Wannier functions tutorial: NiO

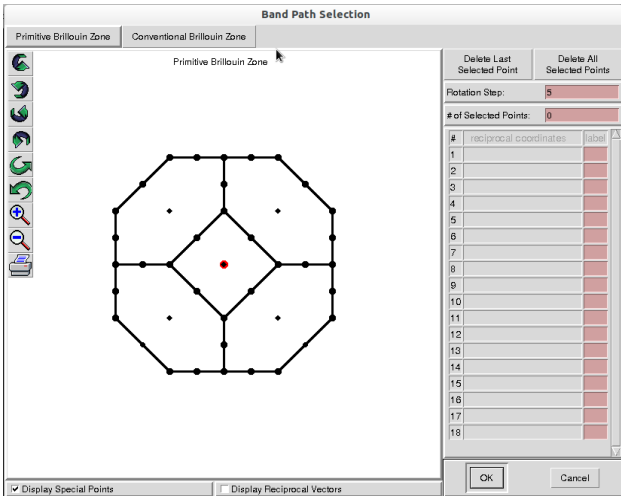
This tutorial will go through the process of using the exciting-plus code to generate Wannier functions (WFs), plot band structures derived from WFs, and plot WFs (2D and 3D), using NiO as our sample material. NiO has the same structure as NaCl (space group 225)

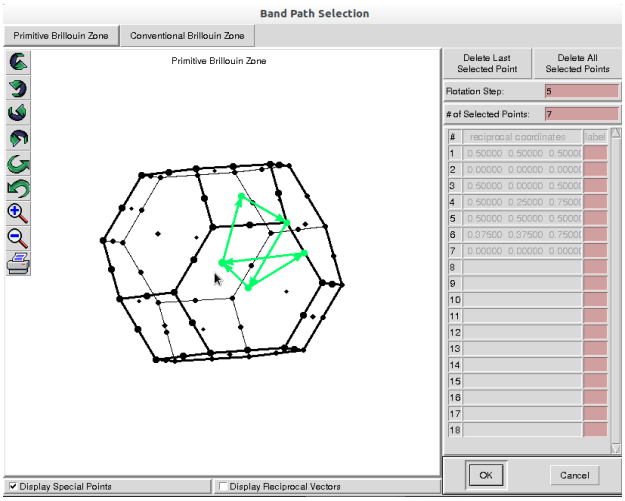
We will begin with a standard ground state calculation in the Bloch basis. First, we will use task 190 to generate a crystal.xsf file that we can import into XCrysDen to choose a path through the Brillouin Zone for the band structure plot. A sample elk.in file is given below:

```
! NiO

tasks
   0
   190


nempty
  20

avec
 0.0 0.5 0.5
 0.5 0.0 0.5
 0.5 0.5 0.0

scale
  7.89357814

atoms
   2                              : nspecies
'Ni.in'                           : spfname
   1                              : natoms; atposl, bfcmt below
    0.0 0.0 0.0    0.00000000  0.00000000  0.00000000
'O.in'                            : spfname
   1                              : natoms; atposl, bfcmt below
    0.50000000    0.50000000    0.50000000    0.00000000  0.00000000  0.00000000

ngridk
  16 16 16

autormt
 .true.
```

At the conclusion of this run, we import the crystal.xsf file into XCrysDen and choose Tools →k-path Selection to bring up a rotatable 3D model of the BZ. Using this display, we will chart the path

L → Γ → X → W → L → K → Γ as shown below:





We see from XCryDen that the primitive cell coordinates for our BZ path are as follows:

L	0.5, 0.5, 0.5
Γ	0.0, 0.0, 0.0
X	0.5, 0.0, 0.5
W	0.5, 0.25, 0.75
L	0.5, 0.5, 0.5
K	0.375, 0.375, 0.75
Γ	0.0, 0.0, 0.0

We will now rerun our calculation with the following changes to the elk.in file:

- We will change task 0 (ground state calculation) to task 1 (resume ground state calculation using the density in STATE.OUT) and set the max self-consistent loops allowed to 1:

```
maxscl
1
```

- We add tasks 822 (Bloch basis band structure calculation) and 811 (Bloch basis density of states calculation)
- We will add the following text block to the elk.in file to specify our path though k-space for the

band structure calculation:

```
plot1d
7                   ! # of vertices along the path
100                 ! # of k-points to be sampled
0.5, 0.5, 0.5 : L   ! coordinates of each vertex along the path
0.0, 0.0, 0.0 : G
0.5, 0.0, 0.5 : X
0.5, 0.25, 0.75 : W
0.5, 0.5, 0.5 : L
0.375, 0.375, 0.75 : K
0.0, 0.0, 0.0 : G
```

After the calculation finishes running, we will have the files “bands.dat” and “tdos.dat” in the working directory.

Note that in “bands.dat” and “tdos.dat”, the energy values are in eV, and the Fermi energy, found in EFERMI.OUT, has already been subtracted from the band energies, setting the Fermi energy to zero in both files! In the capitalized files, such as EFERMI.OUT and BAND.OUT, energy values are in Hartrees and the Fermi energy has not been set to zero!

Before we plot the band structure and the dos, we will decompose the total density of states into the partial density of states of p and d character. To do this, we use the pdos utility included with exciting-plus. (To build this executable, navigate to the top level directory of the exciting-plus source directory, then go to /utilities/pp and run make).

When pdos is run, it will ask you to “Input orbitals for pDOS”. At this prompt, you will specify the set of orbitals by

A, B, C

where A = <number of the atom as given in the elk.in file>

B = <letter denoting the angular momentum (s,p,d, or f)>

C = <number denoting the spin (1 for spin up, 2 for spin down)>

After you successfully enter this, you will be asked to input another orbital set. At this point, you may enter 'q' to quit, or enter additional orbital sets which will be added to produce the partial density of states for the sum of the orbital groups entered. The resulting partial density of states will be written in the file “dos.dat” in the working directory. In the example below, we create a file consisting of the partial density of states of p-character for the system.

```
Number of species :    2
Number of atoms :    2
Number of spins :    1
Input orbitals for pDOS
```

```
1,p,1
Input orbitals for pDOS
2,p,1
Input orbitals for pDOS
q
spin : 1
  atom :    1 orb :  0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0
  atom :    2 orb :  0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0
```

Now we shall use gnuplot to plot the band structure along with the density of states an example gnuplot script is included in Appendix A).

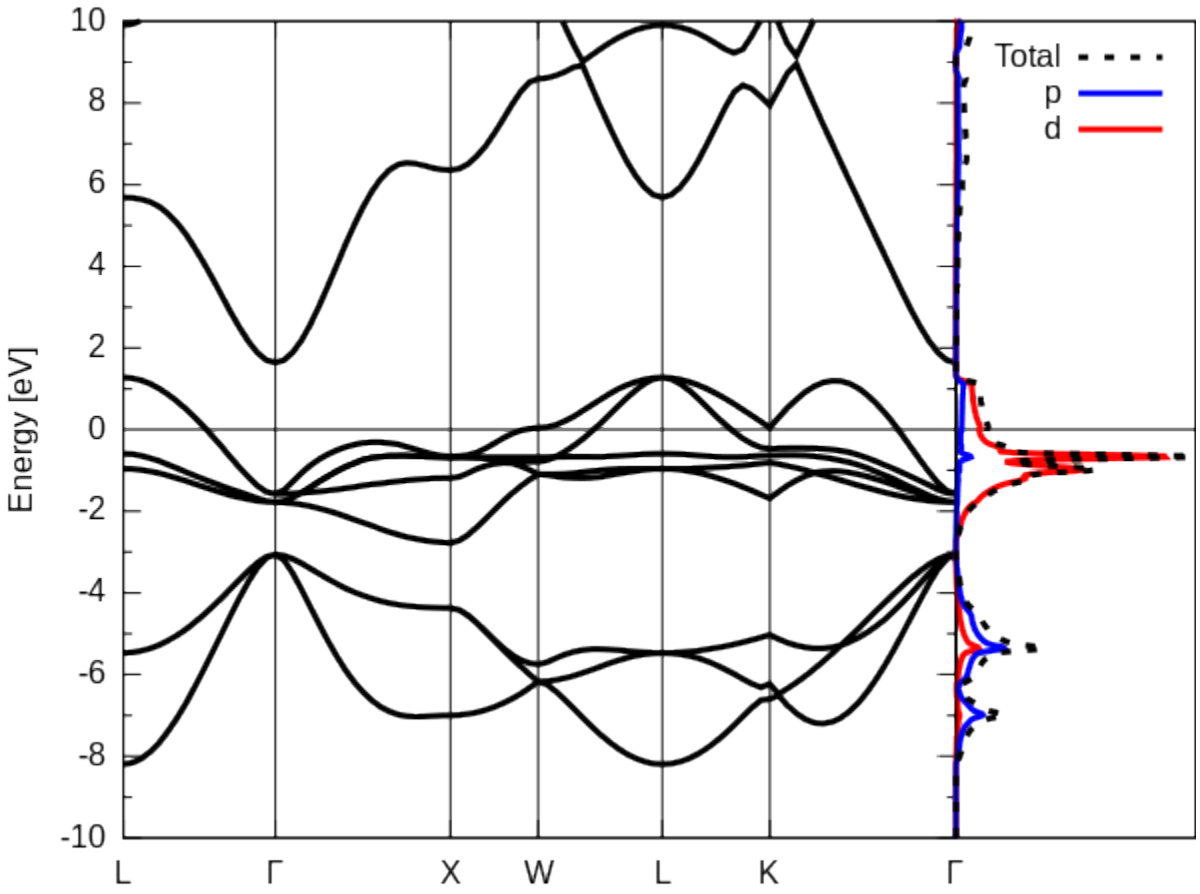
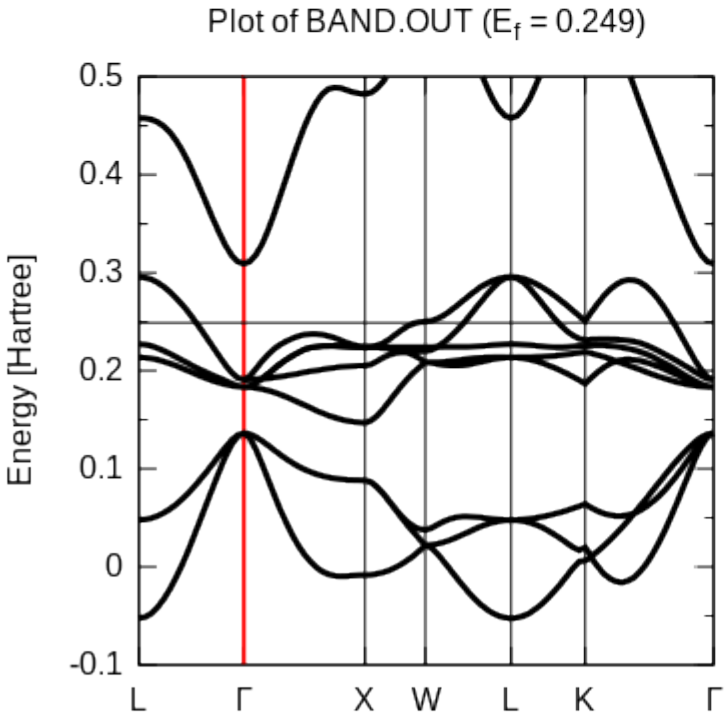


Fig 1: Band structure and dos for NiO (Bloch basis)

What we see in this plot is a group of predominantly d-character bands near the Fermi energy (zero), as well as a few predominantly p-character bands lying lower in energy. These are the states we will use to construct our Wannier basis for this example.

In order to generate WFs, we must rerun our calculation using wannier keyword, as well as tasks 822 and 811 to calculate the band structure and density of states in the Wannier basis. The syntax for the wannier keyword is given in Appendix B, and a sample wannier keyword block will be provided shortly. The most important thing we must determine when generating WFs is which bands we want to provide to the wannier keyword bloch. We have selected the bands to be used from the above plot, but we must look at the EIGVAL.OUT output file to know which band numbers identify these bands within exciting-plus. We will look at the Γ point (**k** = 0) energy values of EIGVAL.OUT to determine which band numbers correspond to our selected bands. As we noted earlier, the energy values in EIGVAL.OUT are give in Hartrees, and the Fermi energy has not been subtracted from the values. The same is true for BAND.OUT, which differs from "bands.dat" only in that respect. Therefore we will compare a plot of BAND.OUT with our plot of bands.dat as we look at EIGVAL.OUT. A plot of BAND.OUT as well as a partial listing of EIGVAL.OUT from this calculation is given below:



145 : nkpt
31 : nstsv

1 0.000000000 0.000000000 0.000000000 : k-point, vkl
(state, eigenvalue and occupancy below)

1	-2.068745583	2.000000000
2	-2.068745558	2.000000000
3	-2.068745558	2.000000000
4	-0.4920887996	2.000000000
5	**0.1364936564**	**2.000000000**
6	**0.1364936855**	**2.000000000**
7	**0.1364936861**	**2.000000000**
8	**0.1834447888**	**2.000000000**
9	**0.1834448080**	**2.000000000**
10	**0.1834448084**	**2.000000000**
11	**0.1909729555**	**2.000000000**
12	**0.1909729605**	**2.000000000**
13	0.3090539965	0.000000000
14	0.9080598093	0.000000000
15	0.9604565777	0.000000000
16	0.9604565778	0.000000000

Comparing with our plot of band.out, we see that the bands we are interested in are numbered 5—12 (in bold in the EIGVAL.OUT sample above)

We will now add the following wannier block to our elk.in file:

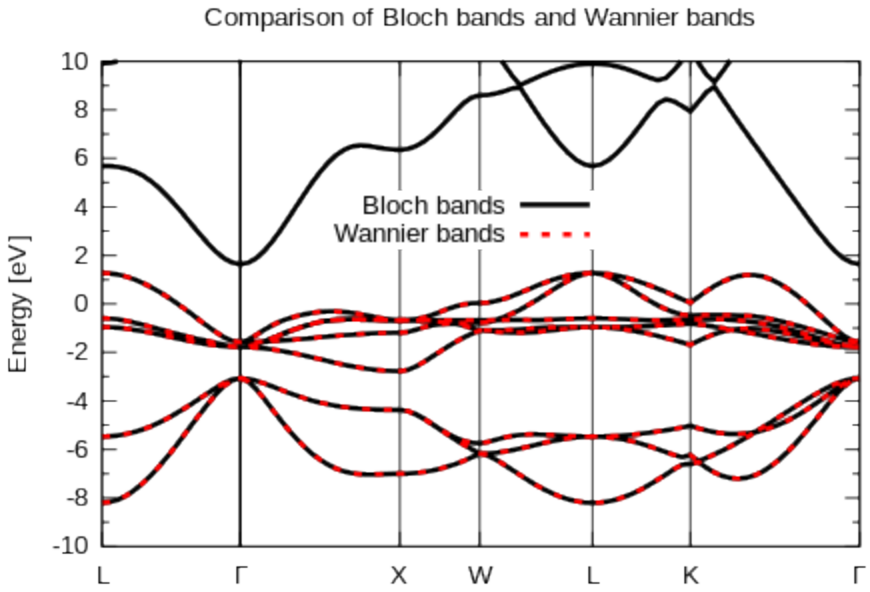
```
wannier
.true.          ! turn on Wannier functions generation
.false.         ! .true. -> add potential correction to WF states (always use .false.)
2 2 1           ! number of atoms with WF, number of orbital groups, number of WF types
5               ! number of orbitals in group#1 (d-orbitals)
5 6 7 8 9       ! orbitals for projection
1 1 1 1 1       ! pure spinor for projection (1 - (phi^{up}, 0),  2 - (0,phi^{down}) )
1 1 1 1 1       ! type of WF
3               ! second group - 3 p-orbitals of oxygen
2 3 4
1 1 1
1 1 1
5 12 0.0        ! for type#1: band interval N1,N2 and potential correction
1 1             ! (atom, orbital) assign to atom #1 orbital group #1 (Ni-3d)
2 2             ! (atom, orbital) assign to atom #2 orbital group #2 (O-2p)
```

Following the syntax of the wannier block, we are specifying that we will have two atoms with WFs centered on them (Ni and O), two orbital groups used (in our case, one group with be p-character orbitals and the other will be d-character orbitals), and 1 WF type (i.e. only one energy/band range will be used).

When listing the orbitals to be used for projection, refer to the table below (see also Appendix B):

s	p y	p z	p x	d xy (t_{2g}) m = -2	d yz (t_{2g}) m = -1	d $3z^2$-r^2 {or} z^2 (e_g) m = 0	d xz (t_{2g}) m = +1	d x^2-y^2 (e_g)
1	2	3	4	5	6	7	8	9

Once this calculation completes, there will be a new file named bands_wann.dat. This file has the same format as bands.dat, but the band structure in bands_wann.dat was calculated from the Wannier basis we constructed from a subset of the Bloch states.


Comparison of Bloch bands and Wannier bands
Energy [eV]
10
8
6
4
2
0
-2
-4
-6
-8
-10
L
Γ
X
W
L
K
Γ
Bloch bands
Wannier bands

In the energy range of the subset of the Bloch bands used to generate our Wannier functions, there is good agreement between the bands calculated from the Wannier basis and the bands calculated from the Bloch basis.

Our next task is to generate the partial density of states from the Wannier basis. We will once again use the pdos utility. This time, when we are asked if we want a Wannier DOS, we will say true. We will then be asked for which Wannier functions should be included in the density of states file, similarly to when we generated a partial density of states file for the Bloch basis. In this case, we enter the desired WFs by their number, found in WANNIER.OUT, followed by a '/' and enter. This will create the file 'dos.dat' in the working directory. The order of the listed WFs in WANNIER.OUT should follow the ordering given in the elk.in file. Since we listed the d-orbitals first in the input file, WFs 1-5 in WANNIER.OUT correspond to orbitals 5-9 (d_{xy} through $d_{x^2-z^2}$) in the above table, and WFs 6-8 correspond to orbtials 2-4 (p_x, p_y, and p_z) . We will use bands 6, 7, and 8 to generate the pdos for the p-character orbitals centered on the oxygen atom, bands 1, 2, and 4 to generate the pdos for the t_{2g} d-character orbitals centered on the nickel atom, and bands 3 and 5 to generate the pdos for the e_g d-character orbitals centered on the nickel atom:

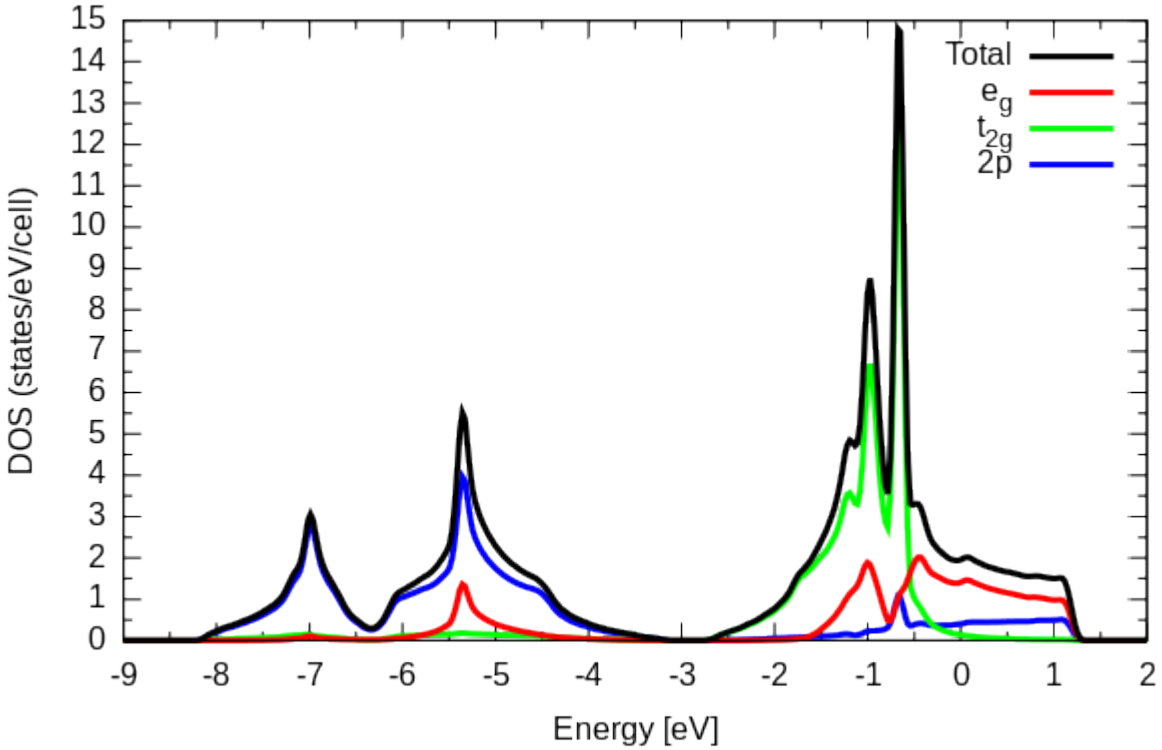


Fig 2: Total and partial density of states for NiO. Partial density of states is computed for WFs with given symmetry.

+

Appendix A

Gnuplot script for generating band structure and dos plots

```
#!/usr/bin/gnuplot
# gamma is 0393

clear
reset
unset multiplot
set term svg dash enhanced
#set term pdfcairo dash enhanced
set out "banddos_b.svg"
#set out "banddos_b.pdf"
libsans =  "Liberation Sans, 12"
set size 0.64
set tmargin 2
set multiplot
set origin 0,0
set size 0.8, 1
set rmargin 0
#set tmargin 2
#set yrange[-25:25]
set key left top
ymax = 10.0
ymin = -10.0
set ylabel "Energy [eV]"
set yrange[ymin:ymax]
set xrange[0:3.777885309]
lpt1 = 0.0
gpt1 = 0.6893449328
xpt1 = 1.485331898
wpt1 = 1.883325380
lpt2 = 2.446173161
kpt1 = 2.933613638
gpt2 = 3.777885309

set xtics("L" lpt1 ,\
        "Γ" gpt1 ,\
        "X" xpt1  ,\
        "W" wpt1  ,\
        "L" lpt2  ,\
        "K" kpt1  ,\
        "Γ" gpt2  \
        )
set arrow 1 from gpt1, ymin to gpt1, ymax lw 1 lt 1 lc rgb 'black' nohead
set arrow 2 from xpt1, ymin to xpt1, ymax lw 1 lt 1 lc rgb 'black' nohead
set arrow 3 from wpt1, ymin to wpt1, ymax lw 1 lt 1 lc rgb 'black' nohead
set arrow 4 from lpt2, ymin to lpt2, ymax lw 1 lt 1 lc rgb 'black' nohead
set arrow 5 from kpt1, ymin to kpt1, ymax lw 1 lt 1 lc rgb 'black' nohead
set xzeroaxis lw 2 lt 0
```

```
plot "bands.dat" w lines lw 3 lt 1 lc rgb "black"
#------------------------------------
unset arrow 1
unset arrow 2
unset arrow 3
unset arrow 4
unset arrow 5
unset ylabel
set key top right invert
set lmargin 0
set origin 0.8, 0
set size 0.2, 1
set noytics
set xtics("" 0.0)
set xrange[0:15]
plot "dos_d_bloch.dat" u 2:1 w lines lw 3 lt 1 lc rgb 'red' title 'd',\
 "dos_p_bloch.dat" u 2:1 w lines lw 3 lt 1 lc rgb 'blue' title 'p',\
 "tdos.dat" u 2:1 w lines lw 3 lt 2 lc rgb 'black' title 'Total'
```

Appendix B

Exciting-plus instructions for creating plots of Wannier functions

In order to create plots of the Wannier functions, you will need to run exciting-plus using task 863 (for 3D plots) or task 862 (for 2D plots), along with the wannier and wannier_plot keyword. The wannier_plot keyword generates information for plotting the functions and takes different values depending on whether you wish to make 2D or 3D plots, but we shall discuss the creation of the Wannier functions themselves, handled by the wannier keyword, first. The wannier keyword takes the following values, as show in in this table.

Note that when more than one parameter name is listed in a cell, it means that these parameters are to be entered one after the other in the order given, on the same line, with spaces inbetween. "WF" will be used to abbreviate "Wannier function" to save space.

Sub-tables with bold borders indicate that the required information will need to be repeated in the input file for each orbital group, WF, or atom as noted before proceeding to further input parameters.

wannier

<parameter name>	<what the parameter does>	Notes
wannier	turns on the wannier function generator	.true. to turn on the wannier function generator, .false. to turn it off (If this is set to false, there's no reason to use the wannier keyword at all)
wann_add_poco	adds potential correction to the wannier states	.true. to turn on the potential correction, .false. to turn it off. *(Scott advises to always use .false.)*
wann_natom wann_norbgrp wann_ntype	wann_natom – number of atoms with WFs centered on them. wann_norbgrp – number of orbital groups (i.e. p group, d group, etc.) wann_ntype – number of WF types	More than one WF type would be used when using two or more separate energy/band windows to generate WFs from. For example, if there is a case where the bloch state dos shows an energy region clearly dominated by one orbital type (p), a band gap, and a second energy region clearly dominated by a different orbital type (d), you might use two different energy windows to generate the WFs, corresponding to two WF types.
wann_norb(i)	The number of wannier orbitals in group i	3 for a p-orbital group, 5 for a d-orbital group, etc.

To be completed for each orbital group:

(unknown) wann_iorb(1,l,i) ?	the orbitals for projection (i.e. set of bloch orbitals to that the localized trial functions will be projected onto—the bloch orbitals from which this set of WFs will be made)	the orbitals are numbered as follows: s | p y | p z | p x | d xy (t_{2g}) | d yz (t_{2g}) | d $3z^2$-r^2 (e_g) | d xz (t_{2g}) | d x^2-z^2 (e_g) 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
(unknown) wann_iorb(2,l,i) ?	This line will contain a series of numbers separated by spaces, either 1 or 2, to denote spin. One number for each orbital given in the group	1 for spin up 2 for spin down
(unknown) wann_iorb(2,l,i) ?	This line will contain a series of numbers, separated by spaces, to denote the type of WF.	(I don't know what this means, but the example files only use 1's)

	One number for each orbital given in the group	

To be completed for each WF type:		
wann_eint(1,i) wann_eint(2,i) wann_v(i)	wann_eint(1,i) – band/energy interval start wann_eint(2,i) – band/energy interval end wann_v – potential correction	The first two parameters determine the start and end of the band/energy range, respectively. If an integer value is given, the range is assumed to be by bands (band numbers can be found in the EIGVAL.OUT file – checking values at the Γ point is probably easiest). If a float value is given, the range is assumed to be by energy (in Hartrees)

To be completed for each atom:		
wann_iprj(1,i) wann_iprj(2,i)	wann_iprj(1,i) – orbital group number wann_iprj(2,i) – atom number	This is the method to assign the specified orbital groups to each atom. The ordering of the atoms is specified in the atoms keyword. Two input values: First – atom number Second – orbital group

wannier_plot parameters for 3D plots(task 863)

wannier_plot

<table>
<tr><th><parameter name></th><th><what the parameter does></th><th>Notes</th></tr>
<tr><td>unknown</td><td>c1 c2 c3 – coordinates for the center of the 3D box volume to be plotted</td><td></td></tr>
<tr><td>unknown</td><td><table><tr><td>x1 y1 z1</td><td rowspan="3">the three vectors defining the 3D box volume to be plotted</td></tr><tr><td>x2 y2 z2</td></tr><tr><td>x3 y3 z3</td></tr></table></td><td>In the example file, the values were:<table><tr><td>10 0 0</td></tr><tr><td>0 10 0</td></tr><tr><td>0 0 10</td></tr></table></td></tr>
<tr><td>unknown</td><td>n1 n2 n3 – the number of points along each vector</td><td>In the example file, the values were: 40 40 40</td></tr>
<tr><td>unknown</td><td>w_n w_f – the number of WFs to plot and the index of the first WF</td><td></td></tr>
</table>

wannier_plot parameters for 2D plots(task 862)

wannier_plot

<table>
<tr><th><parameter name></th><th><what the parameter does></th><th>Notes</th></tr>
<tr><td>unknown</td><td colspan="2">c1 c2 c3 – coordinates for the central point of the 2D plot</td><td></td></tr>
<tr><td>unknown</td><td>x1 y1 z1</td><td>coordinates of the 1st vector defining the 2D plane of the plot</td><td>In the example file, the values were:
5 5 0
5 -5 0</td></tr>
<tr><td></td><td>x2 y2 z2</td><td>coordinates of the 2nd vector defining the 2D plane of the plot</td><td></td></tr>
<tr><td>unknown</td><td colspan="2">j1 j2 j3 – not used</td><td>In the example file, this line was commented as “not used”, but it was in there anyway. Possibly because the code expects another coordinate line here, but doesn't use it for 2D plots</td></tr>
<tr><td>unknown</td><td colspan="2">n1 n2 n3 – the number of grid points in each direction</td><td>In the example file, the values were : 140 140 1
(probably need 1 for the third value always for 2D plots)</td></tr>
<tr><td>unknown</td><td colspan="2">w_n w_f – the number of WFs to plot and the index of the first WF</td><td></td></tr>
</table>