How to add Wannier 90 files on *BinPo*

In the context of *BinPo*, the master files to create the slab Hamiltonians are those obtained as output of Wannier 90 (W90) that hold the transfer integrals in the MLWF basis. In W90, these files are called *seedname_hr.dat* and can be obtained by setting the option "write_hr = True" when running the program. We refer to them as W90 files. The following lines are intended to be a guide about how to add on *BinPo* new W90 files generated by the user.

Step 0: Usually, the way of obtaining the W90 files starts by running DFT calculations and using the outputs as input for W90 program. From the DFT calculations it is important to get some values that will be introduced in *BinPo*: the lattice parameters in Å, the highest occupied level (*HOL*, a.k.a. valence band maximum) and the lowest unoccupied level (*LUL*, a.k.a conduction band minimum), both of them in eV. After running W90 program you should inspect the *seedname_hr.dat* file (W90 file) and take note of the number of rows that you need to skip within the file to get to the columns with the transfer integrals, since by default there will be a header and additional lines written by W90 in such a file.

Step 1: Copy your file into the *~/Wfolder*

We suggest here to keep the file termination with the extension as "_hr.dat" and if changing the *seedname*, make it to match the 'name' main key describe in the next step.

Step 2: Edit the *BPdatabase.py*

Basically, the *BPdatabase.py* module is a Python dictionary containing all the materials properties derived from DFT + W90 calculations. These properties can be accessed by specific keys. In that way, they are called by any *BinPo* components just by importing the module, which is not runnable by itself. Let's see the snippet for STO as an example:

```
'STO' : {'description' : 'cubic SrTiO3, Ti t2g manifold',
         'pseudopotential' : 'PAW PBE full-relativistic from PSlibrary',
         'unit-cell' : 'cubic',
         'Wannier_functions_number' : 6,
         'lattice_parameter' : 3.9425,
         'highest_occupied_level' : 9.6404,
         'lowest_unoccupied_level' : 11.4685,
         'Wannier_file' : 'STO_hr.dat',
         'skip_rows' : 228,
         'manifold' : 't2g',
         },
```

At present, apart from the first two items ('description' and 'pseudopotential', which are purely informative) in the so-defined STO dictionary, the rest of items are mandatory. For the case of hexagonal structures, it is also needed a 'lattice_parameter_c' item (see below). In the *BPmodule.py*, below the current available materials, we pointed out the meaning for each of the items as we reproduce in here:

```
# 'name' : {'description' : 'Details of the system.',
#           'pseudopotential' : 'Details of the pseudopotential used in DFT calculations.',
#           'unit-cell' : 'Unit-cell symmetry (at present, 'cubic' or 'hexagonal').',
#           'Wannier_functions_number' : 'Number of WFs, it must match the number in the W90 file.',
#           'lattice_parameter' : 'Lattice parameter a for cubic or hexagonal cell in Angs.',
#           'lattice_parameter_c' : 'Lattice parameter c in Angs if the unit-cell is hexagonal.',
#           'highest_occupied_level' : 'Valence band maximum in eV.',
#           'lowest_unoccupied_level' : 'Conduction band minimum in eV.',
#           'Wannier_file' : 'Name for the W90 file with extension. Usually name_hr.dat.',
#           'skip_rows' : 'Number of rows to skip within the W90 file.',
#           'manifold' : 'Manifold of bands for the system. At present 't2g' or 'other'.',
#          },
```

As the user can notice, for the time being some parameters are limited like 'unit-cell' and 'manifold'. The possibilities here will be extended in future versions.

So that, all the user has to do in to add a Python dictionary, inside the main materials dictionary of the *BPmodule.py* with the details corresponding to the W90 file saved to the *~/WFolder* in the previous step.

Step 3: Edit the *BPmodule.py*

WARNING: This is the main *BinPo* module, the user must be very cautious to only modify the section we remark here if the goal is just to add a file without altering the code functioning.

After successfully applying the previous step, the user needs to add the new material information on the *CellGenerator* method in *BPmodule.py*. Firstly, this method generates the cell object for the bulk by using the ASE *Atoms* module. Secondly, the cell according to the confinement direction is created by means of *surface* method from the ASE *build* module. It will be used mainly in the slab construction, as well as the definition of the 2D Brillouin zone. As examples, here below we show snippets of the KTO and BiTeBr cases.

KTO example:

```
elif material == 'KTO':
    # created the cubic perovskite
    KTO = Atoms('KTaO3',
            cell = [a0,a0,a0],
            pbc = True,
            scaled_positions = ((0,0,0),(0.5,0.5,0.5),(0.5,0.5,0),(0.5,0,0.5),(0,0.5,0.5))
            )
    # create a Cell ASE object according to the orientation 'hkl'
    if face in ['100','010','001']:
        KTO001 = surface(KTO, (0, 0, 1), 1, periodic = False)
        return KTO001.cell
    if face in ['110','101','011']:
        KTO110 = surface(KTO, (1, 1, 0), 2, periodic = False)
        return KTO110.cell
    if face == '111':
        KTO111 = surface(KTO, (1, 1, 1), 3, periodic = False)
        return KTO111.cell
```

Given that in this example, we have a t_{2g} cubic perovskite, we can see that the orientation plays an important role to create the slab system along a specific confinement direction.

BiTeBr example:

In this case, rotations are not allowed and just the [001] direction is taking into account.

```
elif material == 'BTB':
    BTB = Atoms('BiTeBr',
                cell = [4.2662, 4.2662, 6.487, 90.0, 90.0, 120.0],
                pbc = True,
                scaled_positions=((0.33, 0.67, 0.93), (0.00, 0.00, 0.658), (0.67, 0.33, 0.230))
                )
    if face == '001h':
        BTB001 = surface(BTB, (0,0,1), 1, periodic = False)
        return BTB001.cell
```

IMPORTANT: In any case, when creating the bulk cell by using *Atoms* module, the scaled positions are not important, they are well-specified in there for future applications. However, it is fundamental to correctly introduce the cell parameter, which in this case is defined as cell = [*a*, *b*, *c*, α, β, γ], being *a*, *b* and *c* the lattice parameters and α, β and γ the angles between the lattice vectors. Alternative definitions of the cell are possible. We refer the user to the ASE documentation for more information (https://wiki.fysik.dtu.dk/ase/ase/atoms.html).

Step 4: The new W90 file is already successfully added to *BinPo*. To call it, just set the *BinPo* material keyword in the preprocessing (*-mt*) and the self-consistent step equal to the 'name' that was previously defined within the *BPdatabase.py* module, as seen in Step 2.

Final note: To report bugs of asking some questions about this procedure do not hesitate to contact us. Good luck!