**Quantum Espresso steps**

1) Google "[material] cif file", select "materialsproject.org" link. Download Primitive Cell cif format

2) Go to: https://www.materialscloud.org/work/tools/qeinputgenerator

3) Upload cif file

4) Select formats "CIF file - pymatgen", "SSSP Efficiency PBE", "non-magnetic metal", "normal" and download zip file

5) Run Quantum Espresso

1. SCF
2. NSCF
3. BANDS
4. DOS

Look at the input file (scf). It is composed of three namelists:

* &CONTROL
* &SYSTEM
* &ELECTRONS

Followed by three cards:

* ATOMIC\_SPECIES
* ATOMIC\_POSITIONS
* K\_POINTS

# Quantum Espresso:

Steps to perform DFT

1. **SCF** calculation
2. **NSCF** calculation where same prefix and outdir are used as in the preceding scf calculation, a denser k-point mesh is specified

You change (in the mentioned namelist and cards) mainly:

&CONTROL

calculation = ‘nscf’

verbosity = ‘high’

&SYSTEM

nosym = .true.

KPOINTS

*Use a denser k-point mesh*

1. **BANDS** (you just change the following from the SCF file)

&CONTROL

calculation = ‘bands’

&ELECTRONS

diago\_full\_acc = .true.

KPOINTS

*(of your desired path in the crystal structure)*

*Note on how to select your desired path:*

* *Open xcrysden*
* *File > Open PWscf > chose either input/output file*
* *Tools > k-path Selection*
* *Choose your path by clicking on each point one by one*
* *Press “OK”*
* *Add number in “Total number of k-points along the path”. For example 150*
* *Put file name, select file type “pwscf”*
* *Open newly created pwscf file, copy whole content and paste it at the end of BANDS file (on the KPOINTS card)*

1. **DOS**

**Code to run:**

**SCF simulation:**

pw.x < seedname.scf.in > seedname.scf.out

**NSCF simulation:**

pw.x < seedname.nscf.in > seedname.nscf.out

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

After SCF and NSCF simulation. Create a copy of entire folder to perform BANDS and DOS (separately) as performing either of them and then the other, it will result in error. This is caused because BANDS (or DOS) modify the tmp folder with their own pattern (which makes it impossible to perform DOS followed by BANDS. So it’s better to make 3 copies of your folder (after SCF and NSCF). One perform BANDS, second to perform DOS and third to keep it as a “standard” SCF NSCF completed. Once BANDS and DOS are completed in their own folder, copy the BANDS (and DOS) folders into the main “standard” SCF NSCF completed. So afterwards you have one main one with both done.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Bands simulation:**

pw.x < seedname.bnd.in > seedname.bnd.out

bands.x < seedname.bands.in > seedname.bands.out

gnuplot spaghetti.gp

**DOS simulation:**

dos.x < seedname.dos.in > seedname.dos.out (run in different file)

projwfc.x < seedname.pdos.in > seedname.pdos.out

Adding together the orbitals obtained from pdos into a single file (this example was with Ti and O atoms for TiO2)

sumpdos.x \\*(Ti\)\*\(s\) > atom\_ti\_s.dat

sumpdos.x \\*(Ti\)\*\(p\) > atom\_ti\_p.dat

sumpdos.x \\*(Ti\)\*\(d\) > atom\_ti\_d.dat

sumpdos.x \\*(Ti\)\* > atom\_ti\_tot.dat

sumpdos.x \\*(O\)\*\(s\) > atom\_ti\_s.dat

sumpdos.x \\*(O\)\*\(p\) > atom\_ti\_p.dat

sumpdos.x \\*(O\)\* > atom\_o\_tot.dat

# Wannier90:

Now we have to prepare the input ﬁle for Wannier90. Open the ﬁle seedname.win, which is a template of the Wannier90 input ﬁle (note that Wannier90 input ﬁle must have the .win extension). Change the values marked with XXX inserting the correct values. In particular:

* + Insert the num\_bands value (this must be equal to the nbnd value set in the nscf calculation).
  + Insert the num\_wann value (this is the number of requested Wannier functions: in this case without disentanglement, this is equal to the num\_bands value).
  + Set the mp\_grid value
  + Insert, between the begin kpoints and end kpoints lines, the list of the 64 kpoints, one per line. Note that while pw.x requires four numbers per line (the three coordinates of each kpoint, and the weight), Wannier90 needs only three numbers (the three coordinates). To obtain these lines, use again the kmesh.pl utility, but this time specifying a fourth parameter to get the list in the Wannier90 format:

/home/moiez/Downloads/FileToRun/wannier90-3.1.0/utility/kmesh.pl 7 7 7 > wan\_kpt.dat

/home/moiez/Downloads/FileToRun/wannier90-3.1.0/utility/kmesh.pl 7 7 7 wan > wan\_kpt.dat

(insert correct mesh number used)

**Note:** Using the kmesh.pl utility, we are sure that we provide enough signiﬁcant digits, and that the list of k-points given to pw.x and to Wannier90 is the same.

* + Inspect the remaining part of the input ﬁle, using the Wannier90 user guide (that can be found on the <https://github.com/wannier-developers/wannier90/raw/v3.1.0/doc/compiled_docs/user_guide.pdf> page) for the input ﬂags that you do not understand.

Change num\_bands to be consistent with the new nscf run.

Change the num\_wann keyword to the correct number of Wannier functions: how many do we want, according to the projections list given above.

Set the maximum energy for the frozen window (ﬂag dis\_froz\_max) inside the energy gap (use the band plot obtained to get a value for this ﬂag).

Set the maximum energy for the disentanglement (ﬂag dis\_win\_max) to an energy large enough so as to contain enough bands for each k point; (check where this value lies in the band plot).

Finally, we are ready to perform a Wannier90 calculation. This is done in three steps:

* We first run a preprocessing step using the command (wannier90.x):

wannier90.x -pp seedname

which produce a seedname.wout file and seedname.nnkp file, that contains the relevant information from the Wannier90 input file in a format to be used in the next step.

* Then we run the pw2wannier90.x code (of the Quantum ESPRESSO distribution).

pw2wannier90.x < seedname.pw2wan.in > seedname.pw2wan.out

The input file for pw2wannier90.x is provided (file seedname.pw2wan.in). We are asking the code to calculate the overlap matrices Mmn (that will be written in the seedname.mmn file) and the Amn matrices (file seedname.amn). Since we want to plot the Wannier functions in real space, we need also the *unk(r)* wavefunctions on a real-space grid. We thus also set the write\_unk flag in seedname.pw2wan.in, that will produce a set of files with names UNK00001.1, UNK00002.1, ... Finally, the code will also produce a seedname.eig file, with the eigenvalues on the initial k-grid (Note: this is not needed to obtain the MLWFs of an insulator, but only for the interpolation and band plotting routines). Note that the pw2wannier90.x expects to find the seedname.nnkp file produced in the previous step. Run the code using pw2wannier90.x < seedname.pw2wan.in > seedname.pw2wan.out

* Finally we can run Wannier90 to obtain MLWFs. Execute

wannier90.x seedname

and, when it ﬁnishes, inspect the output ﬁle, called seedname.wout

* + Check lines containing 🡨 DLTA to check for the convergence of the spread during the iterations.
  + Check the lines after the string Final state: you ﬁnd the centers and spreads of the maximally- localised Wannier functions.
  + To check if the obtained MLWFs are correct, it is typically needed to:

∗ *Compare the Wannier-interpolated band structure with the ab-initio one*: the provided Wannier90 input ﬁle computes the interpolated band plot; you can try to compare the ab-initio bandplot obtained in the steps before with the interpolated band structure (ﬁles seedname .band.dat, and seedname .band.gnu)

* + - To plot it with gnuplot: run gnuplot in terminal, and in gnuplot, type

set xtics nomirr set x2tics

set xrange [\*:\*] noextend set x2range [\*:\*] noextend

plot ’qebands.agr’ w l, ’ seedname.band.dat’ axes x2y1 w l

Note that the Wannier90 code also outputs in the seedname.band.kpt ﬁle a list of the kpoints used for the interpolation, that could be used to plot the band structure on the same grid.

∗ *Plot the real-space Wannier functions and check if they are real*: if you ask Wannier90 to plot the Wannier functions, it will print also the ratio of the imaginary and real part of each of them at the end of the seedname.wout ﬁle: check that the value is small.

* + Plot one of the Wannier functions, which are output in ﬁles seedname \_00001.xsf,
  + To visualize the Wannier functions, you need to install VESTA or xcrysden in your computer, and download the xsf ﬁles:

∗ using xcrysden: open the xsf ﬁle, then choose Tools Data Grid OK, and then choose a reasonable isovalue, activate the Render +/- isovalue ﬂag, and press Submit.

→ →

∗ using VESTA: open the xsf ﬁle, VESTA can automatically ﬁnd a isovalue.

* + Check the output: Before the start of the Wannierization iterations, there is a new section (containing the string 🡨DIS) with the iterations of the disentanglement procedure. It is important that at the end of this section the convergence is achieved (with a string <<< Disentanglement convergence criteria satisfied

>>>).

* + A practical note: Especially when using disentanglement, it is possible that the disentanglement con- vergence is not achieved, and/or that the obtained Wannier functions are not real, and/or that the interpolated band structure diﬀers signiﬁcantly from the ab-initio one within the frozen window. Then, you need to change/tune the number of Wannier functions, the projections you chose and/or the energy values for the frozen and disentanglement windows, until you get “good” Wannier functions.
  + Check ﬁnal WF centers and verify that WFs are real; you may also want to plot the Wannier functions
* Compute the properties of Wannier functions (like DOS). The information on the calculated Wannier functions is read from the checkpoint seedname.chk file. Run the following code:

postw90.x seedname

# TRIQS:

Now that we have run DFT simulation with Quantum Espresso and created the Wannier functions with Wannier90, we are ready to perform DMFT simulation. Go to dmft.ipynb and follow remaining steps.