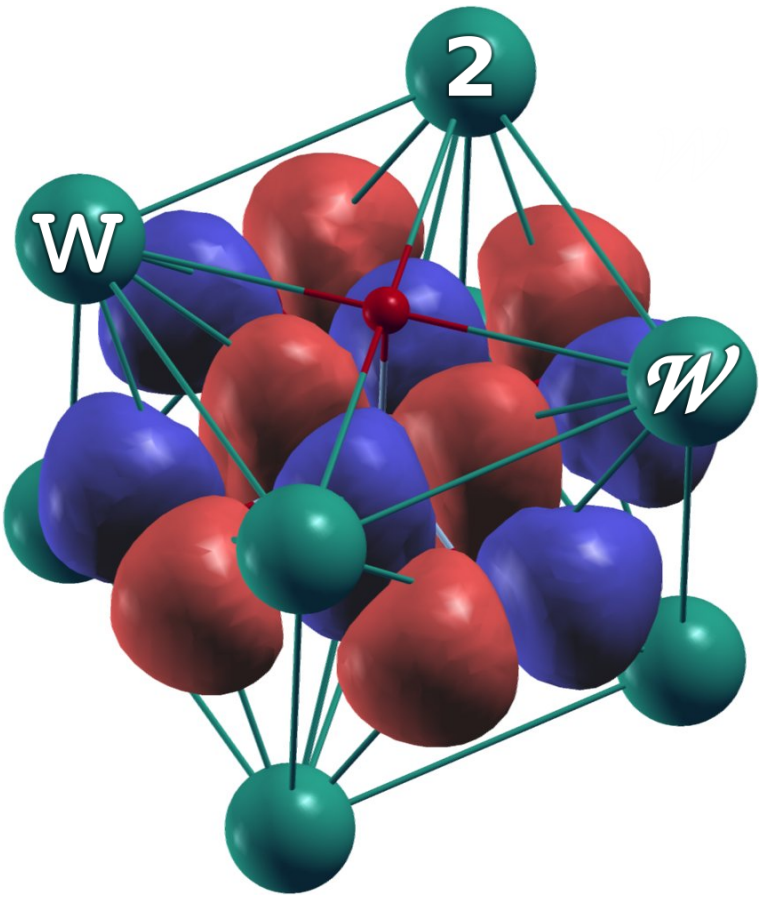
WIEN2WANNIER 2.0 User's Guide

From linearized augmented plane waves to maximally localized Wannier functions.

ELIAS ASSMANN JAN KUNEŠ PHILIPP WISSGOTT

November 28, 2016



Introduction

In the solid state theorist's tool kit, algorithms based on density functional theory (DFT) represent the backbone applications. One of the more popular codes available is the WIEN2k package [1, 2]. It is based on dividing the real space unit cell into "muffin-tin" spheres centered on the ionic cores and an interstitial region. On the former, basis function with more or less atomic features are employed, while on the latter plain waves are used. The combined basis functions are called (linearized) augmented plane waves (LAPW).

While in WIEN2k a k-space representation of wave functions is convenient, there are many applications where a real-space picture is preferable: Determining transport properties (hopping parameters), visualization, and, especially, in codes relying on local orbital basis functions such as dynamical mean-field theory (DMFT [3]). One way to change to a real space representation is a Fourier transform of the Bloch states $\psi_{n\boldsymbol{k}}(\boldsymbol{r})$ from the DFT calculation, yielding

$$w_{m\boldsymbol{R}}(\boldsymbol{r}) = \frac{V}{(2\pi)^3} \int_{BZ} d\boldsymbol{k}\ \mathrm{e}^{-\mathrm{i}\boldsymbol{k}\cdot\boldsymbol{R}} \left(\sum_n U^{(\boldsymbol{k})}_{nm} \psi_{n\boldsymbol{k}}(\boldsymbol{r}) \right).$$

The resulting functions $w_{m\boldsymbol{R}}(\boldsymbol{r})$, parametrized by a direct lattice vector $\boldsymbol{R}$, are called Wannier functions (WF). Due to the additional phase factors $U^{(\boldsymbol{k})}_{nm}$, which all lead to valid Wannier functions, there is considerable ambiguity in the choice of the real space basis set. This can be overcome by choosing Wannier orbitals $w_{m\boldsymbol{R}}(\boldsymbol{r})$ with minimal real-space extent (spread $\langle \Delta \boldsymbol{r}^2 \rangle$). These are known as maximally localized Wannier functions (MLWF). A popular program to compute the MLWF for a given set of Bloch functions is WANNIER90 [4, 5].

Unfortunately, the application of WANNIER90 to the LAPW basis set of WIEN2k is not as straightforward as for a pure plane wave basis. In this guide, we describe the package WIEN2WANNIER which provides an interface between WIEN2k and WANNIER90. In any scientific publications arising from the use of WIEN2WANNIER, we ask that you cite Ref. [6].

> J. KUNEŠ, R. ARITA, P. WISSGOTT, A.TOSCHI, H. IKEDA, and K. HELD,
> Comput. Phys. Commun. 181, 1888 (2010), arXiv:1004.3934

to acknowledge your use of our code. When using WANNIER90, you should cite Ref. [4]

> A.A. MOSTOFI, J.R. YATES, Y.-S. LEE, I. SOUZA, D. VANDERBILT, and N. MARZARI,
> Comput. Phys. Commun. 178, 685 (2008), arXiv:0708.0650

independently of WIEN2WANNIER (cf. WANNIER90 user's guide [5]).

Acknowledgment Many thanks to Karsten Held, Peter Blaha, Karlheinz Schwarz, Nikolaus Frohner, Philipp Hansmann, Nico Parragh and Giorgio Sangiovanni.

Contents

Todo list

1 Quickstart

Contents

This section outlines the procedure for a "standard" WIEN2WANNIER calculation. A detailed description of the individual programs can be found in Section 2. For a quick reference, see the plain-text file `CHEATSHEET`.

1.1 Preparatory steps

Before running WIEN2WANNIER, one needs a converged WIEN2k calculation. Additionally, during the setup for WIEN2WANNIER, the bands which are to be taken into account will have to be specified, and the main character (e.g., *d* bands on atom 2) of these bands should be known.

To obtain this information, a combination of partial DOS and bandstructure, or a "band character" plot is often expedient (e.g. `spaghetti`'s "fat bands" option, or `SpaghettiPrimavera` and `prima.py`, available in the *unsupported software* section of the WIEN2k website).

Copy Required Files

As a "zeroth" step before a Wannier projection, it is recommended to use the script

```
prepare_w2wdir subdir
```

to create the subdirectory *`subdir`* where the rest of the workflow will take place. Thus, one WIEN2k calculation can be the starting point of several WIEN2WANNIER runs.

1.2 Interface & Wannierization

Write Input Files

The script `init_w2w` takes the following steps:

`x kgen -fbz` generates the non-symmetrized k-mesh that WANNIER90 requires. The density of the mesh influences the quality of localization and visualization of the Wannier functions. Remark: The interface was only tested with unshifted k-meshes.

`x findbands` looks in *case*`.output1` for bands in a given energy range $[E_{\min}, E_{\max}]$, and outputs the corresponding band indices $b_{\min}, b_{\max}$. To choose the energy window of interest, consult a (p)DOS and/or band structure plot.

`write_inwf` writes the main input file *case*`.inwf` for the interface. The band indices $b_{\min}$, $b_{\max}$ have to be specified, and initial projections A_{mn} may be given in terms of atomic sites and appropriate spherical harmonics.

`write_win` writes the input file for `wannier90.x`, *case*`.win`, on the basis of *case*`.inwf` and other files.

`x wannier90 -pp` reads the k-mesh in *case*`.win` and writes a list of nearest-neighbor k-points to *case*`.nnkp`.

Run Interface

After running `init_w2w`, one can proceed:

`x lapw1` before the actual interface run, the eigenvectors and eigenvalues for the new (non-symmetrized) k-mesh have to be computed.

`x w2w` computes the overlaps M_{mn}, initial projections A_{mn} and eigenvalues E_n, and writes them to *case*`.mmn`, *case*`.amn`, and *case*`.eig`.

Run Wannierization

Finally,

`x wannier90` computes the $U(\boldsymbol{k})$ by maximum localization. Output is stored in *case*`.wout`. The Wannier orbitals should be converged to a spread which is usually smaller than the unit cell of the structure.

1.3 Verification & Postprocessing

After a successful WANNIER90 run, one should check if the centers and spreads of the Wannier functions (printed in *case*`.wout`) are sensible. Another important consistency check is to compare the Wannier-interpolated bandstructure to the one computed by WIEN2k. WIEN2WANNIER also provides programs to create a real-space plot of the Wannier functions.

Compare Bandstructures

With the option `hr_plot=T`, WANNIER90 writes a bandstructure derived from the Wannier-interpolated Hamiltonian $H(\boldsymbol{k})$ to *case*`_band.dat`. To compare this to the bandstructure computed by `spaghetti`, e.g. in `gnuplot`, use the command (including a conversion from Bohr^{-1} to Ångström^{-1})

```
p 'case.spaghetti_ene' u ($4/0.53):5, 'case_band.dat' w l
```

Plot Wannier Functions

`write_inwplot` asks for a real-space grid on which the Wannier functions should be plotted, and writes *`case`*`.inwplot`.

`x wplot -wf` *`m`* evaluates Wannier function number m on the real-space grid, and writes the density $|w_m(\boldsymbol{r})|^2$ to *`case`*`_`*`m`*`.psink` and the phase $\arg w_m(\boldsymbol{r})$ to *`case`*`_`*`m`*`.psiarg`.

`wplot2xsf` converts all *`case`*`*.psink` plus *`case`*`*.psiarg` files in the directory to `xsf` files which can be opened by XCrySDen.

`xcrysden -xsf` *`case`*`_`*`m`*`.xsf.gz` ; pick "Tools → Data Grid" from the menu and press OK. In the isosurface controls window choose an appropriate isovalue, e.g. 0.1, and check the "Render +/- isovalue" box.

2 Detailed description

Contents

This section lists the programs included in WIEN2WANNIER, along with brief descriptions and usage summaries. All programs have online help (-h), which may be more complete than the material here.

Programs that are described here include: the "main" WIEN2WANNIER programs w2w and wplot; several utility programs used in a typical WIEN2WANNIER run; and some programs that are needed in special cases, or grew out of WIEN2WANNIER development and are provided here in the hope that they will be useful, even if they are not needed in the context of WIEN2WANNIER.

2.1 w2w — compute overlaps

This is the main program of the interface which provides the files *case*.mmn, *case*.amn and *case*.eig for WANNIER90 given the output of a WIEN2k run (most importantly, *case*.vector). w2w is based on lapwdm, since the main task, for the computation of the overlap matrices

$$M_{mn}^{(\boldsymbol{k},\boldsymbol{b})} = \langle \psi_{m\boldsymbol{k}} | \mathrm{e}^{-\mathrm{i}\boldsymbol{b}\cdot\boldsymbol{r}} | \psi_{n\boldsymbol{k}+\boldsymbol{b}} \rangle \tag{2.1}$$

that are stored in *case*.mmn, is to determine the basis functions $\psi_{nk}(r)$ in the entire unit cell (using appropriate boundary conditions on the muffin-tin spheres).

In addition to standard WIEN2k files, a file *case*.nnkp is required which defines the $\boldsymbol{b}$ vectors for each $\boldsymbol{k}$ in Eq. (2.1). wannier90.x delivers this file when called with the flag -pp.

Execution

The program w2w is executed by invoking the command:

```
x w2w [-c -up|-dn -so -p] |
w2w w2w.def [#proc] | w2wc w2w.def [#proc]
```

While w2w itself is not parallelized, like x lapw2 -qtl, it accepts a -p switch (or #proc) to read parallel vector and energy files.

With spin-orbit coupling (-so), w2w must be called separately for -up and -dn, resulting in *case*.mmn{up,dn}, *case*.amn{up,dn}, and *case*.eig{up,dn}. In a second step, the M_{mn} and A_{mn} must be added (and the eigenvalues copied) to produce *case*.mmn, *case*.amn, and *case*.eig, on which WANNIER90 may be run. This step is done by the program w2waddsp, which is automatically invoked by x wannier90 -so.

Input

A sample input file for WIEN2WANNIER is given below. It can be generated using write_inwf.

case.inwf

```
1  BOTH      # compute  Amn | Mmn | both
2   21 23    # band-lo, band-hi
3    3   3    # max LJ in exp(ibr) expansion;  #(initial projections)
4  2         # #(terms)  [d-xy orbital]
5   2  2 -2  0 +1   # IAT;  l, m;  Re(coeff), Im(coeff)
6   2  2 +2  0 -1
7  2         #           [d-(x2-y2) orbital]
8   2  2 -1  1  0
9   2  2 +1  1  0
10 1         #           [d-z2 orbital]
11  2  2  0  1  0
```

end of *case*.inwf

This file is read using Fortran list-oriented reads, i.e., items are separated by white space.

line 1 mode — what to calculate

mode	Amn	Only determine the initial orbital projections *case*.amn
	Mmn	Only determine the overlap matrices *case*.mmn
	both	Determine both the initial orbital projections and the overlap matrices

line 2 Blo, Bhi — band window

Blo, Bhi	*int*	the minimal and maximal Bloch band from the vector file to be taken into account, determining the lower edge of the (outer) energy window

line 3 `LJmax, Nproj`

`LJmax`	*int*	the number of terms which are used in the spherical harmonics expansion to approximate $\exp(-\mathrm{i}\boldsymbol{kb})$, usually 3–4 is sufficient.
`Nproj`	*int*	the number of target Wannier functions

line 4 `Nterm` — number of terms *(Nproj times)*

`Nent`	*int*	the number of Y_ℓ^m terms (lines to follow for this initial projection

line 5 `Iat, l, m, Re, Im` — Y_ℓ^m term *(Nterm times)*

`Iat`	*int*	the atom where the entry is centered
`l, m`	*int*	the orbital and magnetic quantum numbers ℓ and m for this entry
`Re, Im`	*real*	the complex coefficient multiplying Y_ℓ^m

If initial projections are given, there must be `Nproj` blocks **l. 4+5**, one for each initial projection. Within each block, write `Nterm` lines **l. 5**, one for each Y_ℓ^m term. Otherwise, if `mode` is `Mmn`, the initial projections can be omitted.

Note The initial projections (**l. 5**) generated by `write_inwf` are normalized. For non-normalized initial projections (as in the template), the extra factor will be included in *case*`.amn`, but neutralized by the orthonormalization step in WANNIER90.

2.2 `wplot` — Wannier function plotter

`wplot` evaluates the Wannier functions on a real-space grid. It reads the transformation matrices $U(\boldsymbol{k})$ from the file *case*`.chk` written by `wannier90.x` and thereby constructs the Wannier functions from the original WIEN2k Bloch states.

Execution

The program `wplot` is executed by invoking the command:

```
x wplot [-up|-dn -c -so -p -wf m] |
wplot def [m [#proc]] | wplotc def [m [#proc]]
```

`wplot` is not parallelized, but accepts a `-p` switch (or `#proc`) to read parallel `vector` files. Moreover, as a crude form of parallelization, one can run several `wplot` instances in the same directory in parallel without interference (e.g. to plot several Wannier functions on fine grids).

The input file contains the index of the Wannier orbital to be plotted; *m* above overrides this. Output goes to *case*`_`*m*`.psink` ($|w(\boldsymbol{r})|^2$) and *case*`_`*m*`.psiarg` ($\arg w(\boldsymbol{r})$).

Input

`wplot` is based on `lapw7` and shares the general structure of the input file. A sample input file for `wplot` is given below. It can be generated via `write_inwplot`, or adapted from the template in `$WIENROOT/SRC_templates`.

case.inwplot

```
3D ORTHO       # dim.;  O(RTHOGONAL) | N(ON-ORTHOGONAL)
 0  0  0  1   # origin:  x, y, z; idiv
 1  0  0  1   #  x-end:  x, y, z; idiv
 0  1  0  1   #  y-end:  x, y, z; idiv
 0  0  1  1   #  z-end:  x, y, z; idiv
20 20 20       # #(grid points) in x, y, z dir.
NO             # DEP(HASING) | NO (POST-PROCESSING)
ANG   LARGE    # units: ANG|ATU;  rel. comp.: LARGE|SMALL
 1     F       # Wannier function index; apply WF rotation?
```

end of *case*.inwplot

line 1 `mode, switch` *format(A3, A1)*

`mode`	*n*`D`	with $n = 0, \ldots, 3$, an n-dimensional grid will be specified
	`ANY`	read arbitrary grid from *case*`.grid`
`flag`	`O,N`	when `mode = `*n*`D`: check for orthogonality or not
	`C,F`	when `mode = ANY`: cartesian or fractional coordinates

If `mode=`*n*`D` and `flag=O` or absent, grid axes will be checked for pairwise orthogonality; set `flag=N` for nonorthogonal axes. If `mode=ANY` and `flag=C` or absent, the grid points in *case*`.grid` are in cartesian coordinates (3 reals per line); if `flag=F`, they are in fractional coordinates (3 numbers and divisor).

line 2 `ix, iy, iz, idv` *(free format)* — grid origin

`ix,iy,iz,idv`	*int*	Coordinates of the origin of the grid, where x=ix/idv etc. in units of the *conventional* lattice vectors

line 3 `ix, iy, iz, idv` *(free format)* — axis end-point *(n times)*

`ix, iy, iz, idv`	*int*	Coordinates of the end points of each grid axis

If `mode=`*n*`D`, **l. 3** must be given n times, once for each direction; if `mode=ANY`, it must be omitted.

line 4 `nx, ny, nz` *(free format)* — mesh size

`nx,ny,nz`	*int*	number of mesh points in each direction

line 5 `post` *format(A3)* — post-processing option

`post`	`DEP`	"dephasing": each wave function is multiplied by a complex phase factor to align it (as far as possible) to the real axis
	`NO`	No post-processing

line 6 `unit, whpsi` *(free format)*

`unit`	`ANG`	Ångström
	`AU`|`ATU`	Atomic units

`whpsi`	`LARGE`	the large relativistic component for each wave functions is evaluated
	`SMALL`	small relativistic component

line 7 `iwan, wfrot` *(free format)* — WF index, apply WF rotation?

`iwan`	*int*	index of the WF to be plotted, unless overridden on command line
`wfrot`	*logical*	read unitary matrix from *case*`.wfrot` and apply before plotting

Finding the right window for plotting can be tricky. WANNIER90 often yields orbitals that are not centered in the home unit cell; `wplot2xsf` can shift them later on, but in `wplot` one has to "hit" the orbitals as given by WANNIER90 (see section "Final State" in *case*`.wout`). Therefore, it is recommended to start with a coarse grid (for instance $10 \times 10 \times 10$), make sure the window is correct, and only then start a calculation with a finer grid.

The option `wfrot` asks `wplot` to read a $(N_{\mathrm{WF}} \times N_{\mathrm{WF}})$ matrix from *case*`.wfrot`, which is applied to the WFsbefore plotting. This is useful for some post-processing applications, where a rotation in the Wannier subspace is desired.

2.3 `prepare_w2wdir` — copy required files

A WIEN2k computation can be the starting point for various runs of WIEN2WANNIER. This script creates a new subfolder of a WIEN2k directory and is invoked by

```
prepare_w2wdir subdir
```

where `subdir` is the name of the subdirectory to be created.

2.4 `init_w2w` — prepare wien2wannier input files

This script guides the user through the initialization of WIEN2WANNIER phase as described in Section 1.2.

```
init_w2w [-up|-dn] [-b options]
```

In batch mode (`-b`), further options are available instead of interactive input.

2.5 `findbands` — get band indices inside energy interval

This program reads *case*`.output1` to identify the bands which lie within a certain energy window. The program `findbands` is executed by issuing the command:

```
x findbands window [-up|-dn -so -hf -efermi EF] |
findbands def
```

window may be given as `-emin e -emax E` or `-all e E`. The energies are in eV with $E_{\mathrm{F}} = 0$. The Fermi energy is read from *case*`.fermi` (written by `prepare_w2wdir`) unless given as `-efermi` (in Ry).

The output is given in *case*`.outputfind` and consists of the bands in the interval at each k-point, as well as which bands are contained in the interval across all k-points, and which bands cross the interval at any k-point.

2.6 write_inwf — prepare input file for w2w

This program prepares the main input file *case*.inwf for w2w. It is executed by invoking the command

```
write_inwf (interactive) |
write_inwf -bands Nmin Nmax PROJ [PROJ ...] (noninteractive)
```

It will ask (in interactive mode) for a range of bands, which are all the bands to be taken into account by w2w (including those for disentanglement). Then, it will ask for "projection specifications" `PROJ = SITE:ORB[:ZAXIS[:XAXIS]]`, which consist of colon- separated parts naming an atomic site, an orbital, and, optionally, rotated z- and x-axes[1]. Please see `write_inwf -h` and `write_inwf -H axes/sites/orbitals` for further information on these. Interactively, you can also use tab completion to discover input options and command line history to recall past inputs.

The program will keep asking for projections until it has accumulated as many projections as bands were specified. However, one can stop early by pressing `Ctrl-D` (EOF); in this case, there will be fewer Wannier functions than bands (disentanglement).

2.7 write_win — prepare input file for wannier90.x

This program reads *case*.inwf, *case*.klist, and some other files, and produces *case*.win, the input file for wannier90.x. It is executed by invoking the command

```
write_win [-fresh]
```

If *case*.win already exists, write_win updates it. Otherwise (or if -fresh is given), the template in $WIENROOT/SRC_templates is used.

2.8 wannier90 — wrapper for wannier90.x

A wrapper script for wannier90.x is provided to take care of spin polarization and spin-orbit coupling. In the context of WIEN2WANNIER, wannier90.x is executed by invoking the command

```
x wannier90 [-up|-dn|-so|-pp] | wannier90 [-up|-dn|-so|-pp]
```

The wrapper script must be able to find the executable wannier90.x, i.e. you have to either have it in your $PATH, or edit the script wannier90_lapw to provide the location.

With -pp, wannier90.x will produce *case*.nnkp; with -so, w2waddsp will be called to add the spin channels together before running wannier90.x.

Extensive diagnostic output goes to *case*.wout, error messages to *case*.werr; the full results (in particular the $U(k)$ matrices) are stored in the binary file *case*.chk.

2.9 w2waddsp — add spin directions for SOC

In calculations including spin-orbit coupling (SOC), WANNIER90 has to be invoked on *case*.mmn and *case*.amn files which contain the sum of the overlaps / projections for the two spin

[1] write_inwf uses the method of Ref. [7] to rotate the spherical harmonics.

channels. (This procedure is needed also for non-spin-polarized cases, cf. Section 2.1.) To this end, w2waddsp reads *case*.mmn{up,dn}, *case*.amn{up,dn} and writes *case*.mmn, *case*.amn.

There is usually no need to call this program manually, it is run by `x wannier90 -so`. If needed, it is executed by invoking the command

```
x w2waddsp | w2waddsp def
```

after running `x w2w-up` and `-dn`.

2.10 write_inwplot — prepare input file for wplot

This program may help in preparing the input file for wplot. (Alternatively, copy the template from $WIENROOT/SRC_templates.) It is executed by invoking the command

```
write_inwplot case
```

2.11 wplot2xsf — convert wplot output to xsf format

This program converts the files *case*_*m*.psink and *case*_*m*.psiarg produced by wplot to files *case*_*m*.xsf which can be opened, e.g., in XCrySDen [8] or VESTA [9].

Note that only real data can be represented in the xsf file. Therefore, $|w(\boldsymbol{r})|^2 \operatorname{sgn} \mathrm{Re}\, w(\boldsymbol{r})$ is saved by default. (In most cases the Wannier functions are real anyway.)

wplot2xsf has a number of options (see `wplot2xsf -h`), but usually it is executed by invoking the command

```
wplot2xsf [-up|-dn]
```

If all the required files have their standard names, this will convert all the plots in the current directory.

By default, wplot2xsf reads *case*_centres.xyz, and shifts the Wannier functions so that their centers have the coordinates given in that file. If translate_home_cell (and write_xyz) is set in *case*.win, this will result in a plot where the Wannier centers lie in the "home" unit cell.

2.12 convham — Fourier transform Wannier Hamiltonian

This program Fourier transforms the WANNIER90 real space Hamiltonian $H(\boldsymbol{R})$ (*case*_hr.dat) to its k-space representation $H(\boldsymbol{k})$ (*case*.ham_fine) on the k-points given by *case*.klist. In this way, if the real-space Hamiltonian is sufficiently localized, $H(\boldsymbol{k})$ may be interpolated to arbitrarily fine k-grids. convham is executed by invoking the command

```
x convham [-band] |
convham def
```

2.13 shifteig — shift energies in *case*.eig

to zero in *case*.eig. If needed, this program can be used to apply an additional shift by DE (in eV). shifteig is executed by invoking the command

```
x shifteig [-up|-dn] -efermi DE | shifteig def DE
```

2.14 rephase

program reads the psiarg files to determine the most probable phases of all Wannier functions. Then, the input for the interface file *case*.inwf is rewritten in a way that a subsequent run of WIEN2WANNIER and WANNIER90 leads to real Wannier orbitals (this does not work in all cases). The Fortran program rephase is invoked by the command

```
rephase case [-w] [-up/-dn] [-wf=idx_wann]
```

where the option -w means that *case*.inwf is automatically updated (future wien2wannier runs will then have a higher probability of producing real Wannier orbitals) and the option -w idx_wann indicates that the action is only applied for the Wannier function idx_wann.

2.15 joinvec — join parallel vector files

energy files from a parallel calculation into one *case*.vector and one *case*.energy. It is executed by invoking the command

```
x joinvec [-up|-dn -c -so]
```

All .vector_* files are merged into one .vector file containing the header of the first file .vector_1 (and correspondingly for the .energy files).

2.16 write_insp — write input file for spaghetti

Fermi energy read from *case*.scf2 in the input file for spaghetti, *case*.insp. It is invoked by

```
write_insp [-up/-dn]
```

3 Getting help

Online help for all programs can be requested via the option `-h`. In addition to the pointers below, the WIEN2WANNIER distribution also includes a file `CHEATSHEET`, which concisely summarizes the usual steps of a calculation. If this does not help, send questions relating to the usage of WIEN2WANNIER to the Wien2k mailing list at `http://www.wien2k.at/reg_user/mailing_list/`. For questions about WANNIER90 and the MLWF procedure, there is a mailing list at `http://www.wannier.org/forum.html`. The WIEN2WANNIER maintainer can be reached at `wien2wannier@ifp.tuwien.ac.at`.

3.1 FAQ

How does one choose the initial projections?

There is no general rule to choose the initial projections which have to be prepared in the `inwf` file. There are, however, some hints which usually help:

- In the energy interval of interest, identify the main atoms which contribute to the partial DOS. These atoms are usually good centers for the initial projections. The character of these atomic contributions can also be seen via the partial DOS.
- A glance at the bandstructure often helps to consider the multiplicity of equivalent WF and to identify certain hybridizations.

The resulting Wannier orbitals are not localized

Several possible reasons (this list is not complete)

- choosing non-optimal initial projections
- the number of k-points is too small

The bandstructure of Wien2k and wannier90 does not match

First of all, be sure to take into account the Bohr-Ångström conversion, e.g., in `gnuplot`:

```
p 'case.spaghetti_ene' u ($4/0.53):5, 'case_band.dat' w l
```

If the bands themselves differ strongly, then one might have

- non-optimal initial projections
- too few k-points
- chosen the frozen energy window in WANNIER90 in a wrong way.

When plotting a Wannier orbital in XCrySDen I can see nothing

One possible reason is that the WF centers are not necessarily in the home unit cell at $[0\,0\,0]$. The interface programs usually account for this by translating the WF to the home cell. However, the spatial mesh for `wplot` has to be defined with respect to the original centers which come out of WANNIER90. These centers can be found in *case*`.wout` file. Assume the WF is centered at $[-0.5\,-0.5\,;-0.5]$ in the basis of conventional unit vectors. Then, appropriate mesh vectors might be

```
-1 -1 -1 1     #x, y, z, divisor of orig
 0 -1 -1 1     #x, y, z, divisor of x-end
-1  0 -1 1     #x, y, z, divisor of y-end
-1 -1  0 1     #x, y, z, divisor of z-end
```

My Wannier orbitals are not centered at the home unit cell

It happens quite frequently that a WF is not centered within the home unit cell, which is displayed by default by XCrySDen. The interface program `write_win` automatically activates the option `translate_home_cell` in the WANNIER90 input file. Then, WANNIER90 should produce a file *case*`_centres.xyz`, where the vectors of all WF are stored which translate the orbital to the home unit cell. If this file cannot be located by `wplot2xsf` or the option –noshift is activated, no translation is conducted and the WF appear centered at their original position.

Bibliography

[1] P. Blaha, K. Schwarz, G.K.H. Madsen, D. Kvasnicka, and J. Luitz. WIEN2*k, An Augmented Plane Wave + Local Orbitals Program for Calculating Crystal Properties.* Techn. Universität Wien, Austria, 2001. `http://www.wien2k.at`.

[2] P. Blaha, K. Schwarz, G.K.H. Madsen, D. Kvasnicka, J. Luitz. WIEN2*k User's Guide.* `http://www.wien2k.at/reg_user/textbooks/usersguide.pdf`.

[3] K. Held. *Electronic structure calculations using dynamical mean field theory.* Adv. Phys. **56**, 829-926 (2007).

[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. **178**, 685 (2008)

[5] A.A. Mostofi, J.R. Yates, Y.-S. Lee, I. Souza, D. Vanderbilt and N. Marzari. WANNIER90 *User Guide.* `http://wannier.org/doc/user_guide.pdf`.

[6] J. Kuneš, R. Arita, P. Wissgott, A.Toschi, H. Ikeda, and K. Held. WIEN2WANNIER*: From linearized augmented plane waves to maximally localized Wannier functions.* Comput. Phys. Commun. **181**, 1888 (2010).

[7] Z. Romanowski and S. Krukowski. *Transformations of complex spherical harmonics under rotations.* J. Phys. A: Math. Theor. **40**, 15071 (2007).

[8] A. Kokalj. *Computer graphics and graphical user interfaces as tools in simulations of matter at the atomic scale.* Comput. Mater. Sci., **28**, 155 (2003).; `http://www.xcrysden.org`.

[9] K. Momma and F. Izumi. *VESTA 3 for three-dimensional visualization of crystal, volumetric and morphology data.* J. Appl. Cryst. 44, 1272 (2011). `http://jp-minerals.org/vesta/en/doc.html`.