Advanced Quantum ESPRESSO tutorial:
Hubbard and Koopmans functionals from linear response
Virtual tutorial from 9 to 11 November 2022

Hands-On Tutorial
Wannierization and Wannier interpolation of band structure and Fermi surface
J. Qiao, A. Marrazzo, G. Pizzi

Foreword

This document contains 2 exercises to help you learn wannier90 (W90) - the computer program that calculates maximally-localised Wannier functions (MLWFs).

Required executables and tutorial files

- QE: `pw.x`, `bands.x`, `plotband.x`, `pw2wannier90.x`
- Wannier90: `wannier90.x`, `kmesh.pl`

The tutorial files are hosted on GitHub repo:
https://github.com/materialscloud-org/hubbard-koopmans-2022/tree/main/Day1

Software for visualization

- 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/

These are already pre-installed in Quantum Mobile.

1 Silicon valence and conduction bands

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

- Go to the `exercise5/` 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, you can choose either one as 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`: either opening the program and selecting from the menu `File`→`Open PWscf...`→`Open PWscf Input File` and selecting 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      =  '../../files/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
```

- 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)? _____

- Run the ground state calculation using the `pw.x` code of the Quantum ESPRESSO suite. 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 the parallelization to run the simulation faster, e.g. for 2 processors, use `ibrun` instead of `mpirun`.

```
mpirun -np 2 pw.x < 01_scf.in > scf.out
```

- 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.

- Now we want to plot the band structure of silicon. Copy the file `01_scf.in` to a new 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):

  - 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.
  - Ask the code to print 12 bands (flag `nbnd=12` in the `SYSTEM` namelist).
  - Set `diago_full_acc = .true.` in the `ELECTRONS` namelist (see documentation for the meaning of this flag).
  - 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
```

  - Run the calculation using the `pw.x` code, as explained before.
  - 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):

```
mpirun -np 2 bands.x < 03_bandsx.in > bandsx.out
```

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

This will produce the `bands.dat` file.

– 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.

- 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'`.

  – Keep the number of bands to 12.

  – 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 which is bundled in the `Wannier90` code, that you can download using the following command:

```
# download
wget https://raw.githubusercontent.com/wannier-developers/wannier90/develop/utility/kmesh.pl
# make the script executable
chmod +x kmesh.pl
```

Then generate the kpoint coordinates using the following command for a $4 \times 4 \times 4$ mesh:

```
./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:

```
mpirun -np 2 pw.x < 05_nscf.in > nscf.out
```

- Now we have to prepare the input file for Wannier90. Open the file `si2.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 if the `exclude_bands` parameter is not specified).
- Insert the `num_wann` value (this is the number of requested Wannier functions: in this case for valence and conduction bands, this is equal to 8, why? You may find some hints from the `projections` keyword as will be shown in later paragraph).
- 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:

```
./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.
- Set the maximum energy for the frozen window (flag `dis_froz_max`) inside the energy gap (use the band plot obtained in previous step 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).
- 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 (project to 4 sp^3 orbitals for each Si atom in the unit cell).

```
use_ws_distance = .true.

num_bands        =   XXX
num_wann         =   XXX
num_iter         =  500

dis_win_max      =   XXX
dis_froz_max     =   XXX
dis_num_iter     =  500

!! To plot the WFs
! restart               =  plot
wannier_plot           =  true
wannier_plot_supercell =  3

!! To plot the WF interpolated band structure
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

begin projections
Si: sp3
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
```

- 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 -pp si2
     ```

     which produce a `si2.wout` file and `si2.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 `si2.mmn` file) and the A_{mn} matrices (file `si2.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 `si2.eig` file, with the eigenvalues on the initial $4 \times 4 \times 4$ k-grid. Note that the `pw2wannier90.x` expects to find the `si2.nnkp` file produced in the previous step. Run the code using

     ```
     mpirun -np 2 pw2wannier90.x < 06_pw2wan.in > pw2wan.out
     ```

     ```
     &inputpp
       outdir     =  'out/'
       prefix     =  'si'
       seedname   =  'si2'
       write_amn  =  .true.
     ```

```
  write_mmn  =  .true.
  write_unk  =  .true.
/
```

3. Finally we can run Wannier90 to obtain MLWFs. Execute

```
wannier90.x si2
```

and, when it finishes, inspect the output file, called `si2.wout`.

- Before the start of the maximal localisation iterations, there is a 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 >>>`).
- Check lines containing `<- DLTA` to check for the convergence of the spread during the maximal localisation 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 `si2_band.dat`, and `si2_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, 'si2_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 `si2_band.dat`.
    · Or plot it with `xmgrace` (Note you need to have `xmgrace` installed on your computer): in terminal, type:

    ```
    xmgrace qebands.agr si2_band.dat
    ```

    Note that you may need to rescale the x axis.

    The `Wannier90` code also outputs in the `si2_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 `si2.wout` file: check that the value is small.
  * 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.

- Plot one of the Wannier functions, which are output in files `si2_00001.xsf, ....` To visualize the Wannier functions, you need to install `xcrysden` or `VESTA` 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.

- **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? Also try to tune the `dis_froz_max` and see the difference of MLWF spreads and band interpolation?

2 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 full k-points direct calculations. 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: `exercise6/`
- Input Files
  - `01_scf.in` *The* PWSCF *input file for the ground state calculation*

```
&control
    calculation='scf'
    restart_mode='from_scratch',
    pseudo_dir = '../../files/pseudo/',
    outdir='./out'
    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 (crystal)
  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 = '../../files/pseudo/',
```

```
outdir='./out'
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 = './out'
prefix = 'pb'
seedname = 'lead'
write_amn = .true.
write_mmn = .true.
/
```

– `lead.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
...
```

1. Run PWSCF to obtain the ground state of lead
   `pw.x < 01_scf.in > scf.out`
2. Run PWSCF to obtain the Bloch states on a uniform k-point grid
   `pw.x < 04_nscf.in > nscf.out`
3. Run `wannier90` to generate a list of the required overlaps (written into the `lead.nnkp` file).
   `wannier90.x -pp lead`
4. 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`
5. Run `wannier90` to compute the MLWFs.
   `wannier90.x lead`

Inspect the output file `lead.wout`.

1. 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.,

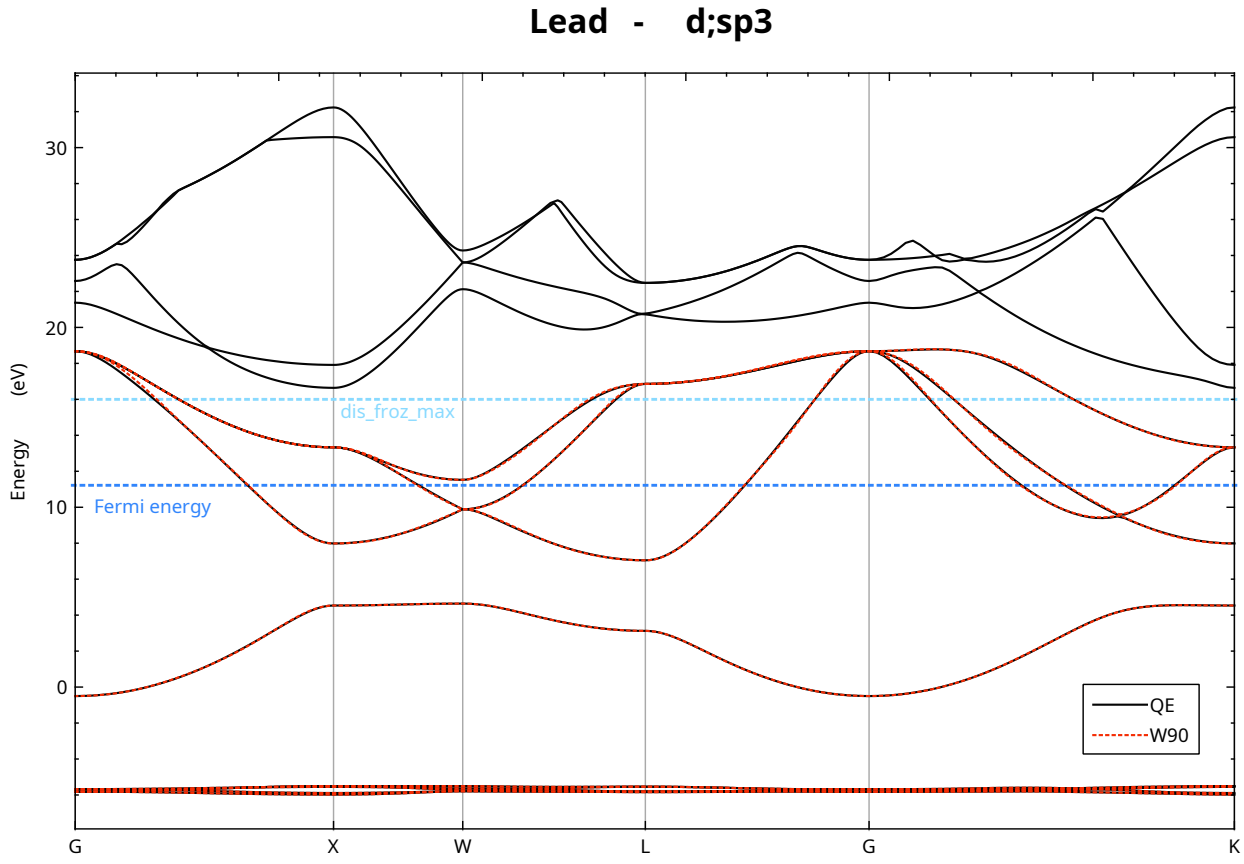
```
xcrysden --bxsf lead.bxsf
```

2. 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.