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

10-16 June 2024, Austin TX

Hands-On Tutorial (Saturday morning)

# Automated Wannierizations

Junfeng Qiao

## Contents



## Foreword

This document contains 3 exercises to help familiarize yourself with two automated Wannierization method: projectability-disentangled Wannier function (PDWF) for entangled bands, and manifold-remixed Wannier function (MRWF) for isolated bands. Specifically, in Exercise 1 we will obtain PDWFs for graphene, to show how projectability disentanglement can be used to extract localized orbitals in conduction bands. In Exercise 2, apart from the standard $s$ and $p$ orbitals for silicon, we will use additional $d$ orbitals to Wannierize higher-energy conduction bands. In Exercise 3, we will use MRWFs to separate the valence and conduction bands of silicon, to show how manifold remixing can be used to automatically construct bonding/anti-bonding orbitals of the valence/conduction bands of insulators.

### Tutorial files

Note that the raw input files are attached in the PDF for completeness, however, you do not need to copy-paste them manually. Instead, all the files are available in the tarball `Sat.6.Qiao.tgz`. You can copy the tarball to your scratch directory and extract the files as follows:

```
$ cd $SCRATCH
$ mkdir EPWSchool2024 ; cd EPWSchool2024
$ cp /work2/05193/sabyadk/stampede3/EPWSchool2024/tutorials/Sat.6.Qiao.tgz .
$ tar -xvf Sat.6.Qiao.tgz; cd Sat.6.Qiao/ex1
```

### Codes

We will use custom versions of Quantum ESPRESSO (QE) and Wannier90 (W90), and several other utility codes for the tutorial. They can be loaded with the following module commands:

```
$ module use /work2/05193/sabyadk/stampede3/EPWSchool2024/autowan/modulefiles
$ module load autowan
```

### Parallel execution

Moreover, probably you want to run the calculations on a compute node with parallelization, so you need to submit a job to the queue system with the following template script (also available in the tarball as `submit.sh`):

```
#!/bin/bash
#SBATCH --job-name=ex1
#SBATCH --time=01:00:00
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=8
#SBATCH -A DMR23030
#SBATCH --reservation=DMR23030_June_15
#SBATCH --partition=small
#SBATCH --hint=nomultithread
#SBATCH --output=slurm.out
#SBATCH --error=slurm.err

module use /work2/05193/sabyadk/stampede3/EPWSchool2024/autowan/modulefiles
module load autowan

# for TACC machine, use ibrun instead of mpirun, e.g.
ibrun pw.x -inp graphene.scf.in > graphene.scf.out
ibrun wannier90.x silicon.win
```

# 1: Graphene — Projectability-disentangled Wannier functions

## Outline

Obtain MLWFs for graphene using projectability disentanglement. For more details on the methodology, see Ref.[1].

## Input files

- Directory: `ex1/`
- `graphene.scf` The `pw.x` input file for ground state calculation

**graphene.scf**

**Input file**

```
&CONTROL
  calculation = 'scf'
  etot_conv_thr =   2.0000000000d-05
  forc_conv_thr =   1.0000000000d-04
  outdir = './out/'
  prefix = 'graphene'
  pseudo_dir = 'pseudo/'
  tprnfor = .true.
  tstress = .true.
  verbosity = 'high'
/
&SYSTEM
  degauss =   1.0000000000d-02
  ecutrho =   1.4700000000d+02
  ecutwfc =   3.7000000000d+01
  ibrav = 0
  nat = 2
  nosym = .false.
  ntyp = 1
  occupations = 'smearing'
  smearing = 'cold'
/
&ELECTRONS
  conv_thr =   4.0000000000d-10
  electron_maxstep = 80
  mixing_beta =   4.0000000000d-01
/
ATOMIC_SPECIES
C      12.011 C.pbe-n-kjpaw_psl.0.1.UPF
ATOMIC_POSITIONS angstrom
C            0.0000000000       1.4202816622       0.0000000000
C            1.2300000000       0.7101408311       0.0000000000
K_POINTS automatic
9 9 1 0 0 0
CELL_PARAMETERS angstrom
      2.4600000000       0.0000000000       0.0000000000
     -1.2300000000       2.1304224933       0.0000000000
      0.0000000000       0.0000000000      20.0000000000
```

- `graphene.bands` The `pw.x` input file for band structure calculation

**graphene.bands**

**Input file**

```
&CONTROL
  calculation = 'bands'
  etot_conv_thr =   2.0000000000d-05
  forc_conv_thr =   1.0000000000d-04
  outdir = './out/'
  prefix = 'graphene'
  pseudo_dir = 'pseudo/'
  tprnfor = .true.
  tstress = .true.
  verbosity = 'high'
/
&SYSTEM
  degauss =   1.0000000000d-02
  ecutrho =   1.4700000000d+02
  ecutwfc =   3.7000000000d+01
  ibrav = 0
  nat = 2
  nbnd = 60
  nosym = .false.
  ntyp = 1
  occupations = 'smearing'
  smearing = 'cold'
/
&ELECTRONS
  conv_thr =   4.0000000000d-10
  diago_full_acc = .true.
! diagonalization = 'cg'
  electron_maxstep = 80
  mixing_beta =   4.0000000000d-01
/
ATOMIC_SPECIES
C      12.011 C.pbe-n-kjpaw_psl.0.1.UPF
ATOMIC_POSITIONS angstrom
C            0.0000000000       1.4202816622       0.0000000000
C            1.2300000000       0.7101408311       0.0000000000
CELL_PARAMETERS angstrom
      2.4600000000       0.0000000000       0.0000000000
     -1.2300000000       2.1304224933       0.0000000000
      0.0000000000       0.0000000000      20.0000000000
K_POINTS crystal
         274
    0.000000    0.000000    0.000000   1.0
    0.005000    0.000000    0.000000   1.0
    0.010000    0.000000    0.000000   1.0
    0.015000    0.000000    0.000000   1.0
    0.020000    0.000000    0.000000   1.0
    0.025000    0.000000    0.000000   1.0
    0.030000    0.000000    0.000000   1.0
...
...
```

- `graphene.bandsx` The `bands.x` input file for extracting band structure eigenvalues

**graphene.bandsx**

**Input file**

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

- `graphene.projwfc` The `projwfc.x` input file for projectability calculation

**graphene.projwfc**

**Input file**

```
&PROJWFC
  deltae =   2.0000000000d-01
  kresolveddos = .false.
  lbinary_data = .false.
  lsym = .false.
  lwrite_overlaps = .false.
  outdir = './out/'
  plotboxes = .false.
  prefix = 'graphene'
  tdosinboxes = .false.
  filproj = 'graphene.bands.dat.proj'
/
```

- `graphene.plotband` The `plotband.x` input file for plotting band structure

**graphene.plotband**

**Input file**

```
graphene.bands.dat
1 2 3 4 5 6 7 8
-21.9130   39.9100
graphene.plotband.gnu.dat
graphene.plotband.ps
-2.3043
5.0  -2.3043
graphene.projbands.gnu
```

- `graphene.nscf` The `pw.x` input file to obtain Bloch states on a uniform grid

**graphene.nscf**

**Input file**

```
&CONTROL
  calculation = 'nscf'
  etot_conv_thr =   2.0000000000d-05
  forc_conv_thr =   1.0000000000d-04
  max_seconds =   4.1040000000d+04
  outdir = './out/'
  prefix = 'graphene'
  pseudo_dir = 'pseudo/'
  tprnfor = .true.
  tstress = .true.
  verbosity = 'high'
/
&SYSTEM
  degauss =   1.0000000000d-02
  ecutrho =   1.4700000000d+02
  ecutwfc =   3.7000000000d+01
  ibrav = 0
  nat = 2
  nbnd = 60
  noinv = .true.
  nosym = .true.
  ntyp = 1
  occupations = 'smearing'
  smearing = 'cold'
/
&ELECTRONS
  conv_thr =   4.0000000000d-10
  diago_full_acc = .true.
  electron_maxstep = 80
  mixing_beta =   4.0000000000d-01
  startingpot = 'file'
/
ATOMIC_SPECIES
C      12.011 C.pbe-n-kjpaw_psl.0.1.UPF
ATOMIC_POSITIONS angstrom
C            0.0000000000       1.4202816622       0.0000000000
C            1.2300000000       0.7101408311       0.0000000000
CELL_PARAMETERS angstrom
      2.4600000000       0.0000000000       0.0000000000
     -1.2300000000       2.1304224933       0.0000000000
      0.0000000000       0.0000000000      20.0000000000
K_POINTS crystal
81
  0.00000000  0.00000000  0.00000000  1.234568e-02
  0.00000000  0.11111111  0.00000000  1.234568e-02
  0.00000000  0.22222222  0.00000000  1.234568e-02
  0.00000000  0.33333333  0.00000000  1.234568e-02
  0.00000000  0.44444444  0.00000000  1.234568e-02
...
...
```

- `graphene.pw2wan` Input file for `pw2wannier90.x`

**graphene.pw2wan**

**Input file**

```
&INPUTPP
! use pseudo-atomic-orbital projection
  atom_proj = .true.
  outdir = './out/'
  prefix = 'graphene'
  seedname = 'graphene'
/
```

- `graphene.win` The `wannier90.x` input file

**graphene.win**

**Input file**

```
num_wann = 8
num_bands = 60

! use pseudo-atomic orbital projection
auto_projections = .true.

! enable projectability disentanglement
dis_froz_proj = .true.
dis_proj_max =   0.85
dis_proj_min =   0.01

! you can also enable energy window disentanglement, which
! will also freeze states inside inner window, so that those
! states are always reproduced.
fermi_energy =  -2.3043
! dis_froz_max =  0.5

num_iter = 4000
conv_tol =   2.0000000000d-10
conv_window = 3
dis_num_iter = 4000
dis_conv_tol =   2.0000000000d-10

mp_grid = 9, 9, 1

! restart = plot
bands_plot = .true.

begin unit_cell_cart
ang
      2.4600000000       0.0000000000       0.0000000000
     -1.2300000000       2.1304224933       0.0000000000
      0.0000000000       0.0000000000      20.0000000000
end unit_cell_cart

begin atoms_cart
ang
C         0.0000000000       1.4202816622       0.0000000000
C         1.2300000000       0.7101408311       0.0000000000
end atoms_cart

begin kpoint_path
G  0.0000000000  0.0000000000  0.0000000000  M  0.5000000000  0.0000000000
0.0000000000
M  0.5000000000  0.0000000000  0.0000000000  K  0.3333333333  0.3333333333
0.0000000000
K  0.3333333333  0.3333333333  0.0000000000  G  0.0000000000  0.0000000000
0.0000000000
end kpoint_path

begin kpoints
  0.00000000  0.00000000  0.00000000
```

```
  0.00000000  0.11111111  0.00000000
  0.00000000  0.22222222  0.00000000
  0.00000000  0.33333333  0.00000000
  0.00000000  0.44444444  0.00000000
  0.00000000  0.55555556  0.00000000
  0.00000000  0.66666667  0.00000000
...
...
```

- `graphene_bandsdiff.gnu` The gnuplot script to compare DFT and Wannier bands

**graphene_bandsdiff.gnu**

**Gnuplot script**

```
#!/usr/bin/env -S gnuplot -p
set terminal pdf enhanced color dashed lw 1 size 6in,9in
set output "graphene_bandsdiff.pdf"
set size ratio 1.5
fermi = -2.3043
# set style data dots
set key
set xrange [0: 4.02877]
set yrange [(-22.91167 - fermi) : (23.53815 - fermi)]
set arrow from  1.47463, (-22.91167 - fermi) to  1.47463,  (23.53815 -
fermi) nohead
set arrow from  2.32601, (-22.91167 - fermi) to  2.32601,  (23.53815 -
fermi) nohead
set xtics ("G"  0.00000,"M"  1.47463,"K"  2.32601,"G"  4.02877)
# scale QE x-axis to be consistent with w90
plot "graphene.bands.dat.gnu" u ($1*2.554):($2 - fermi) w l title "QE", \
     "graphene_band.dat" u 1:($2 - fermi) w l title "W90"
```

## Steps

1. Run `pw.x` to obtain the ground state of graphene

**Terminal**

```
pw.x < graphene.scf > scf.out
```

2. Run `pw.x` to obtain the band structure of graphene

**Terminal**

```
pw.x < graphene.bands > bands.out
```

3. Run `projwfc.x` to obtain the band structure projectability of graphene

**Terminal**

```
projwfc.x < graphene.projwfc > projwfc.out
```

4. Run `bands.x` to obtain a `graphene.bands.dat` file containing the band structure of graphene

**Terminal**

```
bands.x < graphene.bandsx > bandsx.out
```

5. Run `plotband.x` to plot the band structure with projectability for graphene

> **Note**
>
> Run `plotband.x` interactively to see the meaning of each line in the input file.

a. First rename file so that it can be recognized by `plotband.x`

**Terminal**

```
mv graphene.bands.dat.proj.projwfc_up graphene.bands.dat.proj
```

b. Run `plotband.x`

**Terminal**

```
plotband.x < graphene.plotband > plotband.out
```

c. Generate a `graphene.projbands.gnu_projected.ps` file

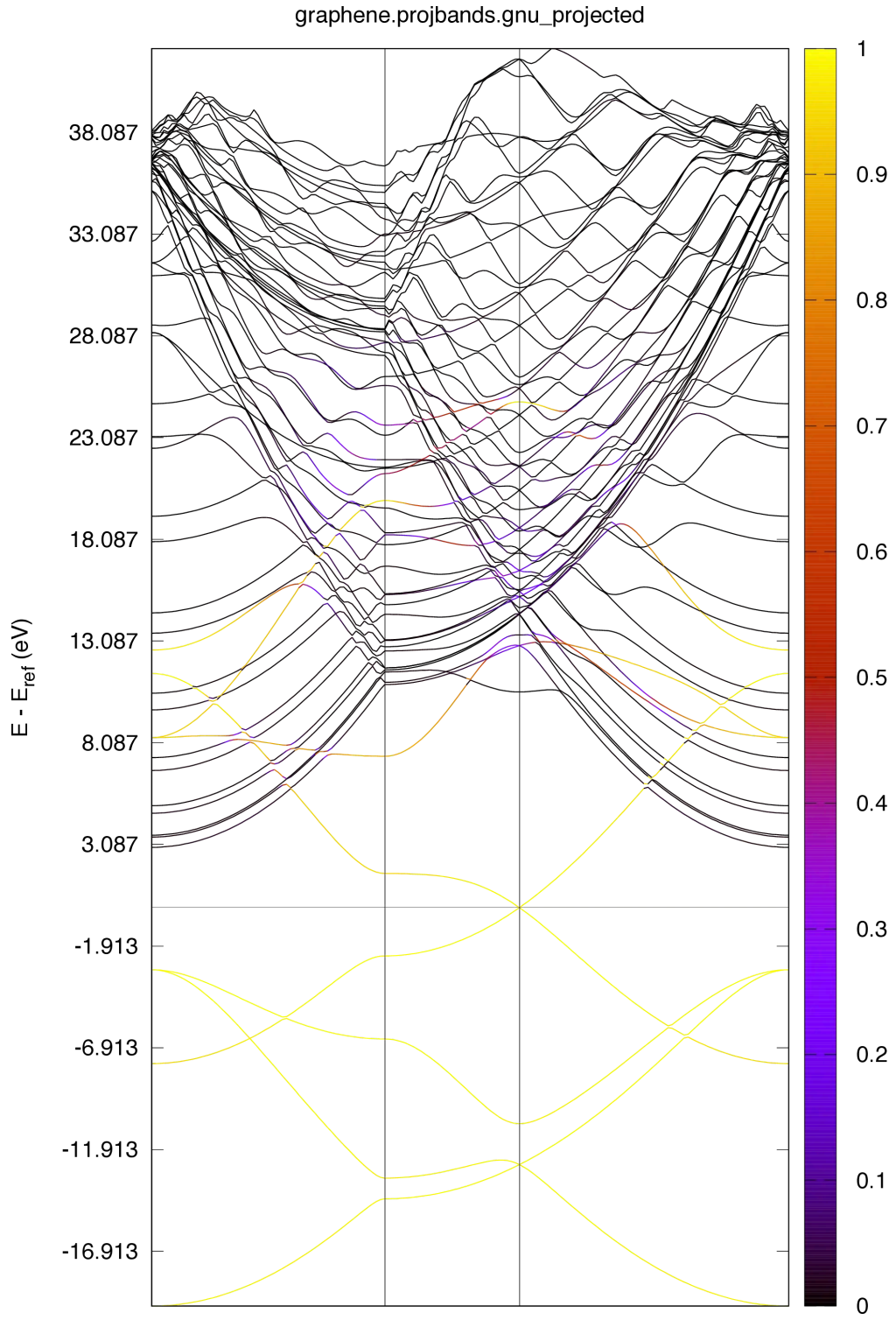
**Terminal**

```
gnuplot graphene.projbands.gnu
```

d. Generate a `graphene.projbands.gnu_projected.pdf` file, see the following Fig. Projected bands

**Terminal**

```
ps2pdf graphene.projbands.gnu_projected.ps
```




*Band structure of graphene with projectability.*

6. Run `pw.x` to obtain the Bloch states on a uniform k-point grid

**Terminal**

```
pw.x < graphene.nscf > nscf.out
```

7. Run `pw.x` to generate a list of the required overlaps (written into the `graphene.nnkp` file).

> **Note**
>
> See `win` input file, no need to specify initial projections, they are automatically chosen from the pseudo-atomic orbitals inside pseudopotentials used in the scf calculation.

**Terminal**

```
wannier90.x -pp graphene
```

8. Run `pw2wannier90.x` to compute the overlap between Bloch states and the projections for the starting guess (written in the `graphene.mmn` and `graphene.amn` files).

**Terminal**

```
pw2wannier90.x < graphene.pw2wan > pw2wan.out
```

9. Run  to compute the MLWFs.

**Terminal**

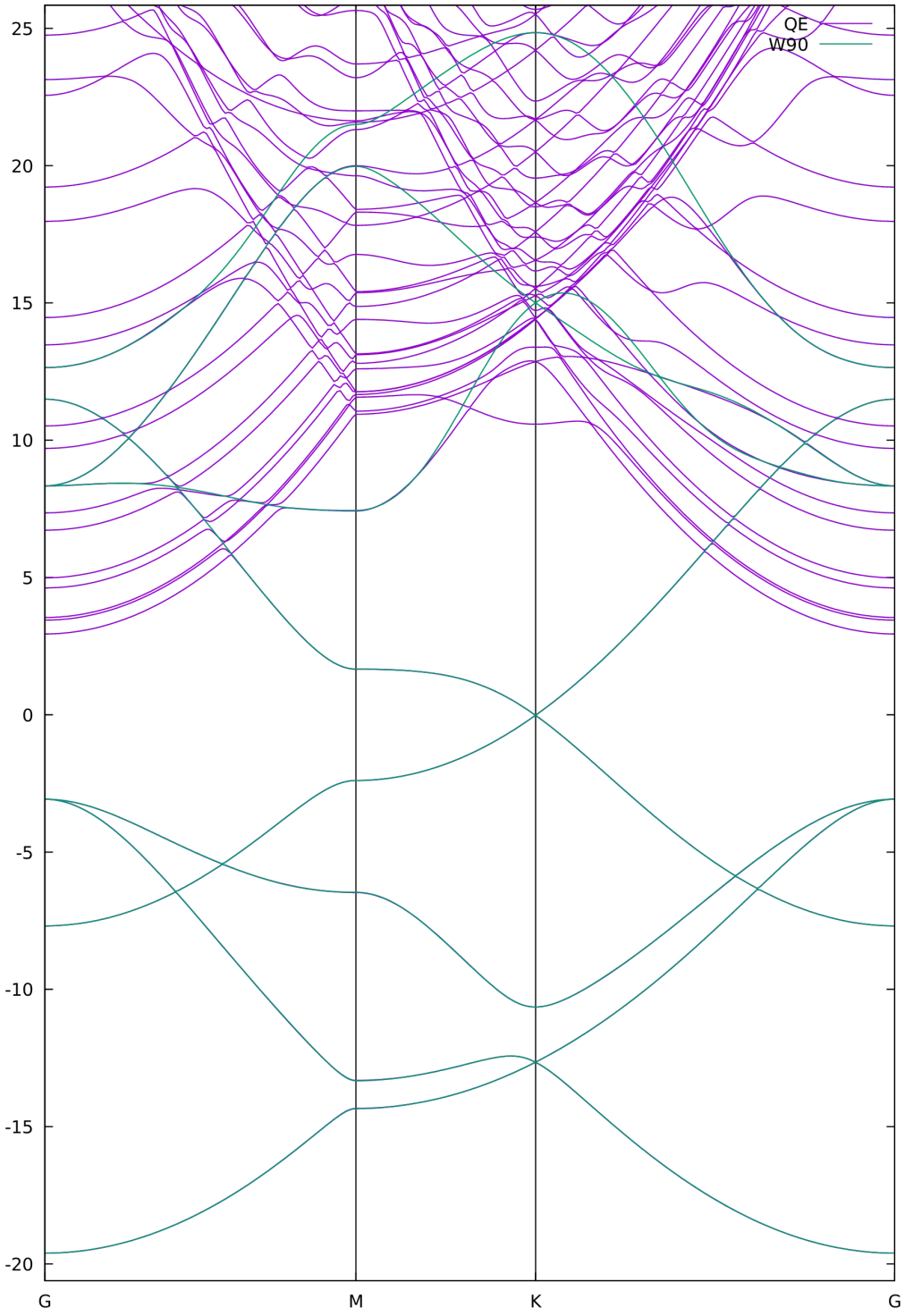
```
wannier90.x graphene
```

10. Run `gnuplot` to compare DFT and Wannier-interpolated bands, this will generate a PDF file `graphene_bandsdiff.pdf`, see the following Fig. Bands comparison.

**Terminal**

```
./graphene_bandsdiff.gnu
```

Notice that high-projectability states in the conduction region are properly reproduced. Try commenting out the `dis_froz_proj, dis_proj_max/min` lines in the `win` input file, and use the energy disentanglement `dis_froz_max/min`, and compare the band interpolations.




*Comparison of DFT and Wannier bands.*

11. (Optional) Clean up all output files

**Terminal**

```
make clean
```

1. Junfeng Qiao, Giovanni Pizzi, and Nicola Marzari. Projectability disentanglement for accurate and automated electronic-structure Hamiltonians. *npj Comput. Mater.*, 9(1):208, Nov 2023. doi:10.1038/s41524-023-01146-w. ↩

2024-05-30 2024-05-27

# 2: Silicon — Projectability-disentangled Wannier functions with custom projectors

## Outline

Obtain MLWFs for silicon using projectability disentanglement, with additional $3d$ projectors to describe high-energy conduction bands. For more details on the methodology, see Ref.[1].

## Input files

- Directory: `ex2/`
- `silicon.scf` The `pw.x` input file for ground state calculation

**silicon.scf**

**Input file**

```
&CONTROL
  calculation = 'scf'
  etot_conv_thr =   2.0000000000d-05
  forc_conv_thr =   1.0000000000d-04
  outdir = './out/'
  prefix = 'silicon'
  pseudo_dir = 'pseudo/'
  verbosity = 'high'
/
&SYSTEM
  degauss =   1.0000000000d-02
  ecutwfc =   25
  ibrav = 0
  nat = 2
  nosym = .false.
  ntyp = 1
  occupations = 'smearing'
  smearing = 'cold'
/
&ELECTRONS
  conv_thr =   4.0000000000d-10
  electron_maxstep = 80
  mixing_beta =   4.0000000000d-01
/
ATOMIC_SPECIES
Si     28.085 Si.pbe-n-van.UPF
ATOMIC_POSITIONS angstrom
Si           4.0482056439       1.3494018813       4.0482056439
Si           0.0000000000       0.0000000000       0.0000000000
K_POINTS automatic
10 10 10 0 0 0
CELL_PARAMETERS angstrom
      0.0000000000       2.6988037626       2.6988037626
      2.6988037626       0.0000000000       2.6988037626
      2.6988037626       2.6988037626       0.0000000000
```

- `silicon.bands` The `pw.x` input file for band structure calculation

**silicon.bands**

**Input file**

```
&CONTROL
  calculation = 'bands'
  etot_conv_thr =   2.0000000000d-05
  forc_conv_thr =   1.0000000000d-04
  outdir = './out/'
  prefix = 'silicon'
  pseudo_dir = 'pseudo/'
  verbosity = 'high'
/
&SYSTEM
  degauss =   1.0000000000d-02
  ecutwfc =   25
  ibrav = 0
  nat = 2
  nbnd = 40
  nosym = .false.
  ntyp = 1
  occupations = 'smearing'
  smearing = 'cold'
/
&ELECTRONS
  conv_thr =   4.0000000000d-10
  diago_full_acc = .true.
  diagonalization = 'cg'
  electron_maxstep = 80
  mixing_beta =   4.0000000000d-01
/
ATOMIC_SPECIES
Si     28.085 Si.pbe-n-van.UPF
ATOMIC_POSITIONS angstrom
Si           4.0482056439       1.3494018813       4.0482056439
Si           0.0000000000       0.0000000000       0.0000000000
CELL_PARAMETERS angstrom
      0.0000000000       2.6988037626       2.6988037626
      2.6988037626       0.0000000000       2.6988037626
      2.6988037626       2.6988037626       0.0000000000
K_POINTS crystal
         451
    0.000000    0.000000    0.000000   1.0
    0.005000    0.000000    0.005000   1.0
    0.010000    0.000000    0.010000   1.0
    0.015000    0.000000    0.015000   1.0
    0.020000    0.000000    0.020000   1.0
    0.025000    0.000000    0.025000   1.0
    0.030000    0.000000    0.030000   1.0
...
...
```

- `silicon.bandsx` The `bands.x` input file for extracting band structure eigenvalues

**silicon.bandsx**

**Input file**

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

- `silicon.nscf` The `pw.x` input file to obtain Bloch states on a uniform grid

**silicon.nscf**

**Input file**

```
&CONTROL
  calculation = 'nscf'
  etot_conv_thr =   2.0000000000d-05
  forc_conv_thr =   1.0000000000d-04
  max_seconds =   4.1040000000d+04
  outdir = './out/'
  prefix = 'silicon'
  pseudo_dir = 'pseudo/'
  verbosity = 'high'
/
&SYSTEM
  degauss =   1.0000000000d-02
  ecutwfc =   25
  ibrav = 0
  nat = 2
  nbnd = 40
  noinv = .true.
  nosym = .true.
  ntyp = 1
  occupations = 'smearing'
  smearing = 'cold'
/
&ELECTRONS
  conv_thr =   4.0000000000d-10
  diago_full_acc = .true.
  electron_maxstep = 80
  mixing_beta =   4.0000000000d-01
  startingpot = 'file'
/
ATOMIC_SPECIES
Si     28.085 Si.pbe-n-van.UPF
ATOMIC_POSITIONS angstrom
Si           4.0482056439       1.3494018813       4.0482056439
Si           0.0000000000       0.0000000000       0.0000000000
CELL_PARAMETERS angstrom
      0.0000000000       2.6988037626       2.6988037626
      2.6988037626       0.0000000000       2.6988037626
      2.6988037626       2.6988037626       0.0000000000
K_POINTS crystal
1000
  0.00000000  0.00000000  0.00000000  1.000000e-03
  0.00000000  0.00000000  0.10000000  1.000000e-03
  0.00000000  0.00000000  0.20000000  1.000000e-03
  0.00000000  0.00000000  0.30000000  1.000000e-03
  0.00000000  0.00000000  0.40000000  1.000000e-03
...
...
```

- `silicon.pw2wan` Input file for `pw2wannier90.x`

**silicon.pw2wan**

**Input file**

```
&INPUTPP
  atom_proj = .true.
  ! use external projectors
  atom_proj_ext = .true.
  ! dir for external projectors
  atom_proj_dir = './ext_proj'
  outdir = './out/'
  prefix = 'silicon'
  seedname = 'silicon'
  ! for excluding specific projectors
  ! this excludes 3d projectors, then the results are similar
  ! to that of using UPF file, i.e., project onto Si s+p orbitals
  ! for the indices of orbitals, see the pw2wan stdout
  ! atom_proj_exclude = 5 6 7 8 9 14 15 16 17 18
/
```

- `silicon.win` The `wannier90.x` input file

**silicon.win**

**Input file**

```
num_wann = 18
num_bands = 40

auto_projections = .true.

dis_froz_proj = .true.
dis_proj_max =   0.95
dis_proj_min =   0.01

! you can also add an energy window to be safe
dis_froz_max = 8.7

fermi_energy =   6.7332

num_iter = 4000
conv_tol =   2.0000000000d-10
conv_window = 3
dis_conv_tol =   2.0000000000d-10
dis_num_iter = 4000
num_cg_steps = 200

mp_grid = 10 10 10

bands_plot = .true.
begin atoms_cart
ang
Si         4.0482056439       1.3494018813       4.0482056439
Si         0.0000000000       0.0000000000       0.0000000000
end atoms_cart

begin unit_cell_cart
ang
      0.0000000000       2.6988037626       2.6988037626
      2.6988037626       0.0000000000       2.6988037626
      2.6988037626       2.6988037626       0.0000000000
end unit_cell_cart

begin kpoint_path
G  0.0000000000  0.0000000000  0.0000000000  X  0.5000000000  0.0000000000
0.5000000000
X  0.5000000000  0.0000000000  0.5000000000  U  0.6250000000  0.2500000000
0.6250000000
K  0.3750000000  0.3750000000  0.7500000000  G  0.0000000000  0.0000000000
0.0000000000
G  0.0000000000  0.0000000000  0.0000000000  L  0.5000000000  0.5000000000
0.5000000000
L  0.5000000000  0.5000000000  0.5000000000  W  0.5000000000  0.2500000000
0.7500000000
W  0.5000000000  0.2500000000  0.7500000000  X  0.5000000000  0.0000000000
0.5000000000
end kpoint_path

```

```
begin kpoints
  0.00000000  0.00000000  0.00000000
  0.00000000  0.00000000  0.10000000
  0.00000000  0.00000000  0.20000000
...
...
```

- `silicon_bandsdiff.gnu` The gnuplot script to compare DFT and Wannier bands

**silicon_bandsdiff.gnu**

**Gnuplot script**

```
#!/usr/bin/env -S gnuplot -p
set terminal pdf enhanced color dashed lw 1 size 6in,6in
set output "silicon_bandsdiff.pdf"
set size ratio 1
fermi = 6.7332
# set style data dots
set key
set xrange [0: 5.22358]
set yrange [ (-6.69831 - fermi) : (34.57817 - fermi)]
set arrow from  1.16407,  (-6.69831 - fermi) to  1.16407,  (34.57817 -
fermi) nohead
set arrow from  1.57563,  (-6.69831 - fermi) to  1.57563,  (34.57817 -
fermi) nohead
set arrow from  2.81031,  (-6.69831 - fermi) to  2.81031,  (34.57817 -
fermi) nohead
set arrow from  3.81842,  (-6.69831 - fermi) to  3.81842,  (34.57817 -
fermi) nohead
set arrow from  4.64154,  (-6.69831 - fermi) to  4.64154,  (34.57817 -
fermi) nohead
set xtics ("G"  0.00000,  "X"  1.16407,  "U|K"  1.57563,  "G"  2.81031, \
          "L"  3.81842,  "W"  4.64154,  "X"  5.22358)
# scale QE x-axis to be consistent with w90
plot "silicon.bands.dat.gnu" u ($1*1.6459):($2 - fermi) w l title "QE", \
     "silicon_band.dat" u 1:($2 - fermi) w l title "W90"
```

## Steps

1. Run `pw.x` to obtain the ground state of silicon

**Terminal**

```
pw.x < silicon.scf > scf.out
```

2. Run `pw.x` to obtain the band structure of silicon

**Terminal**

```
pw.x < silicon.bands > bands.out
```

3. Run `bands.x` to obtain a `silicon.bands.dat` file containing the band structure of silicon

**Terminal**

```
bands.x < silicon.bandsx > bandsx.out
```

4. Run `pw.x` to obtain the Bloch states on a uniform k-point grid

**Terminal**

```
pw.x < silicon.nscf > nscf.out
```

5. Run `pw.x` to generate a list of the required overlaps (written into the `silicon.nnkp` file).

> **Note**
>
> See `win` input file, no need to specify initial projections, they are chosen from the pseudo-atomic orbitals inside the `ext_proj/Si.dat` file.

**Terminal**

```
wannier90.x -pp silicon
```

6. Run `pw2wannier90.x` to compute the overlap between Bloch states and the projections for the starting guess (written in the `silicon.mmn` and `silicon.amn` files).

**Terminal**

```
pw2wannier90.x < silicon.pw2wan > pw2wan.out
```

7. Run `pw.x` to compute the MLWFs.

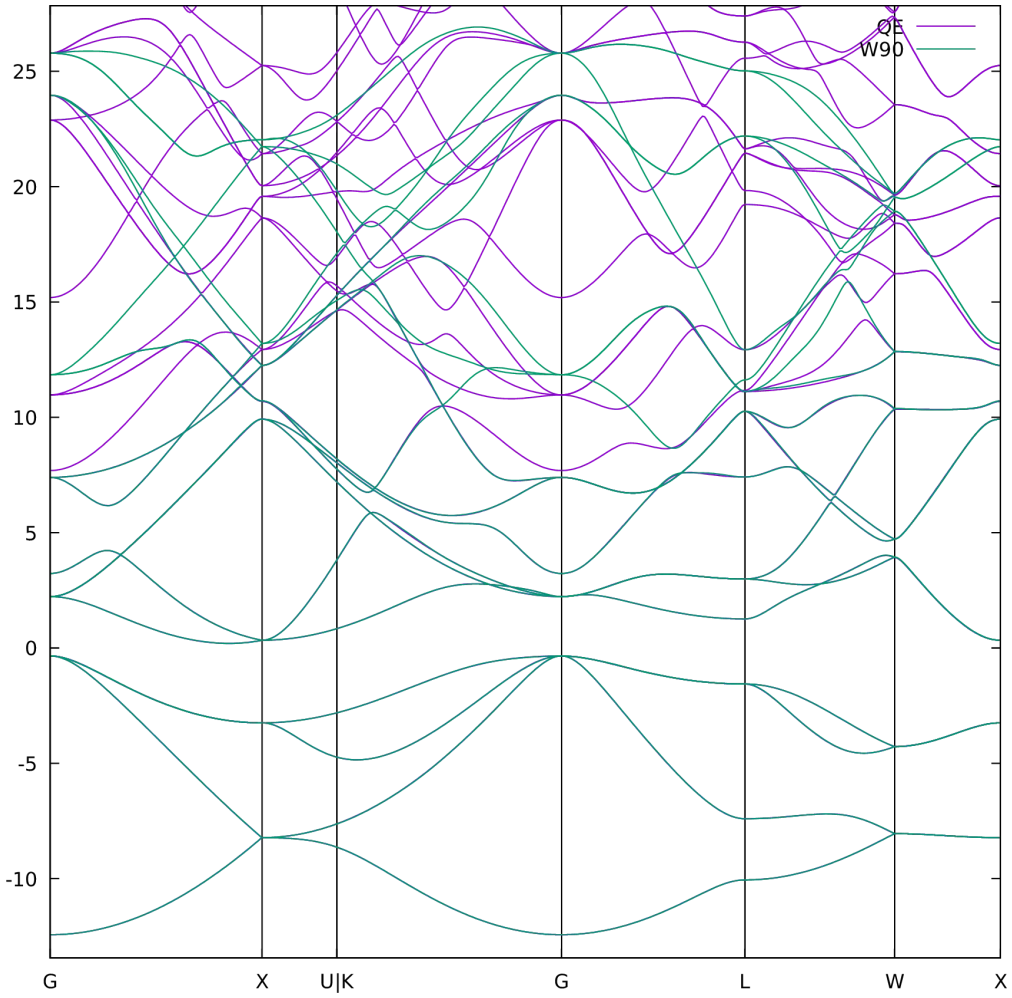
**Terminal**

```
wannier90.x silicon
```

8. Run `gnuplot` to compare DFT and Wannier-interpolated bands, this will generate a PDF file `silicon_bandsdiff.pdf`, see Fig.Bands comparison.

**Terminal**

```
./silicon_bandsdiff.gnu
```




*Comparison of DFT and Wannier bands for silicon.*

9. (Optional) Clean up all output files

**Terminal**

```
make clean
```

## Further ideas

1. Try changing the `atom_proj_exclude` in `silicon.pw2wan` file, i.e., these commented lines

**Input file**

```
10    ! for excluding specific projectors
11    ! this excludes 3d projectors, then the results are similar
12    ! to that of using UPF file, i.e., project onto Si s+p orbitals
      ! for the indices of orbitals, see the pw2wan stdout
```

```
13    ! atom_proj_exclude = 5 6 7 8 9 14 15 16 17 18
14
```

**Hint**

You need to set `num_wann = 8` in the `silicon.win` file as well, to be consistent with the reduced number of projectors.

2. Now that $3d$ projectors provide us a larger space for optimization, you can try increasing the `dis_froz_max` to freeze higher energy bands, if you are targeting at reproducing those eigenvalues.

**Note**

The `dis_proj_min/max` and `dis_froz_min/max` can be enabled simultaneously: the union of inner energy window and high-projectability states will be freezed, and the union of states outside outer energy window and having low projectability will be discarded. Thus, you can still use energy window to make sure near-Fermi energy states are well reproduced, and use "projectability window" to selectively freeze atomic-like states in the conduction region.

3. The default `dis_proj_max = 0.95` might not freeze all the states you want, try changing this value and see the band interpolation results. For other materials, it might worth trying decreasing this value to freeze more states.

---

1. Junfeng Qiao, Giovanni Pizzi, and Nicola Marzari. Projectability disentanglement for accurate and automated electronic-structure Hamiltonians. *npj Comput. Mater.*, 9(1):208, Nov 2023. doi:10.1038/s41524-023-01146-w. ↩

2024-05-30 2024-05-27

# 3: Silicon — Manifold-remixed Wannier functions

## Outline

Separate silicon valence and conduction bands, using manifold-remixed Wannier functions (MRWF). For more details on the methodology, see Ref.[1].

## Input files

- Directory: `ex3/`
- `silicon.scf` The `pw.x` input file for ground state calculation

**silicon.scf**

**Input file**

```
&CONTROL
  calculation = 'scf'
  etot_conv_thr =   2.0000000000d-05
  forc_conv_thr =   1.0000000000d-04
  outdir = './out/'
  prefix = 'silicon'
  pseudo_dir = 'pseudo/'
  verbosity = 'high'
/
&SYSTEM
  degauss =   1.0000000000d-02
  ecutwfc =   25
  ibrav = 0
  nat = 2
  nosym = .false.
  ntyp = 1
  occupations = 'smearing'
  smearing = 'cold'
/
&ELECTRONS
  conv_thr =   4.0000000000d-10
  electron_maxstep = 80
  mixing_beta =   4.0000000000d-01
/
ATOMIC_SPECIES
Si     28.085 Si.pbe-n-van.UPF
ATOMIC_POSITIONS angstrom
Si           4.0482056439       1.3494018813       4.0482056439
Si           0.0000000000       0.0000000000       0.0000000000
K_POINTS automatic
10 10 10 0 0 0
CELL_PARAMETERS angstrom
      0.0000000000       2.6988037626       2.6988037626
      2.6988037626       0.0000000000       2.6988037626
      2.6988037626       2.6988037626       0.0000000000
```

- `silicon.bands` The `pw.x` input file for band structure calculation

**silicon.bands**

**Input file**

```
&CONTROL
  calculation = 'bands'
  etot_conv_thr =   2.0000000000d-05
  forc_conv_thr =   1.0000000000d-04
  outdir = './out/'
  prefix = 'silicon'
  pseudo_dir = 'pseudo/'
  verbosity = 'high'
/
&SYSTEM
  degauss =   1.0000000000d-02
  ecutwfc =   25
  ibrav = 0
  nat = 2
  nbnd = 40
  nosym = .false.
  ntyp = 1
  occupations = 'smearing'
  smearing = 'cold'
/
&ELECTRONS
  conv_thr =   4.0000000000d-10
  diago_full_acc = .true.
  diagonalization = 'cg'
  electron_maxstep = 80
  mixing_beta =   4.0000000000d-01
/
ATOMIC_SPECIES
Si     28.085 Si.pbe-n-van.UPF
ATOMIC_POSITIONS angstrom
Si           4.0482056439       1.3494018813       4.0482056439
Si           0.0000000000       0.0000000000       0.0000000000
CELL_PARAMETERS angstrom
      0.0000000000       2.6988037626       2.6988037626
      2.6988037626       0.0000000000       2.6988037626
      2.6988037626       2.6988037626       0.0000000000
K_POINTS crystal
         451
    0.000000    0.000000    0.000000   1.0
    0.005000    0.000000    0.005000   1.0
    0.010000    0.000000    0.010000   1.0
    0.015000    0.000000    0.015000   1.0
    0.020000    0.000000    0.020000   1.0
    0.025000    0.000000    0.025000   1.0
    0.030000    0.000000    0.030000   1.0
...
...
```

- `silicon.nscf` The `pw.x` input file to obtain Bloch states on a uniform grid

**silicon.nscf**

**Input file**

```
&CONTROL
  calculation = 'nscf'
  etot_conv_thr =   2.0000000000d-05
  forc_conv_thr =   1.0000000000d-04
  max_seconds =   4.1040000000d+04
  outdir = './out/'
  prefix = 'silicon'
  pseudo_dir = 'pseudo/'
  verbosity = 'high'
/
&SYSTEM
  degauss =   1.0000000000d-02
  ecutwfc =   25
  ibrav = 0
  nat = 2
  nbnd = 20
  noinv = .true.
  nosym = .true.
  ntyp = 1
  occupations = 'smearing'
  smearing = 'cold'
/
&ELECTRONS
  conv_thr =   4.0000000000d-10
  diago_full_acc = .true.
  electron_maxstep = 80
  mixing_beta =   4.0000000000d-01
  startingpot = 'file'
/
ATOMIC_SPECIES
Si     28.085 Si.pbe-n-van.UPF
ATOMIC_POSITIONS angstrom
Si           4.0482056439       1.3494018813       4.0482056439
Si           0.0000000000       0.0000000000       0.0000000000
CELL_PARAMETERS angstrom
      0.0000000000       2.6988037626       2.6988037626
      2.6988037626       0.0000000000       2.6988037626
      2.6988037626       2.6988037626       0.0000000000
K_POINTS crystal
1000
  0.00000000  0.00000000  0.00000000  1.000000e-03
  0.00000000  0.00000000  0.10000000  1.000000e-03
  0.00000000  0.00000000  0.20000000  1.000000e-03
  0.00000000  0.00000000  0.30000000  1.000000e-03
  0.00000000  0.00000000  0.40000000  1.000000e-03
...
...
```

- `silicon.pw2wan` Input file for `pw2wannier90.x`

**silicon.pw2wan**

**Input file**

```
&INPUTPP
  atom_proj = .true.
  outdir = './out/'
  prefix = 'silicon'
  seedname = 'silicon'
/
```

- `silicon.win` The `wannier90.x` input file

**silicon.win**

**Input file**

```
num_wann = 8
num_bands = 20

auto_projections = .true.

dis_froz_proj = .true.
dis_proj_max =   0.95
dis_proj_min =   0.01

! we use a large energy frozen window to reproduce higher conduction bands
dis_froz_max = 11

fermi_energy =   6.7332

num_iter = 4000
conv_tol =   2.0000000000d-10
conv_window = 3
dis_conv_tol =   2.0000000000d-10
dis_num_iter = 4000
num_cg_steps = 200

mp_grid = 10 10 10

bands_plot = .true.
begin atoms_cart
ang
Si         4.0482056439       1.3494018813       4.0482056439
Si         0.0000000000       0.0000000000       0.0000000000
end atoms_cart

begin unit_cell_cart
ang
      0.0000000000       2.6988037626       2.6988037626
      2.6988037626       0.0000000000       2.6988037626
      2.6988037626       2.6988037626       0.0000000000
end unit_cell_cart

begin kpoint_path
G  0.0000000000  0.0000000000  0.0000000000  X  0.5000000000  0.0000000000
0.5000000000
X  0.5000000000  0.0000000000  0.5000000000  U  0.6250000000  0.2500000000
0.6250000000
K  0.3750000000  0.3750000000  0.7500000000  G  0.0000000000  0.0000000000
0.0000000000
G  0.0000000000  0.0000000000  0.0000000000  L  0.5000000000  0.5000000000
0.5000000000
L  0.5000000000  0.5000000000  0.5000000000  W  0.5000000000  0.2500000000
0.7500000000
W  0.5000000000  0.2500000000  0.7500000000  X  0.5000000000  0.0000000000
0.5000000000
end kpoint_path

```

```
begin kpoints
  0.00000000  0.00000000  0.00000000
  0.00000000  0.00000000  0.10000000
  0.00000000  0.00000000  0.20000000
  0.00000000  0.00000000  0.30000000
  0.00000000  0.00000000  0.40000000
  0.00000000  0.00000000  0.50000000
  0.00000000  0.00000000  0.60000000
  0.00000000  0.00000000  0.70000000
...
...
```

- `mrwf.jl` The julia code to obtain MRWFs for valence and conduction bands

**mrwf.jl**

**Julia script**

```julia
#!/usr/bin/env -S julia --project=jl_project
# Split valence and conduction Wannier functions.
using Wannier

seedname = "silicon"
win = read_win("$seedname.win")

# number of valence bands
nval = 4

outdir_val = "val"
outdir_cond = "cond"

# read the valcond amn/mmn/eig/chk files
model = read_w90_with_chk(seedname)

# split the original val+cond MLWFs into 2 groups: val, cond
# the first 4 bands are valence bands, the rest are conduction bands
indices = [1:nval, (nval+1):n_wannier(model)]
outdirs = [outdir_val, outdir_cond]

println("Model will be split into $(length(indices)) groups")
for (i, idxs) in enumerate(indices)
    println("  Group $i:")
    println("    indices: $(idxs)")
    println("    outdir : $(outdirs[i])")
end

models_Us = split_wannierize(model, indices)

for (i, (m, U)) in enumerate(models_Us)
    @info "Group $i after parallel transport:" omega(m)
    println("\n")

    # write files
    outdir = outdirs[i]
    mkpath(outdir)
    seedname_i = joinpath(outdir, seedname)
    write_w90(seedname_i, m)

    # prepare win files for val and cond, with correct num_wann
    win_i = Dict(pairs(win))
    for k in [:num_bands, :dis_froz_proj, :dis_proj_min, :dis_proj_max,
        :dis_win_min, :dis_win_max, :dis_froz_min, :dis_froz_max,
        :auto_projections,
    ]
        pop!(win_i, k, nothing)
    end
    # just write a random projection as a placeholder
    win_i[:projections] = ["random"]
    win_i[:num_wann] = n_wannier(m)
    win_i[:num_iter] = 1000
```

```
    write_win("$seedname_i.win"; win_i...)

    # you need to execute w90 in each of these folders
    # - `val` for VB
    # - `cond` for CB
end
```

- `silicon_bandsdiff.gnu` The gnuplot script to compare DFT and Wannier bands

**silicon_bandsdiff.gnu**

**Gnuplot script**

```
#!/usr/bin/env -S gnuplot -p
set terminal pdf enhanced color dashed lw 1 size 6in,6in
set output "silicon_bandsdiff.pdf"
set size ratio 1
fermi = 6.7332
# set style data dots
set key
set xrange [0: 5.22358]
set yrange [ (-6.69831 - fermi) : (17.5 - fermi)]
set arrow from  1.16407,  (-6.69831 - fermi) to  1.16407,  (17.5 - fermi)
nohead
set arrow from  1.57563,  (-6.69831 - fermi) to  1.57563,  (17.5 - fermi)
nohead
set arrow from  2.81031,  (-6.69831 - fermi) to  2.81031,  (17.5 - fermi)
nohead
set arrow from  3.81842,  (-6.69831 - fermi) to  3.81842,  (17.5 - fermi)
nohead
set arrow from  4.64154,  (-6.69831 - fermi) to  4.64154,  (17.5 - fermi)
nohead
set xtics ("G"  0.00000,  "X"  1.16407,  "U|K"  1.57563,  "G"  2.81031, \
           "L"  3.81842,  "W"  4.64154,  "X"  5.22358)
# scale QE x-axis to be consistent with w90
plot "silicon.bands.dat.gnu" u ($1*1.6459):($2 - fermi) w l title "QE", \
     "silicon_band.dat" u 1:($2 - fermi) w l title "W90"
```

- `silicon_bandsdiff_val.gnu` The gnuplot script to compare Wannier bands for valence+conduction and valence.

**silicon_bandsdiff_val.gnu**

Gnuplot script

```
#!/usr/bin/env -S gnuplot -p
set terminal pdf enhanced color dashed lw 1 size 6in,6in
set output "silicon_bandsdiff_val.pdf"
set size ratio 1
fermi = 6.7332
# set style data dots
set key
set xrange [0: 5.22358]
set yrange [ (-6.69831 - fermi) : (17.5 - fermi)]
set arrow from  1.16407,  (-6.69831 - fermi) to  1.16407,  (17.5 - fermi)
nohead
set arrow from  1.57563,  (-6.69831 - fermi) to  1.57563,  (17.5 - fermi)
nohead
set arrow from  2.81031,  (-6.69831 - fermi) to  2.81031,  (17.5 - fermi)
nohead
set arrow from  3.81842,  (-6.69831 - fermi) to  3.81842,  (17.5 - fermi)
nohead
set arrow from  4.64154,  (-6.69831 - fermi) to  4.64154,  (17.5 - fermi)
nohead
set xtics ("G"  0.00000,  "X"  1.16407,  "U|K"  1.57563,  "G"  2.81031, \
           "L"  3.81842,  "W"  4.64154,  "X"  5.22358)
# scale QE x-axis to be consistent with w90
plot "silicon_band.dat" u 1:($2 - fermi) w l title "W90 valcond", \
     "val/silicon_band.dat" u 1:($2 - fermi) w l title "W90 val"
```

- `silicon_bandsdiff_cond.gnu` The gnuplot script to compare Wannier bands for valence+conduction and conduction.

**silicon_bandsdiff_cond.gnu**

**Gnuplot script**

```
#!/usr/bin/env -S gnuplot -p
set terminal pdf enhanced color dashed lw 1 size 6in,6in
set output "silicon_bandsdiff_cond.pdf"
set size ratio 1
fermi = 6.7332
# set style data dots
set key
set xrange [0: 5.22358]
set yrange [ (-6.69831 - fermi) : (17.5 - fermi)]
set arrow from  1.16407,  (-6.69831 - fermi) to  1.16407,  (17.5 - fermi)
nohead
set arrow from  1.57563,  (-6.69831 - fermi) to  1.57563,  (17.5 - fermi)
nohead
set arrow from  2.81031,  (-6.69831 - fermi) to  2.81031,  (17.5 - fermi)
nohead
set arrow from  3.81842,  (-6.69831 - fermi) to  3.81842,  (17.5 - fermi)
nohead
set arrow from  4.64154,  (-6.69831 - fermi) to  4.64154,  (17.5 - fermi)
nohead
set xtics ("G"  0.00000,  "X"  1.16407,  "U|K"  1.57563,  "G"  2.81031, \
           "L"  3.81842,  "W"  4.64154,  "X"  5.22358)
# scale QE x-axis to be consistent with w90
plot "silicon_band.dat" u 1:($2 - fermi) w l title "W90 valcond", \
     "cond/silicon_band.dat" u 1:($2 - fermi) w l title "W90 cond"
```

## Steps

1. Run `pw.x` to obtain the ground state of silicon

**Terminal**

```
pw.x < silicon.scf > scf.out
```

2. Run `pw.x` to obtain the band structure of silicon

**Terminal**

```
pw.x < silicon.bands > bands.out
```

3. Run `bands.x` to obtain a `silicon.bands.dat` file containing the band structure of silicon

**Terminal**

```
bands.x < silicon.bandsx > bandsx.out
```

4. Run `pw.x` to obtain the Bloch states on a uniform k-point grid

**Terminal**

```
pw.x < silicon.nscf > nscf.out
```

5. Run `pw.x` to generate a list of the required overlaps (written into the `silicon.nnkp` file).

**Terminal**

```
wannier90.x -pp silicon
```

6. Run `pw2wannier90.x` to compute the overlap between Bloch states and the projections for the starting guess (written in the `silicon.mmn` and `silicon.amn` files).

**Terminal**

```
pw2wannier90.x < silicon.pw2wan > pw2wan.out
```

7. Run `pw.x` to compute the MLWFs.

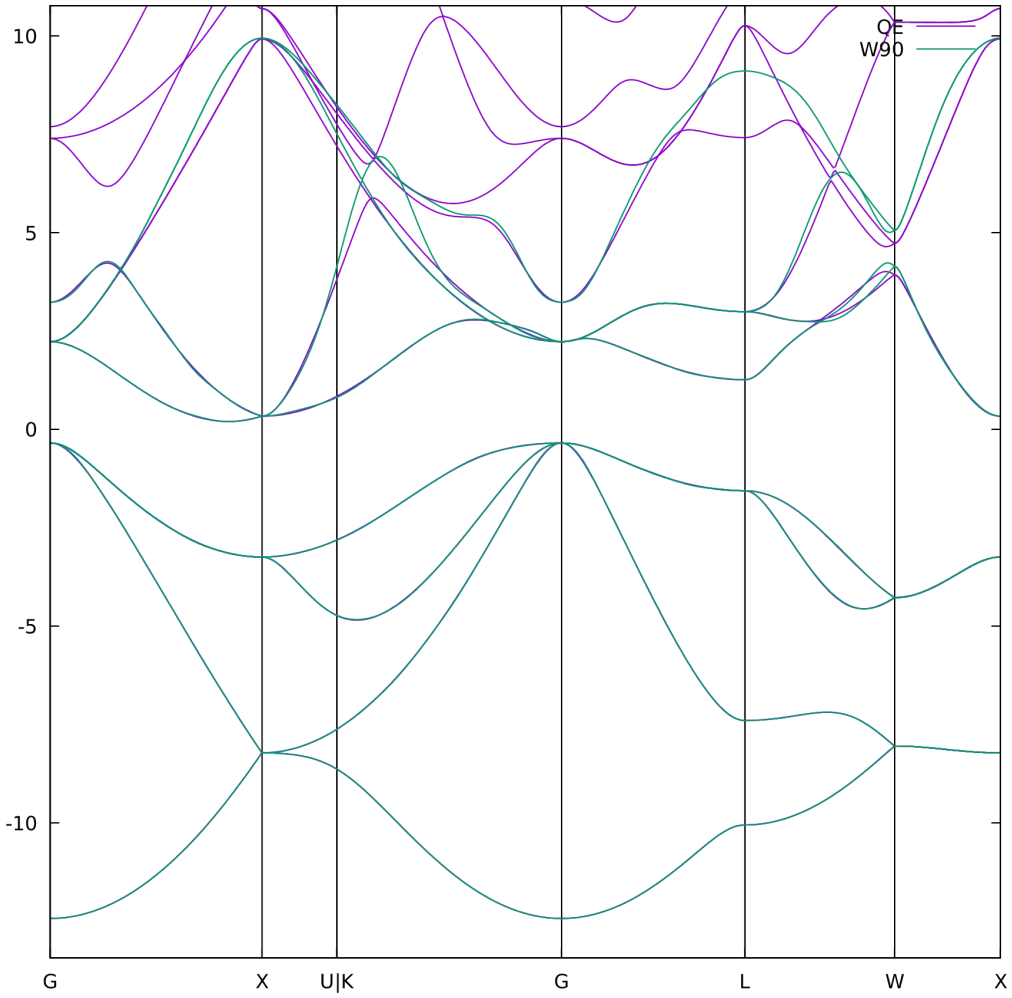
**Terminal**

```
wannier90.x silicon
```

8. Run `gnuplot` to compare DFT and Wannier-interpolated bands, this will generate a PDF file `silicon_bandsdiff.pdf`, see Fig.Bands comparison.

**Terminal**

```
./silicon_bandsdiff.gnu
```




*Comparison of DFT and Wannier bands for silicon.*

9. Run `mrwf.jl` to separate the valence and conduction bands.

**Terminal**

```
./mrwf.jl
```

> **Note**
>
> You might need to run the following command to install necessary Julia packages before running the `mrwf.jl` script.
>
> **Terminal**
>
> ```
> julia --project=jl_project -e 'using Pkg; Pkg.instantiate()'
> ```
>
> If you are wondering what is `--project=jl_project`, it is the directory where the `Project.toml` and `Manifest.toml` files are located. These two files are used to record the Julia package dependencies.

After this, you will have two folders: `val` and `cond` with following files, for valence and conduction bands, respectively.

**Terminal Output**

```
val
├── silicon.amn
├── silicon.eig
├── silicon.mmn
└── silicon.win
cond
├── silicon.amn
├── silicon.eig
├── silicon.mmn
└── silicon.win
```

10. Run `wannier90.x` to maximally localize the spread for the (now both are isolated bands) valence and conduction WFs.

    **Terminal**

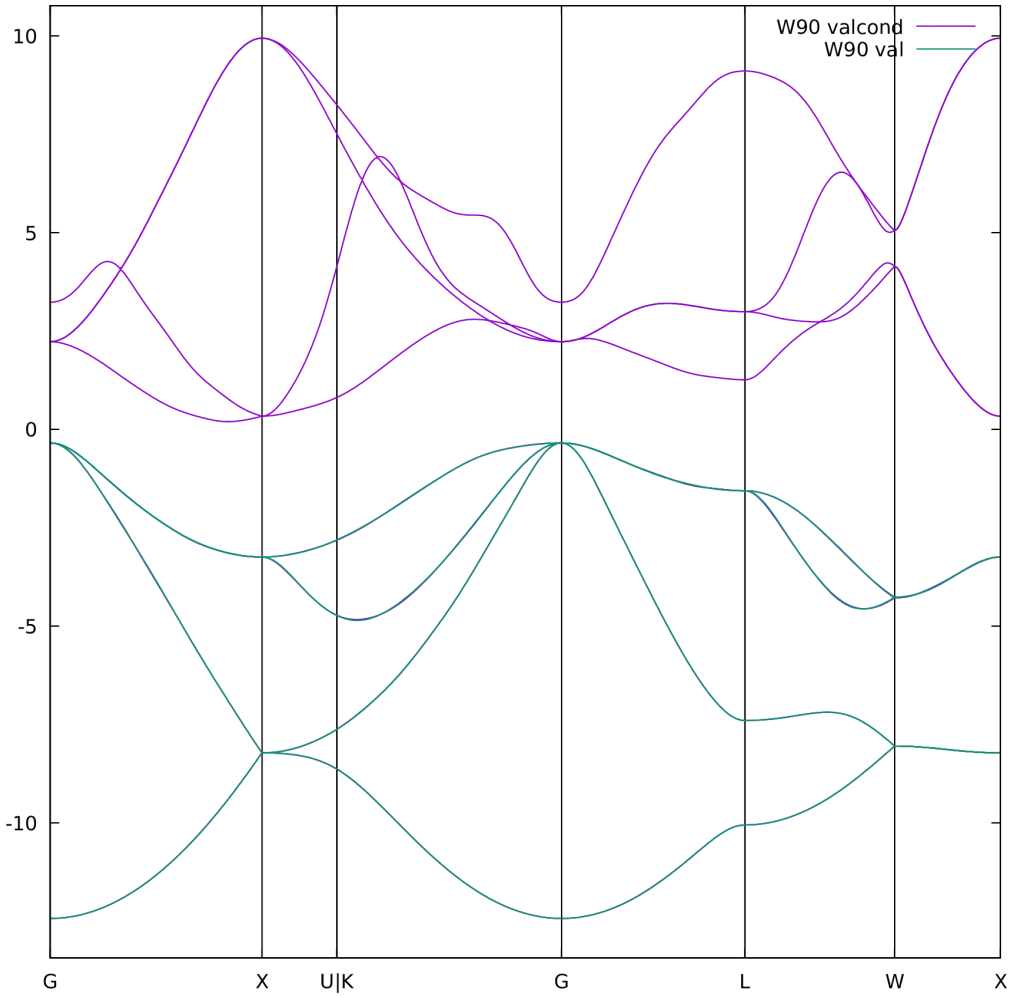
    ```
    cd val
    wannier90.x silicon
    cd ../cond
    wannier90.x silicon
    ```

11. Run `gnuplot` to compare Wannier-interpolated bands, of the three Wannierizations: valence+conduction, valence, and conduction. This will generate two PDF files:

    - `silicon_bandsdiff_val.pdf` (Fig.Bands comparison valence)
    - `silicon_bandsdiff_cond.pdf` (Fig.Bands comparison conduction)

    **Terminal**

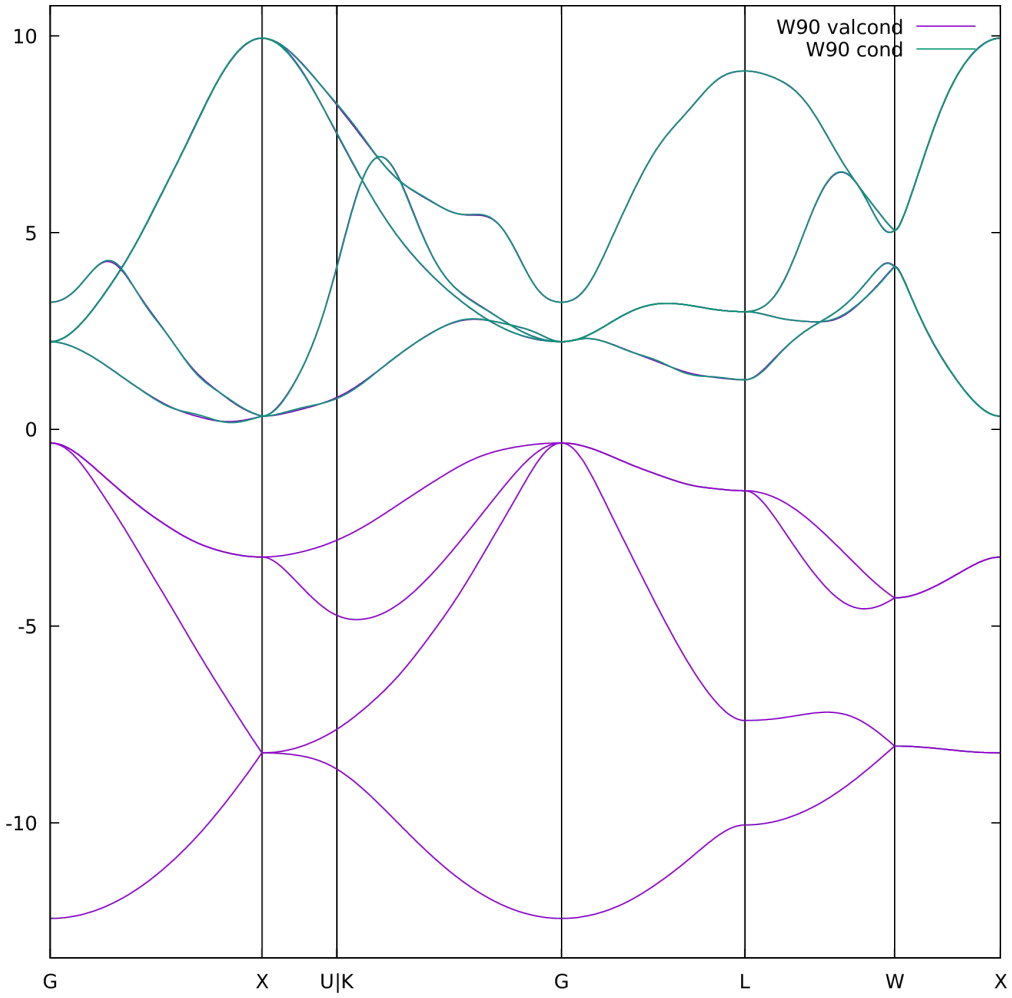
    ```
    ./silicon_bandsdiff_val.gnu
    ```




*Comparison of Wannier bands for valence+conduction and valence.*

**Terminal**

```
./silicon_bandsdiff_cond.gnu
```




*Comparison of Wannier bands for valence+conduction and conduction.*

12. (Optional) Clean up all output files

**Terminal**

```
make clean
```

## Summary

That's it! Though the steps may seem a bit lengthy, we have successfully obtained two sets of Wannier functions, without specifying any initial projections. We started with projectability-disentangled Wannier functions[2] to automatically Wannierize the valence+conduction bands of silicon, then separated them using manifold-remixed Wannier functions[1], finally, we maximally localized the spread of the two sets of Wannier functions, to obtain the MLWFs for valence and conduction bands, respectively.

## Further ideas

1. Try to understand the code in `mrwf.jl`.

---

1. Junfeng Qiao, Giovanni Pizzi, and Nicola Marzari. Automated mixing of maximally localized Wannier functions into target manifolds. *npj Comput. Mater.*, 9(1):206, Nov 2023. doi:10.1038/s41524-023-01147-9. ↩ ↩
2. Junfeng Qiao, Giovanni Pizzi, and Nicola Marzari. Projectability disentanglement for accurate and automated electronic-structure Hamiltonians. *npj Comput. Mater.*, 9(1):208, Nov 2023. doi:10.1038/s41524-023-01146-w. ↩

2024-05-30 2024-05-27