seedname.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. TRANSLATE_HOME_CELL only relevant if WRITE_XYZ is `.true.`

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: `seedname.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_WINDOW	I	The number of iterations over which convergence of Ω is assessed
CONV_TOL	P	The convergence tolerance for finding Ω
CONV_NOISE_AMP	R	The amplitude of random noise applied towards end of minimisation procedure
CONV_NOISE_NUM	I	The number of times random noise is applied
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 WF 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
TRIAL_STEP *	R	The trial step length for the parabolic line search during the minimisation of Ω
FIXED_STEP *	R	The fixed step length to take during the minimisation of Ω, instead of doing a parabolic line search
USE_BLOCH_PHASES **	L	To use phases for initial projections

Table 2.4: `seedname.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. * FIXED_STEP and TRIAL_STEP may not both be defined in the same input file. **Cannot be used in conjunction with disentanglement.

Keyword	Type	Description
Plot Parameters		
WANNIER_PLOT	L	Plot the WF
WANNIER_PLOT_LIST	I	List of WF to plot
WANNIER_PLOT_SUPERCELL	I	Size of the supercell for plotting the WF
WANNIER_PLOT_FORMAT	S	File format in which to plot the WF
WANNIER_PLOT_MODE	S	Mode in which to plot the WF, molecule or crystal
WANNIER_PLOT_RADIUS	R	Cut-off radius of WF*
BANDS_PLOT	L	Plot 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
BANDS_PLOT_MODE	S	Slater-Koster type interpolation or Hamiltonian cut-off
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
HR_PLOT	L	Write the Hamiltonian in the WF basis
HR_CUTOFF	R	
DIST_CUTOFF_MODE	S	
DIST_CUTOFF	R	
ONE_DIM_AXIS	S	
TRANSLATION_CENTRE_FRAC	L	

Table 2.5: `seedname.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. * Only applies when WANNIER_PLOT_FORMAT is `cube`.

Keyword	Type	Description
Transport Parameters		
TRANSPORT	L	
TRANSPORT_MODE	S	
TRAN_WIN_MIN	R	
TRAN_WIN_MAX	R	
TRAN_ENERGY_STEP	R	
TRAN_NUM_BB	I	
TRAN_NUM_LL	I	
TRAN_NUM_RR	I	
TRAN_NUM_CC	I	
TRAN_NUM_LC	I	
TRAN_NUM_CR	I	
TRAN_NUM_BANDC	I	
TRAN_WRITE_HT	L	
TRAN_READ_HT	L	
TRAN_USE_SAME_LEAD	L	
HR_CUTOFF	R	
DIST_CUTOFF_MODE	S	
DIST_CUTOFF	R	
ONE_DIM_AXIS	S	
TRANSLATION_CENTRE_FRAC	L	

Table 2.6: `seedname.win` file keywords controlling transport. 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 WF 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]
```

$$\begin{matrix} A_{1x} & A_{1y} & A_{1z} \\ A_{2x} & A_{2y} & A_{2z} \\ A_{3x} & A_{3y} & A_{3z} \end{matrix}$$

```
end unit_cell_cart
```

Here A_{1x} is the x-component of the first lattice vector $\mathbf{A}_1$, A_{2y} is the y-component of the second lattice vector $\mathbf{A}_2$, etc.

`[units]` specifies the units in which the lattice vectors are defined: either `Bohr` or `Ang`.

The default value is `Ang`.

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.

Cartesian coordinates

```
begin atoms_cart
[units]
```

$$\begin{matrix} P & R^P_x & R^P_y & R^P_z \\ Q & R^Q_x & R^Q_y & R^Q_z \\ \vdots \end{matrix}$$

```
end atoms_cart
```

The first entry on a line is the atomic symbol. The next three entries are the atom's position $\mathbf{R} = (R_x, R_y, R_z)$ in Cartesian coordinates. The first line of the block, `[units]`, specifies the units in which the coordinates are given and can be either `bohr` or `ang`. If not present, the default is `ang`.

Fractional coordinates

```
begin atoms_frac
```

$$\begin{array}{cccc} P & F_1^P & F_2^P & F_3^P \\ Q & F_1^Q & F_2^Q & F_3^Q \\ \vdots & & & \end{array}$$

```
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 $\mathbf{F} = F_1\mathbf{A}_1 + F_2\mathbf{A}_2 + F_3\mathbf{A}_3$ relative to the cell lattice vectors $\mathbf{A}_i$, $i \in [1, 3]$.

2.4.5 integer, dimension :: mp_grid(3)

Dimensions of the regular (Monkhorst-Pack) k-point mesh. For example, for a $2 \times 2 \times 2$ grid:

```
mp_grid : 2  2  2
```

No default.

2.4.6 K-points

Each line gives the coordinate $\mathbf{K} = K_1\mathbf{B}_1 + K_2\mathbf{B}_2 + K_3\mathbf{B}_3$ of a k-point in relative (crystallographic) units, i.e., in fractional units with respect to the primitive reciprocal lattice vectors $\mathbf{B}_i$, $i \in [1, 3]$. 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
```

$$\begin{array}{ccc} K_1^1 & K_2^1 & K_3^1 \\ K_1^2 & K_2^2 & K_3^2 \\ \vdots & & \end{array}$$

```
end kpoints
```

There is no default.

2.4.7 `logical :: gamma_only`

If `gamma_only=TRUE`, then `wannier90` uses a branch of algorithms for disentanglement and localisation that exploit the fact that the Bloch eigenstates obtained from the underlying ab initio calculation are manifestly real. This can be the case when only the Γ-point is used to sample the Brillouin zone. The localisation procedure that is used in the Γ-only branch is based on the method of Ref. [3].

The default value is `FALSE`.

2.4.8 Shells

The MV scheme requires a finite difference expression for $\nabla_{\mathbf{k}}$ defined on a uniform Monkhorst-Pack mesh of k-points. The vectors $\{\mathbf{b}\}$ connect each mesh-point $\mathbf{k}$ to its nearest neighbours. N_{sh} shells of neighbours are included in the finite-difference formula, with M_s vectors in the s^{th} shell. For $\nabla_{\mathbf{k}}$ to be correct to linear order, we require that the following equation is satisfied (Eq. B1 of Ref. [1]):

$$\sum_{s}^{N_{\mathrm{sh}}} w_s \sum_{i}^{M_{\mathrm{s}}} b_{\alpha}^{i,s} b_{\beta}^{i,s} = \delta_{\alpha\beta} \,, \tag{2.1}$$

where $\mathbf{b}^{i,s}$, $i \in [1, M_s]$, is the i^{th} vector belonging to the s^{th} shell with associated weight w_s, and α and β run over the three Cartesian indices.

2.4.9 `integer :: num_shells`

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

The default value is 0.

2.4.10 `integer :: shell_list(num_shells)`

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

2.4.11 `integer :: search_shells`

Specifies the number of shells of neighbours over which to search in attempting to determine an automatic solution to the B1 condition Eq. 2.1.

The default value is 12.

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 to be used by a first-principles code.

```
begin projections
.
.
end projections
```

If `guiding_centres`=`TRUE`, then the projection centres are used as the guiding centres in the Wannierisation routine.

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`, then `postproc_setup` is set to `TRUE`, over-riding its value in the `seedname.win` file.

The default value is `FALSE`.

2.6.2 `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.

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

The length unit to be used for writing quantities in the output file `seedname.wout`.

The valid options for this parameter are:

- `Ang` (default)
- `Bohr`

2.6.4 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.

2.6.5 integer :: exclude_bands(:)

A k-point 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. For example, to exclude bands 2, 6, 7, 8 and 12:

```
exclude_bands : 2, 6-8, 12
```

2.6.6 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 `seedname.chk` was written
- `wannierise`. Restart from the beginning of the wannierise routine
- `plot`. Go directly to the plotting phase

2.6.7 character(len=20) :: wvfn_formatted

If `wvfn_formatted`=`TRUE`, then the wavefunctions will be read from disk as formatted (ie ASCII) files; otherwise they will be read as unformatted files. Unformatted is generally preferable as the files will take less disk space and 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 (i.e., for test cases).

The default value of this parameter is `FALSE`.

2.6.8 character(len=20) :: spin

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

The default value of this parameter is `up`.

2.6.9 integer :: timing_level

Determines the amount of timing information regarding the calculation that will be written to the output file. A value of 1 produces the least information.

The default value is 1.

2.6.10 logical :: translate_home_cell

Determines whether to translate the final Wannier centres to the home unit cell at the end of the calculation. Mainly useful for molecular systems in which the molecule resides entirely within the home unit cell and user wants to write an xyz file (`write_xyz=.TRUE.`) for the WF centres to compare with the structure.

The default value is `false`.

2.6.11 logical :: write_xyz

Determines whether to write the final Wannier centres to an xyz file, `seedname_centres.xyz`, for subsequent visualisation.

The default value is `false`.

2.7 Disentanglement

These keywords control the disentanglement routine of Ref. [2], i.e., the iterative minimisation of Ω_{I}. 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. Units are eV.

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. Units are eV.

The default is the highest eigenvalue in the given states (i.e., 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. Units are eV.

If dis_froz_max is given, then 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 (frozen) energy window for the disentanglement procedure. If dis_froz_max is not specified, then there are no frozen states. Units are eV.

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 200.

2.7.6 real(kind=dp) :: dis_mix_ratio

In the disentanglement procedure, the mixing parameter to use for convergence (see pages 4-5 of Ref. [2]). A value of 0.5 is a 'safe' choice. Using 1.0 (i.e., no mixing) often gives faster convergence, but may cause the minimisation of Ω_{I} to be unstable in some cases.

Restriction: $0.0 <$ dis_mix_ratio ≤ 1.0

The default value is 0.5

2.7.7 real(kind=dp) :: dis_conv_tol

In the disentanglement procedure, the minimisation of Ω_{I} is said to be converged if the fractional change in the gauge-invariant spread between successive iterations is less than dis_conv_tol for dis_conv_window iterations. Units are Å^2.

The default value is 1.0E-10

2.7.8 integer :: dis_conv_window

In the disentanglement procedure, the minimisation is said to be 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

Iterative minimisation of $\widetilde{\Omega}$, the non-gauge-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 :: conv_window`

If `conv_window` > 1, then the minimisation is said to be converged if the change in Ω over `conv_window` successive iterations is less than `conv_tol`. Otherwise, the minimisation proceeds for num_iter iterations (default).

The default value is -1

2.8.4 `real(kind=dp) :: conv_tol`

If `conv_window` > 1, then this is the convergence tolerance on Ω, otherwise not used. Units are Å^2.

The default value is 1.0E-10

2.8.5 `real(kind=dp) :: conv_noise_amp`

If `conv_noise_amp` > 0, once convergence (as defined above) is achieved, some random noise f is added to the search direction, and the minimisation is continued until convergence is achieved once more. If the same value of Ω as before is arrived at, then the calculation is considered to be converged. If not, then random noise is added again and the procedure repeated up to a maximum of `conv_noise_num` times. `conv_noise_amp` is the amplitude of the random noise f that is added to the search direction: $0 < |f| <$ `conv_noise_amp`. This functionality requires `conv_window` > 1. If `conv_window` is not specified, it is set to the value 5 by default.

If `conv_noise_amp` ≤ 0, then no noise is added (default).

The default value is -1.0

2.8.6 integer :: conv_noise_num

If conv_noise_amp > 0, then this is the number of times in the minimisation that random noise is added.

The default value is 3

2.8.7 integer :: num_dump_cycles

Write sufficient information to do a restart every num_dump_cycles iterations.

The default is 100

2.8.8 integer :: num_print_cycles

Write data to the master output file seedname.wout every num_print_cycles iterations.

The default is 1

2.8.9 logical :: write_r2mn

If write_r2mn = true, then the matrix elements $\langle m|r^2|n\rangle$ (where m and n refer to WF) are written to file seedname.r2mn at the end of the Wannierisation procedure.

The default value of this parameter is FALSE.

2.8.10 logical :: guiding_centres

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

The default value is FALSE.

2.8.11 integer :: num_guide_cycles

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

The default value is 1.

2.8.12 integer :: num_no_guide_iter

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

The default value is 0.

2.8.13 real(kind=dp) :: trial_step

The value of the trial step for the parabolic fit in the line search minimisation used in the minimisation of the spread function. Cannot be used in conjunction with `fixed_step` (see below). If the minimisation procedure doesn't converge, try decreasing the value of `trial_step` to give a more accurate line search.

The default value is 2.0

2.8.14 real(kind=dp) :: fixed_step

If this is given a value in the input file, then a fixed step of length `fixed_step` (instead of a parabolic line search) is used at each iteration of the spread function minimisation. Cannot be used in conjunction with `trial_step`. This can be useful in cases in which minimisation with a line search fails to converge.

There is no default value.

2.8.15 logical :: use_bloch_phases

Determines whether to use the Bloch functions as the initial guess for the projections. Can only be used if `disentanglement = false`.

Th default value is `false`.

2.9 Post-Processing

Capabilities:

- Plot the WF
- Plot the interpolated band structure
- Plot the Fermi surface
- Output the Hamiltonian in the WF basis
- Transport calculation (quantum conductance and density of states)

2.9.1 logical :: wannier_plot

If `wannier_plot` = `TRUE`, then 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 WF 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

WF can be plotted in either XCrySDen (xsf) format or Gaussian cube format. The valid options for this parameter are:

– `xcrysden` (default)

– `cube`

If `wannier_plot_format=cube`: Most visualisation programs (including XCrySDen) are only able to handle cube files for systems with orthogonal lattice vectors.[1] `wannier90` checks this on reading the `seedname.win` and reports an error if cube format has been selected and the lattice vectors are not mutually orthogonal.

2.9.4 integer :: wannier_plot_list(:)

A list of WF to plot. The WF numbered as per the `seedname.wout` file after the minimisation of the spread.

The default behaviour is to plot all WF. For example, to plot WF 4, 5, 6 and 10:

```
wannier_plot_list : 4-6, 10
```

2.9.5 character(len=20) :: wannier_plot_mode

Choose the mode in which to plot the WF, either as a molecule or as a crystal. Only relevant if `wannier_plot_format=xcrysden`.

The valid options for this parameter are:

– `crystal` (default)

– `molecule`

[1] It's worth noting that the visualisation program VMD (`http://www.ks.uiuc.edu/Research/vmd`), for example, is able to deal with certain special cases of non-orthogonal lattice vectors. See `http://www.ks.uiuc.edu/Research/vmd/plugins/molfile/cubeplugin.html`. At present `wannier90` only supports orthogonal lattice vectors for cube output.

2.9.6 real(kind=dp) :: wannier_plot_radius

If wannier_plot_format is cube, then wannier_plot_radius determines the cut-off radius of the WF for the purpose of plotting. wannier_plot_radius must be greater than 0. Units are Å.

The default value is 3.5.

2.9.7 logical :: bands_plot

If bands_plot = TRUE, then 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.

2.9.8 kpoint_path

Defines the path in k-space along which to calculate the bandstructure. Each line gives the start and end point (with labels) for a section of the path. Values are in fractional coordinates with respect to the primitive reciprocal lattice vectors.

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

2.9.9 integer :: bands_num_points

If bands_plot = TRUE, then the number of points along the first section of the bandstructure plot given by kpoint_path. Other sections will have the same density of k-points.

The default value for bands_num_points is 100.

2.9.10 character(len=20) :: bands_plot_format

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

- gnuplot (default)
- xmgrace

2.9.11 character(len=20) :: bands_plot_mode

- s-k (default)
- cut

2.9.12 logical :: fermi_surface_plot

If fermi_surface_plot = TRUE, then 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 by XCrySDen in order to plot the Fermi surface.

The default value is FALSE.

2.9.13 integer :: fermi_surface_num_points

If fermi_surface_plot = TRUE, then the number of divisions in the regular k-point grid used to calculate the Fermi surface.

The default value for fermi_surface_num_points is 50.

2.9.14 real(kind=dp) :: fermi_energy

The Fermi energy in eV. Whilst this is not directly used by the wannier90, it is a useful parameter to set as it will be written into the bxsf file.

The default value is 0.0

2.9.15 character(len=20) :: fermi_surface_plot_format

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

- xcrysden (default)

2.9.16 logical :: hr_plot

If hr_plot is TRUE, then the Hamiltonian matrix in the basis of the WF will be written to a file seedname_hr.dat.

The default value is FALSE

2.9.17 logical :: transport

If transport = TRUE, then the code will calculate quantum conductance and density of states.

The default value of this parameter is FALSE.

2.9.18 character :: transport_mode

- bulk (default)
- lcr

2.9.19 real(kind=dp) :: tran_win_min

The default value is -3.0

2.9.20 real(kind=dp) :: tran_win_max

The default value is 3.0

2.9.21 real(kind=dp) :: tran_energy_step

The default value is 0.01

2.9.22 integer :: tran_num_bb

The default value is 0

2.9.23 integer :: tran_num_ll

The default value is 0

2.9.24 integer :: tran_num_rr

The default value is 0

2.9.25 integer :: tran_num_cc

The default value is 0

2.9.26 integer :: tran_num_lc

The default value is 0

2.9.27 integer :: tran_num_cr

The default value is 0

2.9.28 integer :: tran_num_bandc

The default value is 0

2.9.29 logical :: tran_write_ht

The default value is FALSE

2.9.30 logical :: tran_read_ht

The default value is FALSE

2.9.31 logical :: tran_use_same_lead

The default value is TRUE

2.9.32 real(kind=dp) :: translation_centre_frac(3)

The default value is (0.0,0.0,0.0)

2.9.33 real(kind=dp) :: hr_cutoff

The absolute value of the largest matrix element of the Hamiltonian in the WF basis at lattice vector $\mathbf{R}$ is given by $h_{\max}(\mathbf{R}) = |\max\ H_{mn}(\mathbf{R})|$. If $h_{\max}(\mathbf{R}) >$ hr_cutoff, then the matrix elements $H_{mn}(\mathbf{R})$ are retained in the Hamiltonian that is written to seedname.h.dat. Otherwise they are deemed to be insignificant and are discarded.

The default value is 0.0

2.9.34 character :: dist_cutoff_mode

– three_dim (default)

– one_dim

2.9.35 real(kind=dp) :: dist_cutoff

The default value is 0.0

2.9.36 `character :: one_dim_axis`

- `x`
- `y`
- `z`

No 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.

The code is 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 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 Θ_{lm_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 3.3 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
```

```
    ⋮
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. The default value 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 (see above).

`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 $l = 2$, $m_{\mathrm{r}} = 1$ (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`.

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 Short-Cuts

3.2.1 Random projections

It is possible to specify the projections, for example, as follows:

```
Begin Projections
random
C:sp3
End Projections
```

in which case `wannier90` uses four sp^3 orbitals centred on each C atom and then chooses the appropriate number of randomly-centred s-type Gaussian functions for the remaining projection functions. If the block only consists of the string `random` and no specific projection centres are given, then all of the projection centres are chosen randomly.

3.2.2 Bloch phases

Setting `use_bloch_phases = true` in the input file absolves the user of the need to specify explicit projections. In this case, the Bloch wave-functions are used as the projection orbitals, namely $A_{mn}^{(\mathbf{k})} = \langle\psi_{m\mathbf{k}}|\psi_{n\mathbf{k}}\rangle = \delta_{mn}$.

3.3 Orbital Definitions

The angular functions $\Theta_{lm_{\mathrm{r}}}(\theta,\varphi)$ associated with particular values of l and m_{r} 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	m_r	Name	$\Theta_{lm_r}(\theta,\varphi)$
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 $\Theta_{lm_r}(\theta,\varphi)$ associated with particular values of l and m_r for $l \geq 0$.

l	$m_{\rm r}$	Name	$\Theta_{lm_{\rm r}}(\theta,\varphi)$
-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 $\Theta_{lm_{\rm r}}(\theta,\varphi)$ associated with particular values of l and $m_{\rm r}$ 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

Code Overview

`wannier90` can operate in two modes:

1. *Post-processing mode:* read in the overlaps and projections from file as computed inside a first-principles code. We expect this to be the most common route to using `wannier90`, and is described in Ch. 5;

2. *Library mode:* as a set of library routines to be called from within a first-principles code that passes the overlaps and projections to the `wannier90` library routines and in return gets the unitary transformation corresponding to MLWF. This route should be used if the MLWF are needed within the first-principles code, for example in post-LDA methods such as LDA+U or SIC, and is described in Ch. 6.

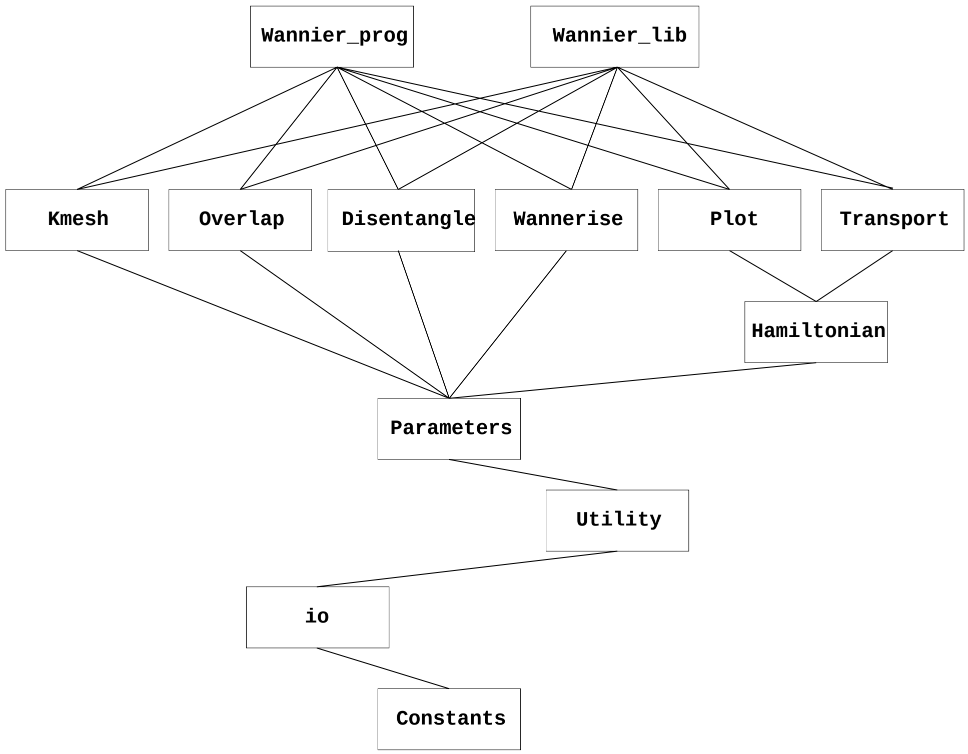

Figure 4.1: Schematic overview of the module structure of wannier90. Modules may only use data and subroutines from lower modules.

Chapter 5

wannier90 as a post-processing tool

This is a description of how to use wannier90 as a post-processing tool.

The code must be run twice. On the first pass either the logical keyword postproc_setup must be set to .true. in the input file seedname.win or the code must be run with the command line option -pp. Running the code then generates the file seedname.nnkp which provides the information required to construct the $M_{mn}^{(\mathbf{k},\mathbf{b})}$ overlaps (Ref. [1], Eq. (25)) and $A_{mn}^{(\mathbf{k})}$ (Ref. [1], Eq. (62); Ref. [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 seedname.nnkp file

OUTPUT, if postproc_setup = .true.

The file seedname.nnkp provides the information needed to determine the required overlap elements $M_{mn}^{(\mathbf{k},\mathbf{b})}$ and projections $A_{mn}^{(\mathbf{k})}$. It is written automatically when the code is invoked with the -pp command-line option (or when postproc_setup=.true. in seedname.win. There should be no need for the user to edit this file.

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
```

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
   centre   l  mr  r
```

```
     z-axis   x-axis   zona
   .
   .
end projections
```

Notes:

`n_proj`: integer; the number of projection centres, equal to the number of MLWF `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.

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`$\times$`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 k-point) 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
   0.50000    0.50000    0.50000    -1  2  1
     0.000  0.000  1.000   1.000  0.000  0.000   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
   0.50000    0.50000    0.50000    -1  4  1
     0.000  0.000  1.000   1.000  0.000  0.000   2.00
end projections
```

where the first line tells us that in total four projections are specified, and the subsquent lines provide the projection centre, the angular and radial parts of the orbital (see Section 3.3 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 seedname.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 $m, n \in [1, \texttt{num\_bands}]$. The order of these elements is such that the first index m is fastest.

5.3 seedname.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 seedname.win. The real numbers are the real and imaginary parts, respectively, of the actual $A_{mn}^{(\mathbf{k})}$.

5.4 seedname.eig file

INPUT.

Required if any of disentanglement, plot_bands, plot_fermi_surface or hr_plot 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 `seedname.win`, 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

There is a seamless interface between `wannier90` and PWSCF, a plane-wave DFT code that comes as part of the `quantum-espresso` package (see `www.quantum-espresso.org` or `www.pwscf.org`). At the time of writing, interfaces to other electronic structure codes are in progress, e.g., CASTEP (`www.castep.org`), ABINIT (`www.abinit.org`) and FLEUR (`www.fleur.de`). But, for the time being, you will need to download and compile PWSCF (i.e., the `pw.x` code) and the post-processing interface `pw2wannier90.x`. Please refer to the documentation that comes with the `quantum-espresso` distribution for instructions.

1. Run 'scf'/'nscf' calculation(s) with `pw`
2. Run `wannier90` with `postproc_setup` = `.true.` to generate `seedname.nnkp`
3. Run `pw2wannier90`. First it reads an input file, e.g., `seedname.pw2wan`, 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`. `pw2wannier90` generates `seedname.mmn`, `seedname.amn` and `seedname.eig`
4. Run `wannier90` with `postproc_setup` = `.false.` to disentangle bands (if required), localise MLWF, and use MLWF for plotting, bandstructures, Fermi surfaces etc.

Examples of how the interface with PWSCF works are given in the `wannier90` Tutorial.

5.5.1 seedname.pw2wan

A number of keywords may be specified in the `pw2wannier90` input file:

- `outdir` – Location to write output files. Default is `'./'`
- `prefix` – Prefix for the PWSCF calculation. Default is `' '`
- `seedname` – Seedname for the `wannier90` calculation. Default is `'wannier'`

- spin_component – Spin component. Takes values 'up', 'down' or 'none' (default).
- wan_mode – Either 'standalone' (default) or 'library'
- write_unk – Set to .true. to write the periodic part of the Bloch functions for plotting in wannier90. Default is .false.
- reduce_unk – Set to .true. to reduce file-size (and resolution) of Bloch functions by a factor of 8. Default is .false. (only relevant if write_unk=.true.)[1]
- wvfn_formatted – Set to .true. to write formatted wavefunctions. Default is .false. (only relevant if write_unk=.true.)
- write_amn – Set to .false. if $A_{mn}^{(\mathbf{k})}$ not required. Default is .true.
- write_mmn – Set to .false. if $M_{mn}^{(\mathbf{k},\mathbf{b})}$ not required. Default is .true.

For examples of use, refer to the wannier90 Tutorial.

[1]Note that there is a small bug with this feature in v3.2 (and subsequent patches) of quantum-espresso. Please use a later version (if available) or the CVS version of pw2wannier90.f90, which has been fixed.

Chapter 6

wannier90 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 (Ref. [1], Eq. (25)) and $A_{mn}^{(\mathbf{k})} = \langle\psi_{m\mathbf{k}}|g_n\rangle$ projections (Ref. [1], Eq. (62); Ref. [2], Eq. (22)). Once the overlaps and projection have been computed, calling `wannier_run` activates the minimisation and plotting routines in `wannier90`.

6.1 Subroutines

6.1.1 wannier_setup

```
wannier_setup(seed_name,mp_grid,num_kpts,real_lattice,recip_lattice,
              kpt_latt,num_bands_tot,num_atoms,atom_symbols,atoms_cart,
              gamma_only,nntot,nnlist,nncell,num_bands,num_wann,proj_site,
              proj_l,proj_m,proj_radial,proj_z,proj_x,proj_zona,
              exclude_bands)
```

- `character(len=*), intent(in) :: seed_name`
  The seedname of the current calculation.

- `integer, dimension(3), intent(in) :: mp_grid`
  The dimensions of the Monkhorst-Pack k-point grid.

- `integer, intent(in) :: num_kpts`
  The number of k-points on the Monkhorst-Pack 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(3,3), intent(in) :: recip_lattice`
  The reciprocal lattice vectors in Cartesian co-ordinates in units of reciprocal 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_tot`
  The total number of bands in the first-principles calculation (note: including semi-core states).

- `integer, intent(in) :: num_atoms`
  The total number of atoms in the system.

- `character(len=20), 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.

- `logical, intent(in) :: gamma_only`
  Set to `.true.` if the underlying electronic structure calculation has been performed with only Γ-point sampling and, hence, if the Bloch eigenstates that are used to construct $A_{mn}^{(\mathbf{k})}$ and $M_{mn}^{(\mathbf{k},\mathbf{b})}$ are real.

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

- `integer, intent(out) :: num_bands`
  The number of bands in the first-principles calculation used to form the overlap matricies (note: excluding eg. semi-core states).

- `integer, intent(out) :: num_wann`
  The number of MLWF to be extracted.

- `real(kind=dp), dimension(3,num_bands_tot), intent(out) :: proj_site`
  Projection function centre in crystallographic co-ordinates relative to the direct lattice vectors.

- `integer, dimension(num_bands_tot), intent(out) :: proj_l`
  l specifies the angular part $\Theta_{lm_{\mathrm{r}}}(\theta,\varphi)$ of the projection function (see Tables 3.1, 3.2 and 3.3).

- `integer, dimension(num_bands_tot), intent(out) :: proj_m`
  m_{r} specifies the angular part $\Theta_{lm_{\mathrm{r}}}(\theta,\varphi)$, of the projection function (see Tables 3.1, 3.2 and 3.3).

- `integer, dimension(num_bands_tot), intent(out) :: proj_radial`
  r specifies the radial part $R_{\rm r}(r)$ of the projection function (see Tables 3.1, 3.2 and 3.3).

- `real(kind=dp), dimension(3,num_bands_tot), intent(out) :: proj_z`
  Defines the axis from which the polar angle θ in spherical polar coordinates is measured. Default is `0.0 0.0 1.0`.

- `real(kind=dp), dimension(3,num_bands_tot), intent(out) :: proj_x`
  Must be orthogonal to `z-axis`; default is `1.0 0.0 0.0` or a vector perpendicular to `proj_z` if `proj_z` is given; defines the axis from with the azimuthal angle φ in spherical polar coordinates is measured.

- `real(kind=dp), dimension(num_bands_tot), intent(out) :: proj_zona`
  The value of $\frac{Z}{a}$ associated with the radial part of the atomic orbital. Units are in reciprocal Angstrom.

- `integer, dimension(num_bands_tot), intent(out) :: exclude_bands`
  Kpoints independant list of bands to exclude from the calculation of the MLWF (e.g., semi-core states).

Conditions:

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

⋆ `num_nnmax` $= 12$

This subroutine returns the information required to determine the required overlap elements $M_{mn}^{(\mathbf{k},\mathbf{b})}$ and projections $A_{mn}^{(\mathbf{k})}$, i.e., `M_matrix` 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 `wannier90` input file.

6.1.2 wannier_run

```
wannier_run(seed_name,mp_grid,num_kpts,real_lattice,recip_lattice,
            kpt_latt,num_bands,num_wann,nntot,num_atoms,atom_symbols,
            atoms_cart,gamma_only,M_matrix_orig,A_matrix,eigenvalues,
            U_matrix,U_matrix_opt,lwindow,wann_centres,wann_spreads,
            spread)
```

- `character(len=*), intent(in) :: seed_name`
  The seedname of the current calculation.

- `integer, dimension(3), intent(in) :: mp_grid`
  The dimensions of the Monkhorst-Pack k-point grid.

- `integer, intent(in) :: num_kpts`
  The number of k-points on the Monkhorst-Pack 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(3,3), intent(in) :: recip_lattice`
  The reciprical lattice vectors in Cartesian co-ordinates in units of inverse 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 MLWF 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=20), 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.

- `logical, intent(in) :: gamma_only`
  Set to `.true.` if the underlying electronic structure calculation has been performed with only Γ-point sampling and, hence, if the Bloch eigenstates that are used to construct $A_{mn}^{(\mathbf{k})}$ and $M_{mn}^{(\mathbf{k},\mathbf{b})}$ are real.

- `complex(kind=dp), dimension(num_bands,num_bands,nntot,num_kpts),`
  `intent(in) :: M_matrix`
  The matrices of overlaps between neighbouring periodic parts of the Bloch eigenstates at each k-point, $M_{mn}^{((\mathbf{k},\mathbf{b}))}$ (Ref. [1], 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, $A_{mn}^{(\mathbf{k})}$ (Ref. [1], Eq. (62); Ref. [2], 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 (Ref. [1], Eq. (59))

- `complex(kind=dp), dimension(num_bands,num_wann,num_kpts),`
  `intent(out) :: U_matrix_opt`
  The unitary matrices that describe the optimal sub-space at each k-point (see Ref. [2], Section IIIA). The array is packed (see below)

- `logical, dimension(num_bands,num_kpts), intent(out) :: lwindow`
  The element `lwindow(nband,nkpt)` is `.true.` if the band `nband` lies within the outer energy window at kpoint `nkpt`.

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

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

- `real(kind=dp), dimension(3), intent(out) :: spread`
  The values of Ω, Ω_{I} and $\tilde{\Omega}$ (Ref. [1], Eq. (13)).

Conditions:

⋆ $\texttt{num\_wann} \leq \texttt{num\_bands}$

⋆ $\texttt{num\_kpts} = \texttt{mp\_grid(1)} \times \texttt{mp\_grid(2)} \times \texttt{mp\_grid(3)}$.

If $\texttt{num\_bands} = \texttt{num\_wann}$ then `U_matrix_opt` is the identity matrix and `lwindow=.true.`

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

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

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 `wannier90` 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. For a description of input parameters, see Chapter 2; for examples, see Section 9.1 and the `wannier90` Tutorial.

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
- Convergence thresholds for the minimisation of Ω
- `zona` (see Section 3.1)
- `wannier_plot_cube`: cut-off radius for plotting WF in Gaussian cube format

Notes:

- The units (either `ang` (default) or `bohr`) in which the lattice vectors, atomic positions or projection centres are given can be set in the first line of the blocks `unit_cell_cart`, `atoms_cart` and `projections`, respectively, in `seedname.win`.
- Energy is always in eV.
- Convergence thresholds are always in Å^2

- Positions of k-points are always in crystallographic coordinates relative to the reciprocal lattice vectors.
- `zona` is always in reciprocal Angstrom (Å^{-1})
- 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 `seedname.wout` are written.
- `wannier_plot_radius` is in Angstrom

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 $V = \mathbf{A}_1 \cdot (\mathbf{A}_2 \times \mathbf{A}_3)$.

7.2 seedname.mmn

INPUT. Written by the underlying electronic structure code. See Chapter 5 for details.

7.3 seedname.amn

INPUT. Written by the underlying electronic structure code. See Chapter 5 for details.

7.4 seedname.eig

INPUT. Written by the underlying electronic structure code. See Chapter 5 for details.

7.5 seedname.nnkp

OUTPUT. Written by `wannier90` when `postproc_setup=.TRUE.` (or, alternatively, when `wannier90` is run with the `-pp` command-line option). See Chapter 5 for details.

7.6 seedname.wout

OUTPUT. The master output file. Here we give a description of the main features of the output. The verbosity of the output is controlled by the input parameter `iprint`. The higher the value, the more detail is given in the output file. The default value is 1, which prints minimal information.

7.6.1 Header

The header provides some basic information about `wannier90`, the authors, and the execution time of the current run.

```
+---------------------------------------------------+
|                                                   |
|                     WANNIER90                     |
|                                                   |
+---------------------------------------------------+
|                                                   |
|        Welcome to the Maximally-Localized         |
|        Generalized Wannier Functions code         |
|            http://www.wannier.org                 |
|                                                   |
|  Authors:                                         |
|    Arash A. Mostofi   (Imperial College London)   |
|    Jonathan R. Yates  (University of Cambridge)   |
|    Young-Su Lee       (KIST, S. Korea)            |
|                                                   |
                          .
                          .
|                                                   |
|   Copyright (c) 1997-2007 J. Yates, A. Mostofi,   |
|   Y.-S. Lee, N. Marzari, I. Souza, D. Vanderbilt  |
|                                                   |
|          Release: 1.1       1st Sep 2007          |
|                                                   |
                          .
                          .
|                                                   |
+---------------------------------------------------+
|    Execution started on 10Sep2007 at 12:46:57     |
+---------------------------------------------------+
```

7.6.2 System information

This part of the output file presents information that `wannier90` has read or inferred from the master input file `seedname.win`. This includes real and reciprocal lattice vectors, atomic positions, k-points, parameters for job control, disentanglement, localisation and plotting.

```
                        ------
                        SYSTEM
```

```
                          ------

                    Lattice Vectors (Ang)
          a_1     3.938486   0.000000   0.000000
          a_2     0.000000   3.938486   0.000000
          a_3     0.000000   0.000000   3.938486

         Unit Cell Volume:      61.09251  (Ang^3)

              Reciprocal-Space Vectors (Ang^-1)
          b_1     1.595330   0.000000   0.000000
          b_2     0.000000   1.595330   0.000000
          b_3     0.000000   0.000000   1.595330

*----------------------------------------------------------------------------*
|   Site       Fractional Coordinate          Cartesian Coordinate (Ang)     |
+----------------------------------------------------------------------------+
| Ba   1   0.00000   0.00000   0.00000   |    0.00000   0.00000   0.00000    |
| Ti   1   0.50000   0.50000   0.50000   |    1.96924   1.96924   1.96924    |
                                         .
                                         .
*----------------------------------------------------------------------------*

                            ------------
                            K-POINT GRID
                            ------------

             Grid size =  4 x  4 x  4      Total points =   64

*---------------------------------- MAIN ------------------------------------*
|  Number of Wannier Functions               :              9               |
|  Number of input Bloch states              :              9               |
|  Output verbosity (1=low, 5=high)          :              1               |
|  Length Unit                               :             Ang              |
|  Post-processing setup (write *.nnkp)      :              F               |
                                             .
                                             .
*----------------------------------------------------------------------------*
```

7.6.3 Nearest-neighbour k-points

This part of the output files provides information on the b-vectors and weights chosen to satisfy the condition of Eq. 2.1.

```
*---------------------------------- K-MESH ----------------------------------*
+----------------------------------------------------------------------------+
```

```
|                    Distance to Nearest-Neighbour Shells                      |
|                    ------------------------------------                      |
|          Shell             Distance (Ang^-1)          Multiplicity          |
|          -----             -----------------          ------------          |
|            1                    0.398833                     6              |
|            2                    0.564034                    12              |
                                      .
                                      .
+-----------------------------------------------------------------------------+
| The b-vectors are chosen automatically                                      |
| The following shells are used:   1                                          |
+-----------------------------------------------------------------------------+
|                        Shell   # Nearest-Neighbours                         |
|                        -----   --------------------                         |
|                          1               6                                  |
+-----------------------------------------------------------------------------+
| Completeness relation is fully satisfied [Eq. (B1), PRB 56, 12847 (1997)]  |
+-----------------------------------------------------------------------------+
```

7.6.4 Disentanglement

Then (if required) comes the part where Ω_{I} is minimised to disentangle the optimally-connected subspace of states for the localisation procedure in the next step.

First, a summary of the energy windows that are being used is given:

```
*------------------------------- DISENTANGLE --------------------------------*
+-----------------------------------------------------------------------------+
|                              Energy  Windows                                |
|                              ---------------                                |
|                   Outer:    2.81739  to   38.00000  (eV)                    |
|                   Inner:    2.81739  to   13.00000  (eV)                    |
+-----------------------------------------------------------------------------+
```

Then, each step of the iterative minimisation of Ω_{I} is reported.

```
                   Extraction of optimally-connected subspace
                   ------------------------------------------
+---------------------------------------------------------------------+<-- DIS
|  Iter     Omega_I(i-1)      Omega_I(i)      Delta (frac.)    Time   |<-- DIS
+---------------------------------------------------------------------+<-- DIS
      1       3.82493590       3.66268867       4.430E-02      0.36    <-- DIS
      2       3.66268867       3.66268867       6.911E-15      0.37    <-- DIS
                                      .
                                      .
```

```
            <<<      Delta < 1.000E-10  over  3 iterations     >>>
            <<< Disentanglement convergence criteria satisfied >>>

       Final Omega_I     3.66268867 (Ang^2)

 +----------------------------------------------------------------------------+
```

The first column gives the iteration number. For a description of the minimisation procedure and expressions for $\Omega_{\rm I}^{(i)}$, see the original paper [2]. The procedure is considered to be converged when the fractional difference between $\Omega_{\rm I}^{(i)}$ and $\Omega_{\rm I}^{(i-1)}$ is less than `dis_conv_tol` over `dis_conv_window` iterations. The final column gives a running account of the wall time (in seconds) so far. Note that at the end of each line of output, there are the characters "`<-- DIS`". This enables fast searching of the output using, for example, the Unix command `grep`:

```
my_shell> grep DIS wannier.wout | less
```

7.6.5 Wannierisation

The next part of the input file provides information on the minimisation of $\widetilde{\Omega}$. At each iteration, the centre and spread of each WF is reported.

```
*------------------------------- WANNIERISE ---------------------------------*
 +--------------------------------------------------------------------+<-- CONV
 | Iter  Delta Spread     RMS Gradient      Spread (Ang^2)      Time  |<-- CONV
 +--------------------------------------------------------------------+<-- CONV

 ------------------------------------------------------------------------------
 Initial State
  WF centre and spread    1  (  0.000000,  1.969243,  1.969243 )     1.52435832
  WF centre and spread    2  (  0.000000,  1.969243,  1.969243 )     1.16120620
                                        .
                                        .
      0     0.126E+02     0.0000000000       12.6297685260       0.29  <-- CONV
        O_D=      0.0000000 O_OD=      0.1491718 O_TOT=     12.6297685 <-- SPRD
 ------------------------------------------------------------------------------
 Cycle:      1
  WF centre and spread    1  (  0.000000,  1.969243,  1.969243 )     1.52414024
  WF centre and spread    2  (  0.000000,  1.969243,  1.969243 )     1.16059775
                                        .
                                        .
  Sum of centres and spreads ( 11.815458, 11.815458, 11.815458 )    12.62663472

      1    -0.313E-02     0.0697660962       12.6266347170       0.34  <-- CONV
        O_D=      0.0000000 O_OD=      0.1460380 O_TOT=     12.6266347 <-- SPRD
 Delta: O_D= -0.4530841E-18 O_OD= -0.3133809E-02 O_TOT= -0.3133809E-02 <-- DLTA
 ------------------------------------------------------------------------------
```

```
Cycle:      2
 WF centre and spread    1  (  0.000000,  1.969243,  1.969243 )     1.52414866
 WF centre and spread    2  (  0.000000,  1.969243,  1.969243 )     1.16052405
                                     .
                                     .
  Sum of centres and spreads ( 11.815458, 11.815458, 11.815458 )    12.62646411

      2    -0.171E-03     0.0188848262       12.6264641055       0.38  <-- CONV
       O_D=      0.0000000 O_OD=      0.1458674 O_TOT=     12.6264641 <-- SPRD
Delta: O_D= -0.2847260E-18 O_OD= -0.1706115E-03 O_TOT= -0.1706115E-03 <-- DLTA
------------------------------------------------------------------------------
                                     .
                                     .
------------------------------------------------------------------------------
Final State
 WF centre and spread    1  (  0.000000,  1.969243,  1.969243 )     1.52416618
 WF centre and spread    2  (  0.000000,  1.969243,  1.969243 )     1.16048545
                                     .
                                     .
 Sum of centres and spreads ( 11.815458, 11.815458, 11.815458 )    12.62645344

        Spreads (Ang^2)       Omega I      =    12.480596753
       ================       Omega D      =     0.000000000
                              Omega OD     =     0.145856689
   Final Spread (Ang^2)       Omega Total  =    12.626453441
------------------------------------------------------------------------------
```

It looks quite complicated, but things look more simple if one uses `grep`:

`my_shell> grep CONV wannier.wout`

gives

```
+--------------------------------------------------------------------+<-- CONV
| Iter  Delta Spread     RMS Gradient      Spread (Ang^2)      Time  |<-- CONV
+--------------------------------------------------------------------+<-- CONV
      0     0.126E+02     0.0000000000       12.6297685260       0.29  <-- CONV
      1    -0.313E-02     0.0697660962       12.6266347170       0.34  <-- CONV
                                                  .
                                                  .
     50     0.000E+00     0.0000000694       12.6264534413       2.14  <-- CONV
```

The first column is the iteration number, the second is the change in Ω from the previous iteration, the third is the root-mean-squared gradient of Ω with respect to variations in the unitary matrices $\mathbf{U}^{(\mathbf{k})}$, and the last is the time taken (in seconds). Depending on the input parameters used, the procedure either runs for `num_iter` iterations, or a convergence criterion is applied on Ω. See Section 2.8 for details.

Similarly, the command

```
my_shell> grep SPRD wannier.wout
```

gives

```
        O_D=      0.0000000 O_OD=      0.1491718 O_TOT=     12.6297685 <-- SPRD
        O_D=      0.0000000 O_OD=      0.1460380 O_TOT=     12.6266347 <-- SPRD
                                                .
                                                .
        O_D=      0.0000000 O_OD=      0.1458567 O_TOT=     12.6264534 <-- SPRD
```

which, for each iteration, reports the value of the diagonal and off-diagonal parts of the non-gauge-invariant spread, as well as the total spread, respectively. Recall from Section 1.1 that $\Omega = \Omega_{\mathrm{I}} + \Omega_{\mathrm{D}} + \Omega_{\mathrm{OD}}$.

7.6.6 Plotting

After WF have been localised, `wannier90` enters its plotting routines (if required). For example, if you have specified an interpolated bandstucture:

```
*---------------------------------------------------------------------------*
|                               PLOTTING                                    |
*---------------------------------------------------------------------------*

Calculating interpolated band-structure
```

7.6.7 Summary timings

At the very end of the run, a summary of the time taken for various parts of the calculation is given. The level of detail is controlled by the `timing_level` input parameter (set to 1 by default).

```
*===========================================================================*
|                             TIMING INFORMATION                            |
*===========================================================================*
|    Tag                                                Ncalls      Time (s)|
|---------------------------------------------------------------------------|
|kmesh: get                                        :         1         0.212|
|overlap: read                                     :         1         0.060|
|wann: main                                        :         1         1.860|
|plot: main                                        :         1         0.168|
*---------------------------------------------------------------------------*

All done: wannier90 exiting
```

7.7 seedname.chk

INPUT/OUTPUT. Information required to restart the calculation or enter the plotting phase. If we have used disentanglement this file also contains the rectangular matrices $\mathbf{U}^{\mathrm{dis}(\mathbf{k})}$.

7.8 seedname.r2mn

OUTPUT. Written if `write_r2mn` = `true`. The matrix elements $\langle m|r^2|n\rangle$ (where m and n refer to MLWF)

7.9 seedname_band.dat

OUTPUT. Written if `bands_plot=.TRUE.`; The raw data for the interpolated band structure.

7.10 seedname_band.gnu

OUTPUT. Written if `bands_plot=.TRUE.` and `bands_plot_format=gnuplot`; A `gnuplot` script to plot the interpolated band structure.

7.11 seedname_band.agr

OUTPUT. Written if `bands_plot=.TRUE.` and `bands_plot_format=xmgrace`; A `grace` file to plot the interpolated band structure.

7.12 seedname_band.kpt

OUTPUT. Written if `bands_plot=.TRUE.`; The k-points used for the interpolated band structure, in units of the reciprocal lattice vectors. This file can be used to generate a comparison band structure from a first-principles code.

7.13 seedname.bxsf

OUTPUT. Written if `fermi_surface_plot=.TRUE.`; A Fermi surface plot file suitable for plotting with XCrySDen.

7.14 seedname_w.xsf

OUTPUT. Written if wannier_plot=.TRUE. and wannier_plot_format=xcrysden. Contains the $\mathtt{w}^{\mathrm{th}}$ WF in real space in a format suitable for plotting with XCrySDen or VMD, for example.

7.15 seedname_w.cube

OUTPUT. Written if wannier_plot=.TRUE. and wannier_plot_format=cube. Contains the $\mathtt{w}^{\mathrm{th}}$ WF in real space in Gaussian cube format, suitable for plotting in XCrySDen, VMD, gopenmol etc.

7.16 UNKp.s

INPUT. Read if wannier_plot=.TRUE. and used to plot the MLWF.

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.

7.17 seedname_centres.xyz

OUTPUT. Written if translate_home_cell=.TRUE.; xyz format atomic structure file suitable for viewing with your favourite visualiser (jmol, gopenmol, vmd, etc.).

7.18 seedname_hr.dat

OUTPUT. Written if `hr_plot=.TRUE.`. The first line gives the date and time at which the file was created. The subsequent lines each contain, respectively, the components of the vector $\mathbf{R}$ in terms of the lattice vectors $\{\mathbf{A}_i\}$, the indices m and n, and the real and imaginary parts of the Hamiltonian matrix element $H_{mn}^{(\mathbf{R})}$ in the WF basis, e.g.,

```
 Created on 24May2007 at 23:32:09
    0    0   -2     1     1    -0.001013     0.000000
    0    0   -2     2     1     0.000270     0.000000
    0    0   -2     3     1    -0.000055     0.000000
    0    0   -2     4     1     0.000093     0.000000
    0    0   -2     5     1    -0.000055     0.000000
    .
    .
    .
```

Chapter 8

Frequently Asked Questions

8.1 General Questions

8.1.1 What is `wannier90`?

`wannier90` is a computer package, written in Fortran90, for obtaining maximally-localised Wannier functions, using them to calculate bandstructures, Fermi surfaces, dielectric properties, sparse Hamiltonians and many things besides.

8.1.2 Where can I get `wannier90`?

The most recent release of `wannier90` is always available on our website `www.wannier.org`.

8.1.3 Where can I get the most recent information about `wannier90`?

The latest news about `wannier90` can be followed on our website `www.wannier.org`.

8.1.4 Is `wannier90` free?

Yes! `wannier90` is available for use free-of-charge under the GNU General Public Licence. See the file `LICENCE` that comes with the `wannier90` distribution or the GNU hopepage at `www.gnu.org`.

8.1.5 Who wrote `wannier90`?

`wannier90` is written by Arash A. Mostofi (Imperial College London), Jonathan. R. Yates (University of Cambridge) and Young-Su Lee (Korea Institute of Science and Technology). `wannier90` is based on algorithms written in 1996-7 by Nicola Marzari (Massachusetts Institute of Technology) and David Vanderbilt (Rutgers University), and in 2000-1 by Ivo Souza

(University of California at Berkeley), Nicola Marzari and David Vanderbilt.

The interface to PWSCF was written by Stefano de Gironcoli (SISSA, Trieste).

8.2 Getting Help

8.2.1 Is there a Tutorial available for `wannier90`?

Yes! The `examples` directory of the `wannier90` distribution contains input files for a number of tutorial calculations. The `doc` directory contains the accompanying tutorial handout.

8.2.2 Where do I get support for `wannier90`?

There are a number of options:

1. The `wannier90` User Guide, available in the `doc` directory of the distribution, and from the webpage (`www.wannier.org/user_guide.html`)
2. The `wannier90` webpage for the most recent announcements (`www.wannier.org`)
3. The `wannier90` mailing list (see `www.wannier.org/forum.html`)

8.2.3 Is there a mailing list for `wannier90`?

Yes! You need to register: go to `www.wannier.org/forum.html` and follow the instructions.

8.3 Providing Help: Finding and Reporting Bugs

8.3.1 I think I found a bug. How do I report it?

- Check and double-check. Make sure it's a bug.
- Check that it is a bug in `wannier90` and not a bug in the software interfaced to `wannier90`.
- Check that you're using the latest version of `wannier90`.
- Send an email to `developers@wannier.org`. Make sure to describe the problem and to attach all input and output files relating to the problem that you have found.

8.3.2 I have got an idea! How do I report a wish?

We're always happy to listen to suggestions. Email your idea to the `wannier90` developers at `developers@wannier.org`.

8.3.3 I want to help! How can I contribute to wannier90?

Great! There's always plenty of functionality to add. Email us at developers@wannier.org to let us know about the functionality you'd like to contribute.

8.3.4 I like wannier90! Should I donate anything to its authors?

Our Swiss bank account number is... just kidding! There is no need to donate anything, please just cite our paper in any publications that arise from your use of wannier90:

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

8.4 Installation

8.4.1 How do I install wannier90?

Follow the instructions in the file README.install in the main directory of the wannier90 distribution.

8.4.2 Are there wannier90 binaries available?

Not at present.

8.4.3 Is there anything else I need?

Yes. wannier90 works on top of an electronic structure calculation. At the time of writing, wannier90 is interfaced to the PWSCF code, a plane-wave, pseudopotential, density-functional theory code, which is part of the quantum-espresso package. You will need to download it from the webpage www.quantum-espresso.org or www.pwscf.org. Then compile PWSCF and the wannier90 interface program pw2wannier90. For instructions, please refer to the documentation that comes with the quantum-espresso distribution.

For examples of how to use PWSCF and wannier90 in conjunction with each other, see the wannier90 Tutorial.

Interfaces to other electronic structure codes, such as CASTEP,[1] FLEUR[2] and ABINIT,[3] are currently in progress and should become available in the near future.

[1] www.castep.org
[2] www.flapw.de
[3] www.abinit.org

8.5 Compile-time Problems

8.6 Run-time Problems

8.7 Using wannier90

Chapter 9

Sample Input Files

9.1 Master input file: `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
```

9.2 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
  -0.12500   -0.12500   -0.12500     1  1  1
     0.000  0.000  1.000   1.000  0.000  0.000   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
  -0.12500   -0.12500   -0.12500     1  3  1
     0.000  0.000  1.000   1.000  0.000  0.000   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
   0.12500    0.12500    0.12500     1  1  1
     0.000  0.000  1.000   1.000  0.000  0.000   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
   0.12500    0.12500    0.12500     1  3  1
     0.000  0.000  1.000   1.000  0.000  0.000   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
```

Bibliography

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

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

[3] Gygi, F., Fattebert, J.-L., and Schwegler, E., Computation of Maximally Localized Wannier Functions using a simultaneous diagonalization algorithm, *Comp. Phys. Commun.* **155**, 1 (2003).

[4] A. A. Mostofi, J. R. Yates, Y.-S. Lee, I. Souza, D. Vanderbilt and N. Marzari, `wannier90`: A Tool for Obtaining Maximally-Localized Wannier Functions, *Comput. Phys. Commun.*, submitted (2007); `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] M. Posternak, A. Baldereschi, S. Massidda and N. Marzari, Maximally Localized Wannier Functions in Antiferromagnetic MnO within the FLAPW Formalism, *Phys. Rev. B* **65**, 184422 (2002).