2024 School on Electron-Phonon Physics, Many-Body Perturbation Theory, and Computational Workflows

Wannier interpolation of band structures

Hands-on session (Tue.4)

Hands-on based on QE-v7.3 and Wannier90-v3.1

Foreword

This document contains 4 exercises to guide you through the basic functionalities of wannier90 (W90) - a computer program that calculates maximally-localized Wannier functions (MLWFs) and related properties.
If you have never run W90 before, please walk through exercises 1+2, the rest is optional; if you have already run successfully some calculations with W90, then we suggest starting from exercise 3.
▶ W90 is at the center of a software ecosystem leveraging the Wannier representation, ranging from interfaces with first-principles simulation software (e.g., Quantum ESPRESSO and EPW) to an increasing number of post-processing packages. Wannier functions can be used to understand chemical bonding, to characterize polarization, magnetization, and topology, or as an optimal basis set, providing very accurate interpolations in reciprocal space or large-scale Hamiltonians in real space. The reader is encouraged to give a look at Ref. [1] for a quick and broad overview of the basic theory and fundamental concepts behind the codes that compose the *Wannier function software ecosystem*; the review article comes with abundant and relevant pointers to more in-depth references.

Code path and tutorial files

- `QE`:

`/work2/05193/sabyadk/stampede3/EPWSchool2024/q-e/bin`

- `Wannier90` (main executable):

`/work2/05193/sabyadk/stampede3/EPWSchool2024/q-e/bin/wannier90.x`

- `Wannier90` (also with utilities):

`/work2/05193/sabyadk/stampede3/EPWSchool2024/autowan/wannier90/`

Please copy the tutorial tarball into your scratch directory using:
`cp /work2/05193/sabyadk/stampede3/EPWSchool2024/tutorials/Tue.4.Marrazzo.tar $SCRATCH`

You can type `tar -xvf Tue.4.Marrazzo.tar` to extract the tutorial files.

Software for visualization

The installations of the following software depend on your operating system. For Linux, first try with the package manager (e.g. `apt` for Debian/Ubuntu), if not available then download the binary distributions from the websites. For Windows or MacOS, first try to download binary distributions, if not available then you might need to download the source file and compile it yourself.

- To plot the band structure:
  - `gnuplot` http://www.gnuplot.info/index.html
  - `xmgrace` https://plasma-gate.weizmann.ac.il/Grace/
- To plot the Wannier functions, Fermi surface:
  - `xcrysden` http://www.xcrysden.org/
  - `VESTA` https://jp-minerals.org/vesta/en/
  - `FermiSurfer` https://mitsuaki1987.github.io/fermisurfer/

Note `gnuplot` is available on Stampede3, if your operating system supports X11 forwarding, then you can connect to Stampede3 by `ssh -Y USER_NAME@login3.stampede3.tacc.utexas.edu` and directly use `gnuplot` on Stampede3. However, depending on your internet connection, the `gnuplot` GUI might be slow. We recommend installing these software packages on your local computer, but if you could not install them, you can still run all calculations.

1 Silicon valence bands

In this exercise you will learn how to obtain maximally-localized Wannier functions (MLWFs) for the valence bands of silicon.

▶ 1st step: Go to the `ex1` folder and inspect the input file `01_scf.in`. The first step is to perform a ground-state calculation for a silicon crystal (FCC) with two atoms per unit cell. Check if you understand all parameters (you can use the web page https://www.quantum-espresso.org/Doc/INPUT_PW.html for keywords that you do not know or ask us). To visualize the crystal structure, there are two options, choose the one you prefer:

- `Quantum ESPRESSO input generator and structure visualizer`: open the following link in a browser,
  https://www.materialscloud.org/work/tools/qeinputgenerator,
  click `Choose File` button, select the `01_scf.in` file, and click "Generate the PWscf input file" button. In the new webpage you find a 3D visualization of the structure.
- `xcrysden` (if you have it installed locally in your computer): either open the program, select from the menu `File`→`Open PWscf...`→`Open PWscf Input File` and then select the input file, or directly from the command line with the command `xcrysden --pwi 01_scf.in`.

```
&control
calculation     =  'scf'
restart_mode    =  'from_scratch'
prefix          =  'si'
pseudo_dir      =  'pseudo/'
outdir          =  'out/'
/
&system
ibrav           =   0
nat             =   2
ntyp            =   1
ecutwfc         =  25.0
ecutrho         = 200.0
/
&electrons
conv_thr        =   1.0d-10
/
ATOMIC_SPECIES
Si  28.  Si.pbe-n-van.UPF
ATOMIC_POSITIONS crystal
Si  -0.25   0.75   -0.25
Si   0.00   0.00    0.00
K_POINTS automatic
10 10 10 0 0 0
CELL_PARAMETERS bohr
-5.1    0.0    5.1
0.0    5.1    5.1
-5.1    5.1    0.0
```

▶ 2nd step: The Si pseudopotential that we will use for the calculation has $Z_{val} = 4$ (this information can be obtained reading the first lines of the pseudopotential file, for instance with the command `less pseudo/Si.pbe-n-van.UPF`). Using the information given above, and knowing that FCC Si is a semiconductor, how many occupied valence bands do you expect (and why)? _____

▶ 3rd step: Run the ground state calculation using the `pw.x` code of the Quantum ESPRESSO suite, which you can find in the following directory:

/work2/05193/sabyadk/stampede3/EPWSchool2024/q-e/bin/. In this directory you will also find all the other executables required for this tutorial unless otherwise specified. The syntax for codes in the Quantum ESPRESSO suite is: *command* < *inputfile* > *outputfile* (i.e. `pw.x < 01_scf.in > scf.out`). You may want to use parallelization to run simulations faster, by using a submission script such as

```
#!/bin/bash
#SBATCH --job-name=epwschool
#SBATCH --time=0-00:20:00
#SBATCH --error=job.err%j
#SBATCH --output=job.out%j
#SBATCH --nodes=1
#SBATCH --ntasks=4
#SBATCH --cpus-per-task=1
#SBATCH -A dmr23030
#SBATCH -p spr
#SBATCH --reservation=NSF_Summer_School_Tue

module load qe/7.3

ibrun -np 4 /work2/05193/sabyadk/stampede3/EPWSchool2024/q-e/bin/pw.x < 01_scf.in \
> scf.out
```

This is for 4 using processors on TACC Stampede3 system, where one has to use `ibrun` instead of `mpirun`. **Note:** In the following every time you see a new command like this,

`ibrun -np X path_to_code < input_file > output_file`,

just replace the last line in the submission script by this new `ibrun ...` command.

▶ 4th step: After the calculation finishes, inspect the output file `scf.out` to check if there are any errors/warnings. Compare your answer to the previous point (number of electrons and occupied valence bands) with the information provided in the output file.

▶ 5th step: Now we want to plot the band structure of silicon (we will use this plot also for the next exercise, where we need also the conduction bands: therefore, we plot also a few of the lowest conduction bands). Copy the file `01_scf.in` to the file `02_bands.in`. Do the following modifications to the file `02_bands.in` (use the INPUT_PW documentation from the link above for an explanation of the meaning of the flags, if needed):

▶ 6th step: In the `CONTROL` namelist, change the `calculation` keyword from `'scf'` to `'bands'` to perform a band structure calculation starting from the ground state density obtained from the scf run.

▶ 7th step: Ask the code to print 12 bands (flag `nbnd=12` in the `SYSTEM` namelist).

▶ 8th step: Set `diago_full_acc = .true.` in the `ELECTRONS` namelist (see documentation for the meaning of this flag).

▶ 9th step: Change the k-point list to plot the band structure along the following path (coordinates are given in crystal units), using 50 points per segment:

- $\mathrm{L}(0.5, 0.5, 0.5) \rightarrow \Gamma(0, 0, 0)$
- $\Gamma(0, 0, 0) \rightarrow \mathrm{X}(0.5, 0, 0.5)$

You can do this simply using the following `K_POINTS` card:

```
K_POINTS crystal_b
3
0.5 0.5 0.5 50
0.  0.  0.  50
```

```
0.5 0.  0.5 50
```

▶ 10th step: Run the calculation using the `pw.x` code, as explained before.
▶ 11th step: When the calculation has finished, run the file `03_bandsx.in` through the `bands.x` executable (make sure to read and understand the input file):

```
ibrun -np 2 /work2/05193/sabyadk/stampede3/EPWSchool2024/q-e/bin/bands.x \
< 03_bandsx.in > bandsx.out
```

```
&bands
prefix  = 'si'
outdir = 'out/'
filband = 'bands.dat'
lsym = .false.
/
```

This will produce the `bands.dat` file.
▶ 12th step: Finally, execute the `plotband.x` code (interactively) and answer to its questions. In particular, the input file is the `bands.dat` file created in the previous step; call the xmgrace file `qebands.agr`. When asked, call the ps file `qebands.ps`. You will be asked to provide the value of the Fermi level, which in this case can be put equal to the highest occupied energy level (see the output file `scf.out`). When asked for the deltaE and reference E for the energy axis, type 2 and Fermi level (use space to separate the 2 numbers), you can also tweak these 2 numbers to adjust the visual output of the figure. At the end, open the xmgrace file (or directly the postscript PS file) and inspect the band structure, identifying the valence and conduction bands.
▶ 13th step: Now we are ready to calculate the wavefunctions on a complete grid of k-points. Copy the `02_bands.in` file that you created before to `05_nscf.in`, and modify the following:

- Change the `calculation` type from `'bands'` to `'nscf'`.
- Change the number of bands from 12 back to the number of valence bands that you expect (see your answer a few lines above: the answer should be 4), since for this exercise we need only the valence bands.
- Change the k-point list to a full $4 \times 4 \times 4$ Monkhorst-Pack mesh, that will be used to calculate the overlap matrices needed to obtain Wannier functions. To obtain the list of k-points, use the `kmesh.pl` utility in the utility folder of the `Wannier90` code, that you can find at: `/work2/05193/sabyadk/stampede3/EPWSchool2024/autowan/wannier90/utility/` using the following command for a $4 \times 4 \times 4$ mesh:
  `/work2/05193/sabyadk/stampede3/EPWSchool2024/autowan/wannier90/utility/kmesh.pl 4 4 4`
  (use the command without parameters to get an explanation of its usage).
- Run the nscf calculation using the `pw.x` code:
  `ibrun -np 2 /work2/05193/sabyadk/stampede3/EPWSchool2024/q-e/bin/pw.x \`
  `< 05_nscf.in > nscf.out`

▶ 14th step: Now we have to prepare the input file for Wannier90. Open the file `ex1.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 to `4 4 4` (since we are using a $4 \times 4 \times 4$ k-mesh).
- 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:
  `/work2/05193/sabyadk/stampede3/EPWSchool2024/autowan/wannier90/utility/kmesh.pl 4 4 4 wan`
  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. Try to understand, in particular, the `projections` section. Can you say where the four s-like orbitals are located with respect to the Si atoms? _____

```
use_ws_distance = .true.

num_bands        =   XXX
num_wann         =   XXX
num_iter         =  100

iprint           =    2
num_dump_cycles  =   10
num_print_cycles =   10

!! To plot the WFs
! restart                =  plot
wannier_plot           =  true
wannier_plot_supercell =  3
! wannier_plot_list      =  1,5

!! To plot the WF interpolated bandstructure
bands_plot      =  true
begin kpoint_path
L 0.50000  0.50000 0.5000 G 0.00000  0.00000 0.0000
G 0.00000  0.00000 0.0000 X 0.50000  0.00000 0.5000
end kpoint_path

!! !! Bond-centred s-orbitals
begin projections
f=-0.125,-0.125, 0.375:s
f= 0.375,-0.125,-0.125:s
f=-0.125, 0.375,-0.125:s
f=-0.125,-0.125,-0.125:s
end projections

begin atoms_frac
Si -0.25  0.75  -0.25
```

```
Si  0.00  0.00   0.00
end atoms_frac

begin unit_cell_cart
bohr
-5.10   0.00   5.10
0.00   5.10   5.10
-5.10   5.10   0.00
end unit_cell_cart

mp_grid  =  XXX XXX XXX
begin kpoints
XXX
end kpoints
```

▶ 15th step: Finally, we are ready to perform a Wannier90 calculation. This is done in three steps:

1. We first run a preprocessing step using the command (`wannier90.x` is located here: `/work2/05193/sabyadk/stampede3/EPWSchool2024/q-e/bin/`, in the following tutorial the absolute path is omitted for simplicity, but do remember to use the appropriate absolute path when running the executables, or add the path to the environment variable `$PATH`)
   `wannier90.x -pp ex1`
   which produce a `ex1.wout` file and `ex1.nnkp` file, that contains the relevant information from the Wannier90 input file in a format to be used in the next step.

2. Then we run the `pw2wannier90.x` code (of the Quantum ESPRESSO distribution).
   The input file for `pw2wannier90.x` is provided (file `06_pw2wan.in`). We are asking the code to calculate the overlap matrices M_{mn} (that will be written in the `ex1.mmn` file) and the A_{mn} matrices (file `ex1.amn`). Since we want to plot the Wannier functions in real space, we need also the $u_{nk}(r)$ wavefunctions on a real-space grid. We thus also set the `write_unk` flag in `06_pw2wan.in`, that will produce a set of files with names `UNK00001.1`, `UNK00002.1`, ... Finally, the code will also produce a `ex1.eig` file, with the eigenvalues on the initial $4 \times 4 \times 4$ 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 `ex1.nnkp` file produced in the previous step. Run the code using
   `ibrun -np 2 pw2wannier90.x < 06_pw2wan.in > pw2wan.out`

```
&inputpp
outdir     =  'out/'
prefix     =  'si'
seedname   =  'ex1'
write_amn  =  .true.
write_mmn  =  .true.
write_unk  =  .true.
/
```

3. Finally we can run Wannier90 to obtain MLWFs. Execute
   `wannier90.x ex1`
   and, when it finishes, inspect the output file, called `ex1.wout`.

   ▶ 16th step: Check lines containing `<-- DLTA` to check for the convergence of the spread during the iterations.

▶ 17th step: Check the lines after the string `Final state`: you find the centers and spreads of the maximally-localized Wannier functions.

▶ 18th step: To check if the obtained MLWFs are correct, one typically do the following checks:

- *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 `ex1_band.dat`, and `ex1_band.gnu`)
  - To plot it with `gnuplot`: run `gnuplot` in terminal, and in `gnuplot`, type

    ```
    set xtics nomirr
    set x2tics
    set x2range [0:0.21721815E+01]
    set xrange [0:1.3195]
    plot 'bands.dat.gnu' w p pt 7, 'ex1_band.dat' axes x2y1 w l
    ```

    Note you can reuse the script in following exercises when comparing band structures, by replacing the file names `qebands.agr` and `ex1_band.dat`.
  - Or plot it with `xmgrace` (Note you need to have `xmgrace` installed on your computer, and download the `agr,dat` files from Stampede3): in terminal, type:

    `xmgrace qebands.agr ex1_band.dat`

    Note that you may need to rescale the x axis.

  Note that the `Wannier90` code also outputs in the `ex1_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 `ex1.wout` file: check that the value is small.

▶ 19th step: Plot one of the Wannier functions, which are stored in the files `ex1_00001.xsf`, ... To visualize 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.

2 Silicon valence and conduction bands

While in exercise 1 we obtained a Wannier-interpolated band structure for the occupied bands only, now we will see how to obtain an interpolated band structure for the conductions bands.
▶ 1st step: Copy all the `ex1` folder to a new folder named `ex2`. The first step, i.e. the SCF calculation (`01_scf.in`), is identical. Hence, if you copied also the out directory, you don't need to rerun it.
▶ 2nd step: In the `05_nscf.in` file, change the value of `nbnd` to 12 to calculate the eigenenergies and wavefunctions for 12 bands, and run the nscf calculation (using `pw.x` as in the previous exercise).
▶ 3rd step: Rename `ex1.win` to `ex2.win`, and modify the following flags:

- Change `num_bands` to 12 to be consistent with the new nscf run.
- Change the projections to 4 sp^3 orbitals for each Si atom in the unit cell: to do this, the projections section should read:

```
begin projections
Si:sp3
end projections
```

- 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 in exercise 1 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; 17.0 eV should be a reasonable value (check where this value lies in the band plot).

▶ 4th step: Inside the file `06_pw2wan.in`, change the seedname to '`ex2`' to reflect the new name of the .win file.
▶ 5th step: Run `wannier90.x -pp ex2`.
▶ 6th step: Run `pw2wannier90.x` using `06_pw2wan.in` as input file.
▶ 7th step: Run `wannier90.x ex2`.
▶ 8th step: 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 convergence 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.

▶ 9th step: Check final WF centers and verify that WFs are real; you may also want to plot the Wannier functions, or compare the interpolated band structure with the ab-initio one (obtained in exercise 1).

▶ 10th step: **Optional (Do it only if you have enough time)**: Do the symmetry and the centers of the Wannier functions agree with your intuition? (We would like 4 sp^3-like orbitals centered on each Si atom, with similar spreads). Try to rerun everything with a $6 \times 6 \times 6$ kgrid for the `nscf` and `Wannier90` step to check if the results improve, and how the spreads change with respect to the grid density?

3 Lead Fermi surface and band structure

In this exercise we will see how to interpolate the band structure of lead, in particular around the Fermi energy. The first goal is to obtain the Fermi surface from Wannier interpolation. This will clearly show some of the advantages of using Wannier-interpolation schemes with respect to direct calculations on very dense k-point grids. In order to build a MLWFs model that describes the band structure around the Fermi energy, we need to have an idea of the orbital character of the bands we are interested in. The crystal structure of pure lead has one atom per primitive cell, and since we also want to describe some states above the Fermi Energy, we include five d orbitals ($d_{xy}, d_{xz}, d_{yx}, d_{z^2}$ and $d_{x^2-y^2}$), and four sp^3 orbitals. Hence our guess for the projections is

- 5 d orbitals centered on the lead atom
- 4 sp^3 orbitals centred on the lead atom

Now we are ready to obtain MLWFs and describe the states of lead around the Fermi-level.

- Directory: ex3
- Input Files
  - `01_scf.in` *The* PWSCF *input file for the ground state calculation*

```
&control
calculation='scf'
restart_mode='from_scratch',
pseudo_dir = './pseudo/',
outdir='./'
prefix='pb'
/
&system
ibrav = 2, celldm(1) = 9.3555, nat= 1, ntyp= 1,
ecutwfc = 47.0, ecutrho = 189,
occupations='smearing', smearing='cold', degauss=0.02
/
&electrons
conv_thr = 1.0e-9
mixing_beta = 0.7
/
ATOMIC_SPECIES
Pb 207.2 Pb.pbe-dn-kjpaw_psl.0.2.2.UPF
ATOMIC_POSITIONS
Pb 0.0 0.0 0.0
K_POINTS (automatic)
8 8 8  0 0 0
```

  - `04_nscf.in` *The* PWSCF *input file to obtain Bloch states on a uniform grid*

```
&control
calculation='nscf'
pseudo_dir = './pseudo/',
outdir='./'
prefix='pb'
/
&system
```

```
ibrav = 2, celldm(1) = 9.3555, nat= 1, ntyp= 1,
ecutwfc = 47.0, ecutrho = 189,
occupations='smearing', smearing='cold', degauss=0.02
nosym=.true.,nbnd=13
/
&electrons
conv_thr = 1.0e-9
/
ATOMIC_SPECIES
Pb 207.2 Pb.pbe-dn-kjpaw_psl.0.2.2.UPF
ATOMIC_POSITIONS
Pb 0.0 0.0 0.0
K_POINTS crystal
512
0.00000000  0.00000000  0.00000000  1.953125e-03
0.00000000  0.00000000  0.12500000  1.953125e-03
...
```

– `05_pw2wan.in` *Input file for* `pw2wannier90`

```
&inputpp
outdir = './'
prefix = 'pb'
seedname = 'ex3'
write_amn = .true.
write_mmn = .true.
/
```

– `ex3.win` *The* `wannier90` *input file*

```
use_ws_distance = .true.

num_bands       = 13
num_wann        =  9
num_iter        = 200

dis_win_max     = 38.0
dis_froz_max    = 16.0
dis_num_iter    =  50
dis_mix_ratio   = 1.0

Begin Kpoint_Path
G 0.00  0.00  0.00    X 0.50  0.50  0.00
X 0.50  0.50  0.00    W 0.50  0.75  0.25
W 0.50  0.75  0.25    L 0.00  0.50  0.00
L 0.00  0.50  0.00    G 0.00  0.00  0.00
G 0.00  0.00  0.00    K 0.00  0.50 -0.50
End Kpoint_Path

! SYSTEM

begin unit_cell_cart
bohr
-4.67775 0.00000 4.67775
```

```
0.00000 4.67775 4.67775
-4.67775 4.67775 0.00000
end unit_cell_cart

begin atoms_frac
Pb 0.00   0.00   0.00
end atoms_frac

begin projections
Pb:d;sp3
end projections

! KPOINTS

mp_grid : 8 8 8

begin kpoints
0.00000000  0.00000000  0.00000000
0.00000000  0.00000000  0.12500000
...
```

▶ 1st step: Run PWSCF to obtain the ground state of lead
`pw.x < 01_scf.in > scf.out`
▶ 2nd step: Run PWSCF to obtain the Bloch states on a uniform k-point grid
`pw.x < 04_nscf.in > nscf.out`
▶ 3rd step: Run `wannier90` to generate a list of the required overlaps (written into the `lead.nnkp` file).
`wannier90.x -pp ex3`
▶ 4th step: Run `pw2wannier90` to compute the overlap between Bloch states and the projections for the starting guess (written in the `lead.mmn` and `lead.amn` files).
`pw2wannier90.x < 05_pw2wan.in > pw2wan.out`
▶ 5th step: Run `wannier90` to compute the MLWFs.
`wannier90.x ex3`
▶ 6th step: Inspect the output file `ex3.wout`.
▶ 7th step: Use Wannier interpolation to obtain the Fermi surface of lead. Rather than re-running the whole calculation we can use the unitary transformations obtained in the first calculation and restart from the plotting routine. Add the following keywords to the `lead.win` file:

```
restart = plot
fermi_energy = [insert your value here]
fermi_surface_plot = true
```

and re-run `wannier90`. The value of the Fermi energy can be obtained from the output of the initial first principles calculation. `wannier90` calculates the band energies, through wannier interpolation, on a dense mesh of k-points in the Brillouin zone. The density of this grid is controlled by the keyword `fermi_surface_num_points`. The default value is 50 (i.e., 50^3 points). The Fermi surface file `lead.bxsf` can be viewed using `XCrySDen`, e.g.,

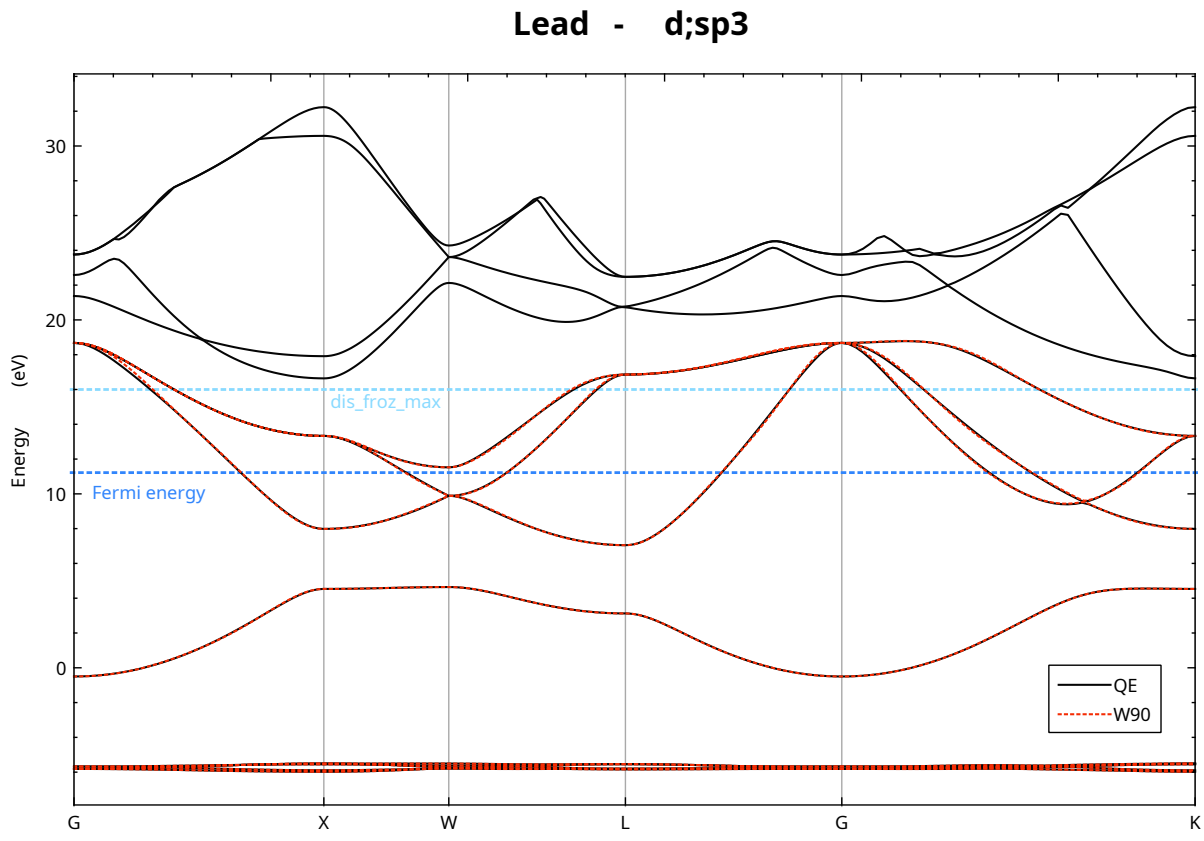
```
xcrysden --bxsf ex3.bxsf
```

▶ 8th step: Plot the interpolated band structure. A suitable path in k-space is

```
begin kpoint_path
G 0.00 0.00 0.00 X 0.50 0.50 0.00
X 0.50 0.50 0.00 W 0.50 0.75 0.25
W 0.50 0.75 0.25 L 0.00 0.50 0.00
L 0.00 0.50 0.00 G 0.00 0.00 0.00
G 0.00 0.00 0.00 K 0.00 0.50 -0.50
end kpoint_path
```

Further ideas (if you have time)

- Compare the Wannier interpolated band structure with the full PWSCF band structure. Obtain MLWFs using a denser k-point grid. To plot the band structure you can use the PWSCF tool `bands.x`.
- Investigate the effects of the outer and inner energy windows on the interpolated bands.
- Instead of extracting a subspace of $d + sp^3$ states, we could extract a different nine dimensional space (i.e., with s, p and d character). Examine this case and compare the interpolated band structures.
- Remove the low-energy d states from the wannierization (hint: use the `exclude_bands` option in the Wannier90 input file) and compare both the spread and band structure you obtain.



Band Structure of lead showing the position of the Fermi energy and inner energy windows.

OPTIONAL

Do the following exercises only if you have enough time, i.e. you completed all the previous exercises.

4 Lead SCDM parameters from projectability

- Outline: *Compute the Wannier interpolated band structure of lead using the selected columns of the density matrix (SCDM) method[2] to calculate the initial guess. The free parameters in the SCDM method, i.e., μ and σ, are obtained by fitting a complementary error function to the projectabilities. We choose the number of MLWFs to be the number of pseudo-atomic orbitals (PAOs) in the pseudopotential, 9 in this case. All the steps shown in this example have been automated in the AiiDA[3, 4] workflow that can be downloaded from the MaterialsCloud website[5]. If you are interested, more details can be found in a dedicated lecture of the 2022 Wannier Summer School[6] and in a specific tutorial page[7] which contains the most recent codes and applications.*

- Directory: `ex4/`

- Input files

  - `01_scf.in` *The* PWSCF *input file for ground state calculation*

```
&control
calculation='scf'
restart_mode='from_scratch',
pseudo_dir = './pseudo/',
outdir='./'
prefix='pb'
/
&system
ibrav = 2, celldm(1) = 9.3555, nat= 1, ntyp= 1,
ecutwfc = 47.0, ecutrho = 189,
occupations='smearing', smearing='cold', degauss=0.02
/
&electrons
conv_thr = 1.0e-9
mixing_beta = 0.7
/
ATOMIC_SPECIES
Pb 207.2 Pb.pbe-dn-kjpaw_psl.0.2.2.UPF
ATOMIC_POSITIONS
Pb 0.0 0.0 0.0
K_POINTS (automatic)
8 8 8  0 0 0
```

  - `02_bands.in` *The input file for the band structure calculation along a kpath*

```
&control
calculation     =  'bands'
prefix          =  'pb'
pseudo_dir      =  'pseudo/'
outdir          =  './'
```

```
/
&system
ibrav = 2, celldm(1) = 9.3555, nat= 1, ntyp= 1,
ecutwfc = 47.0, ecutrho = 189,
occupations='smearing', smearing='cold', degauss=0.02
nosym=.true.,nbnd=13
/
&electrons
conv_thr        =   1.0d-9
/
ATOMIC_SPECIES
Pb 207.2 Pb.pbe-dn-kjpaw_psl.0.2.2.UPF
ATOMIC_POSITIONS crystal
Pb 0.0 0.0 0.0
K_POINTS crystal_b
6
0.00  0.00  0.00    50
0.50  0.50  0.00    50
0.50  0.75  0.25    50
0.00  0.50  0.00    50
0.00  0.00  0.00    50
0.00  0.50 -0.50    50
```

– `03_bandsx.in` *The input file for* `bands.x`

```
&bands
prefix  = 'pb'
outdir = './'
filband = 'bands.dat'
/
```

– `04_nscf.in` *The* PWSCF *input file to obtain Bloch states on a uniform grid*

```
&control
calculation='nscf'
pseudo_dir = './pseudo/',
outdir='./'
prefix='pb'
/
&system
ibrav = 2, celldm(1) = 9.3555, nat= 1, ntyp= 1,
ecutwfc = 47.0, ecutrho = 189,
occupations='smearing', smearing='cold', degauss=0.02
nosym=.true.,nbnd=13
/
&electrons
conv_thr = 1.0e-9
/
ATOMIC_SPECIES
Pb 207.2 Pb.pbe-dn-kjpaw_psl.0.2.2.UPF
ATOMIC_POSITIONS
Pb 0.0 0.0 0.0
K_POINTS crystal
512
```

```
0.00000000  0.00000000  0.00000000  1.953125e-03
0.00000000  0.00000000  0.12500000  1.953125e-03
...
```

– `05_proj.in` *The input file for* `projwfc`

```
&projwfc
prefix = 'pb',
outdir = './',
lsym = .FALSE.,
filproj = 'lead-proj'
/
```

– `06_pw2wan.in` *The input file for* `pw2wannier90`

```
&inputpp
outdir = './'
prefix = 'pb'
seedname = 'ex4'
write_amn = .true.
write_mmn = .true.
/
```

– `generate_weights.sh` *The bash script to extract the projectabilities from the output of* `projwfc`

```
#!/bin/bash
# This script is used to extract the projectability from
# the output of projwfc.x (proj.out).

# Remove temporary files if exist
rm -f e.dat
rm -f p.dat
rm -f tmp.dat
rm -f p_vs_e.dat

# Check proj.out exists
[[ -f "proj.out" ]] || { echo "proj.out not found!"; echo "Aborting!"; exit 1; }

# Get energies and projectability in the correct order
cat proj.out |grep '=='|awk '{print $5}' > e.dat
cat proj.out |grep '|psi|^2'|awk '{print $3}' > p.dat
paste e.dat p.dat > tmp.dat

sort -k1n tmp.dat > p_vs_e.dat

# Clean workspace
rm e.dat p.dat tmp.dat
```

– `ex4.win` *The* `wannier90` *input file*

```
use_ws_distance = .true.
dis_num_iter = 0
```

```
auto_projections = .true.

num_bands       = 13
num_wann        =  9
num_iter        = 200

Begin Kpoint_Path
G 0.00  0.00  0.00    X 0.50  0.50  0.00
X 0.50  0.50  0.00    W 0.50  0.75  0.25
W 0.50  0.75  0.25    L 0.00  0.50  0.00
L 0.00  0.50  0.00    G 0.00  0.00  0.00
G 0.00  0.00  0.00    K 0.00  0.50 -0.50
End Kpoint_Path

bands_plot = .true.

! SYSTEM

begin unit_cell_cart
bohr
-4.67775 0.00000 4.67775
0.00000 4.67775 4.67775
-4.67775 4.67775 0.00000
end unit_cell_cart

begin atoms_frac
Pb 0.00   0.00   0.00
end atoms_frac

! KPOINTS

mp_grid : 8 8 8

begin kpoints
0.00000000  0.00000000  0.00000000
0.00000000  0.00000000  0.12500000
...
```

1. Run PWSCF to obtain the ground state of lead
   `pw.x -in 01_scf.in > scf.out`
2. Run `pw.x` to obtain QE band structure along a kpath
   `pw.x -in 02_bands.in > bands.out`
3. Run `bands.x` to generate a dat file for plotting
   `bands.x -in 03_bandsx.in > bandsx.out`
4. Run PWSCF to obtain the Bloch states on a $8 \times 8 \times 8$ uniform k-point grid
   `pw.x -in 04_nscf.in > nscf.out`
5. Run `wannier90` to generate a list of the required overlaps (written into the `lead.nnkp` file)
   `wannier90.x -pp ex4`
   Note there is no `projections` block and we set `auto_projections = .true.`.
6. Automatically find μ_{SCDM} and σ_{fit} by using the protocol defined in Ref. [8]. If you have limited time, you can skip these and continue with the next step.

(a) Run `projwfc` to compute the projectabilities of the Bloch states onto the Bloch sums obtained from the PAOs in the pseudopotential
`projwfc.x -in 05_proj.in > proj.out`

(b) Run `generate_weights` to extract the projectabilitites from `proj.out` in a format suitable to be read by Xmgrace or gnuplot
`./generate_weights.sh`

(c) Plot the projectabilities and fit the data with the complementary error function

$$f(\epsilon;\mu,\sigma) = \frac{1}{2}\mathrm{erfc}(-\frac{\mu-\epsilon}{\sigma}).$$

We are going to use Xmgrace to plot the projectabilities and perform the fitting. Open Xmgrace by typing `xmgrace` in the terminal.

To Import the `p_vs_e.dat` file, click on `Data` from the top bar and then `Import -> ASCII...`. At this point a new window `Grace: Read sets` should pop up. Select `p_vs_e.dat` in the `Files` section, click `Ok` at the bottom and close the window. You should now be able to see a quite noisy function that is bounded between 1 and 0. You can modify the appearence of the plot by clicking on `Plot` in the top bar and then `Set appearance...`. In the `Main` section of the pop-up window change the symbol type from `None` to `Circle`. Change the line type from straight to none, since the lines added by default by Xmgrace are not meaningful. For the fitting, go to `Data -> Transformations -> Non-linear curve fitting`. In this window, select the source from the `Set` box and in the `Formula` box insert the following
`y = 0.5 * erfc( ( x - A0 ) / A1 )`
Select 2 as number of parameters, give 17.7 as initial condition for `A0` and 1.6 for `A1`. Click `Apply`. A new window should pop up with the stats of the fitting. In particular you should find a `Correlation coefficient` of 0.9778 and a value of 17.7617 for `A0` and 1.62839 for `A1`. These are the value of μ_{fit} and σ_{fit} we are going to use for the SCDM method. In particular, $\mu_{SCDM} = \mu_{fit} - 3\sigma_{fit} = 12.87653$ eV and $\sigma_{SCDM} = \sigma_{fit} = 1.62839$ eV. The motivation for this specific choice of μ_{fit} and σ_{fit} may be found in Ref. [8], where the authors also show validation of this approach on a dataset of 200 materials. You should now see the fitting function, as well as the projectabilities, in the graph (see Fig. 2).

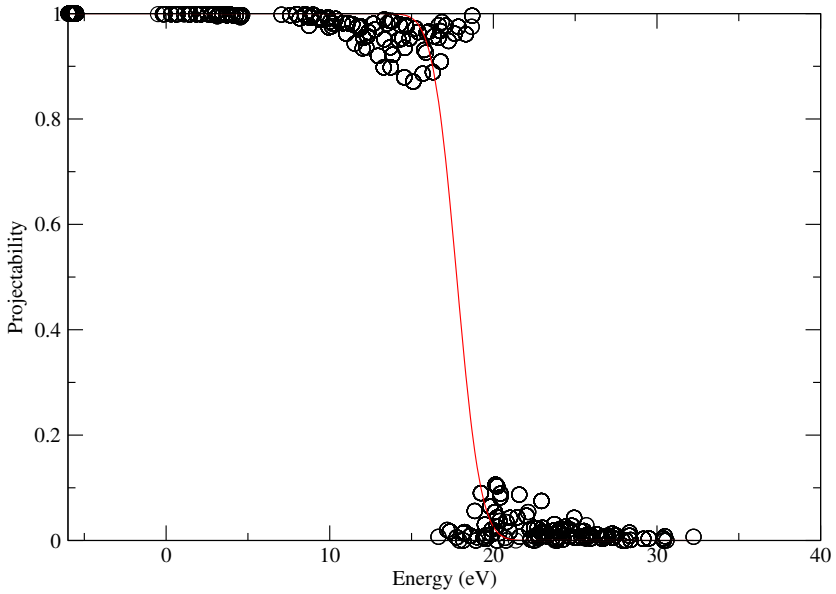
7. Open `06_pw2wan.in` and append the following lines

```
scdm_entanglement = 'erfc'
scdm_proj = .true.
scdm_mu = 12.87653
scdm_sigma = 1.62839
```

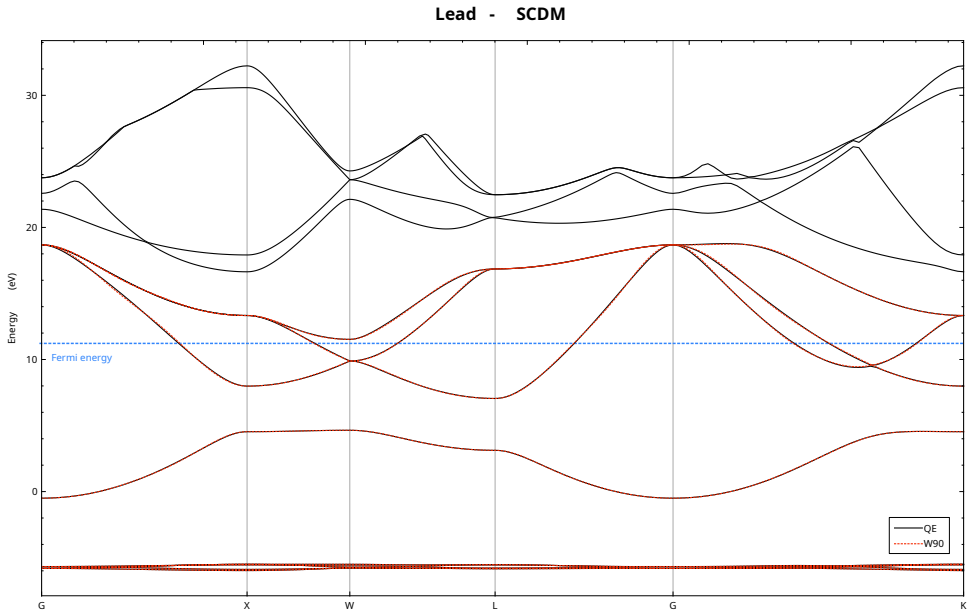
8. Run `pw2wannier90` to compute the overlaps between Bloch states and the projections for the starting guess (written in the `lead.mmn` and `lead.amn` files)
`pw2wannier90.x -in 06_pw2wan.in > pw2wan.out`

9. Run `wannier90` to obtain the interpolated band structure (see Fig. 3).
`wannier90.x ex4`

Please cite Ref. [8] in any publication employing the procedure outlined in this example to obtain μ and σ.



Each black circle represents the projectability as defined in Eq. (22) of Ref. [8] of the state $|n\mathbf{k}\rangle$ as a function of the corresponding energy $\epsilon_{n\mathbf{k}}$ for lead. The red line is the fitted erfc function.



Band structure of lead from DFT calculations (solid black) and Wannier interpolation using the SCDM method to construct the initial guess (red dashed line).

[1] Antimo Marrazzo, Sophie Beck, Elena R. Margine, Nicola Marzari, Arash A. Mostofi, Junfeng Qiao, Ivo Souza, Stepan S. Tsirkin, Jonathan R. Yates, and Giovanni Pizzi. The Wannier-Functions Software Ecosystem for Materials Simulations, 2023. URL https://doi.org/10.48550/arXiv.2312.10769.

[2] Anil Damle and Lin Lin. Disentanglement via entanglement: A unified method for wannier localization. *Multiscale Modeling & Simulation*, 16(3):1392–1410, jan 2018. doi: 10.1137/17m1129696.

[3] Giovanni Pizzi, Andrea Cepellotti, Riccardo Sabatini, Nicola Marzari, and Boris Kozinsky. Aiida: automated interactive infrastructure and database for computational science. *Computational Materials Science*, 111:218 – 230, 2016. doi: 10.1016/j.commatsci.2015.09.013.

[4] Sebastiaan P. Huber, Spyros Zoupanos, Martin Uhrin, Leopold Talirz, Leonid Kahle, Rico Häuselmann, Dominik Gresch, Tiziano Müller, Aliaksandr V. Yakutovich, Casper W. Andersen, Francisco F. Ramirez, Carl S. Adorf, Fernando Gargiulo, Snehal Kumbhar, Elsa Passaro, Conrad Johnston, Andrius Merkys, Andrea Cepellotti, Nicolas Mounet, Nicola Marzari, Boris Kozinsky, and Giovanni Pizzi. AiiDA 1.0, a scalable computational infrastructure for automated reproducible workflows and data provenance. 7(1):300, 2020. ISSN 2052-4463. doi: 10.1038/s41597-020-00638-4. URL https://www.nature.com/articles/s41597-020-00638-4.

[5] V. Vitale, G. Pizzi, A. Marrazzo, J. R. Yates, N. Marzari, and A. A. Mostofi. Automated high-throughput wannierisation. *Materials Cloud Archive*, 2019. doi:10.24435/materialscloud:2019.0044/v2.

[6] Wannier90 developers. Wannier 20220 Summer School. https://indico.ictp.it/event/9789/other-view?view=ictptimetable, 2022.

[7] aiida-wannier90-workflows developers. Tutorials: Automated high-throughput Wannierisation. https://aiida-wannier90-workflows.readthedocs.io/en/v2.0.0b1/user_guide/get_started/scdm/scdm.html, 2022.

[8] Valerio Vitale, Giovanni Pizzi, Antimo Marrazzo, Jonathan R. Yates, Nicola Marzari, and Arash A. Mostofi. Automated high-throughput wannierisation. *npj Computational Materials*, 6(1):66, jun 2020. doi: 10.1038/s41524-020-0312-y.