Tutorial: Electronic Structure of MoS_2 Using Quantum ESPRESSO and Wannier90, with Tight-Binding Hamiltonian Extraction

November 17, 2025

Contents

1 Introduction

This tutorial guides the reader through:

1. A complete Quantum ESPRESSO (QE) workflow to calculate the electronic band structure of monolayer MoS_2.
2. A fully automated Wannier90 workflow to construct a tight-binding (TB) model.

3. Generation and analysis of the real-space Wannier Hamiltonian $H(\mathbf{R})$ from `mos2_hr.dat`.

4. Python codes to:

   - reconstruct $H(\mathbf{k})$,
   - plot the Wannier-interpolated band structure,
   - compare with the QE band structure,

The goal is to obtain at least an 8-band TB model reproducing four valence and four conduction bands near the Fermi level.

2 Crystal Structure of Monolayer MoS_2

MoS_2 crystallizes in a hexagonal lattice with space group $P6_3/mmc$. A monolayer consists of a Mo plane sandwiched between two S planes.

A typical lattice constant is

$$a = 3.18 \text{ Å}.$$

We will construct a QE input using a 2D simulation cell with a vacuum region of about 15–20 Å in the z direction.

3 Self-Consistent Field Calculation (SCF)

3.1 Input File: `mos2_scf.in`

```
&control
    calculation = 'scf',
    prefix = 'mos2',
    outdir = './tmp'
/

&system
    ibrav = 4,
    a = 3.18,
    c = 20.0,
    nat = 3, ntyp = 2,
    ecutwfc = 60.0,
    ecutrho = 480.0,
    occupations='fixed'
/

&electrons
    conv_thr = 1.0d-8
/

ATOMIC_SPECIES
 Mo 95.94  Mo.pbesol-spn-kjpaw_psl.1.0.0.UPF
 S  32.06  S.pbesol-n-kjpaw_psl.1.0.0.UPF

ATOMIC_POSITIONS {crystal}
Mo 0.000 0.000 0.000
```

```
S  0.3333333 0.6666667 0.079
S  0.6666667 0.3333333 -0.079

K_POINTS automatic
12 12 1  0 0 0
```

Run it:

```
pw.x < mos2_scf.in > mos2_scf.out
```

4 Non-Self-Consistent Field Calculation (NSCF) for Band Structure

4.1 Input File: mos2_nscf.in

```
&control
    calculation='nscf',
    prefix='mos2',
    outdir='./tmp'
/

&system
    ibrav = 4,
    a = 3.18,
    c = 20.0,
    nat = 3, ntyp = 2,
    ecutwfc = 60.0,
    ecutrho = 480.0,
    nbnd = 60
/

&electrons
    conv_thr = 1.0d-8
/

ATOMIC_SPECIES
 Mo 95.94  Mo.pbesol-spn-kjpaw_psl.UPF
 S  32.06  S.pbesol-n-kjpaw_psl.UPF

ATOMIC_POSITIONS {crystal}
Mo 0.000 0.000 0.000
S  0.3333333 0.6666667 0.079
S  0.6666667 0.3333333 -0.079

K_POINTS automatic
21 21 1 0 0 0
```

Run:

```
pw.x < mos2_nscf.in > mos2_nscf.out
```

5 Band Structure Path

5.1 High-Symmetry Path K–Γ–M–K

```
K_POINTS crystal_b
4
0.333333 0.666667 0.0 40
0.000000 0.000000 0.0 40
0.500000 0.000000 0.0 40
0.333333 0.666667 0.0 40
```

5.2 Band Input: mos2_bands.in

```
&control
    calculation='bands',
    prefix='mos2',
    outdir='./tmp'
/

&system
    ibrav=4, a=3.18, c=20.0,
    nat=3, ntyp=2, ecutwfc=60, ecutrho=480,
    nbnd=60
/

&electrons
    conv_thr=1.0d-8
/

K_POINTS crystal_b
4
0.333333 0.666667 0 40
0.000000 0.000000 0 40
0.500000 0.000000 0 40
0.333333 0.666667 0 40
```

Run:

```
pw.x < mos2_bands.in > mos2_bands.out
bands.x < mos2_bands_post.in > mos2_bands_post.out
```

6 Wannier90 Workflow

6.1 Wannier90 Seed File: mos2.win

```
num_wann = 8
num_bands = 60
begin projections
  Mo:d
  S:p
end projections
```

```
begin kpoint_path
  K   0.333333 0.666667 0
  G   0.000000 0.000000 0
  M   0.500000 0.000000 0
  K   0.333333 0.666667 0
end kpoint_path

berry = false
spinors = false

write_hr = true
bands_plot = true
bands_plot_format = gnuplot

mp_grid = 21 21 1
```

6.2 Wannier Data Preparation

Run:

```
pw2wannier90.x < mos2_p2w.in > mos2_p2w.out
wannier90.x -pp mos2
wannier90.x mos2
```

After this, `mos2_hr.dat` will be generated containing the real-space tight-binding Hamiltonian.

7 Tight-Binding Formalism

The Wannier Hamiltonian is

$$H_{mn}(\mathbf{k}) = \sum_{\mathbf{R}} H_{mn}(\mathbf{R}) e^{i\mathbf{k}\cdot\mathbf{R}}.$$

The band energies are obtained from diagonalization:

$$H(\mathbf{k})\, u_{n\mathbf{k}} = \varepsilon_{n\mathbf{k}} u_{n\mathbf{k}}.$$

8 Python: Read `mos2_hr.dat` and Build $H(\mathbf{k})$

8.1 Full Wannier Band Structure Reconstruction

```
#!/usr/bin/env python3
import numpy as np
from numpy import linalg as LA
import matplotlib.pyplot as plt

# ------------------------------------------------------------
# Read Wannier90 hr.dat
# ------------------------------------------------------------
def read_hr(filename):
    f = open(filename, "r")
    num_wann = int(f.readline().split()[0])
```

```
    nrpts = int(f.readline().split()[0])

    # Degeneracies
    deg = []
    while len(deg) < nrpts:
        deg.extend([int(x) for x in f.readline().split()])
    deg = np.array(deg)

    # Hamiltonian blocks
    R = []
    H = np.zeros((nrpts, num_wann, num_wann), dtype=complex)

    for ir in range(nrpts):
        rvec = None
        for m in range(num_wann):
            for n in range(num_wann):
                line = f.readline().split()
                r1, r2, r3 = map(int, line[:3])
                mm = int(line[3]) - 1
                nn = int(line[4]) - 1
                re, im = float(line[5]), float(line[6])
                if rvec is None:
                    rvec = (r1, r2, r3)
                H[ir, mm, nn] = re + 1j*im
        R.append(rvec)
    f.close()
    return num_wann, np.array(R), deg, H

# ------------------------------------------------------------
# Build H(k)
# ------------------------------------------------------------
def Hk(kvec, num_wann, R, H_R):
    Hk = np.zeros((num_wann, num_wann), dtype=complex)
    for ir, rv in enumerate(R):
        phase = np.exp(1j * np.dot(kvec, rv))
        Hk += H_R[ir] * phase
    return Hk

# ------------------------------------------------------------
# Define high symmetry path
# ------------------------------------------------------------
def interpolate(a, b, n):
    return np.linspace(a, b, n, endpoint=False)

K = np.array([1/3, 2/3, 0])
G = np.array([0, 0, 0])
M = np.array([0.5, 0, 0])

path = []
path.extend(interpolate(K, G, 40))
path.extend(interpolate(G, M, 40))
path.extend(interpolate(M, K, 40))

# ------------------------------------------------------------
```

```
# Main
# ------------------------------------------------------------
num_wann, R, deg, H_R = read_hr("mos2_hr.dat")
bands = []

for k in path:
    e, _ = LA.eigh(Hk(k, num_wann, R, H_R))
    bands.append(e.real)

bands = np.array(bands)

# Plot
plt.figure(figsize=(6,4))
plt.plot(bands, color="black")
plt.ylabel("Energy␣(eV)")
plt.title("Wannier-interpolated␣band␣structure␣of␣MoS2")
plt.tight_layout()
plt.show()
```

9 Python: Compare Wannier Bands with QE Bands

Assume QE bands are stored in a plain-text file `qe_bands.dat` with columns $(k, E_1, E_2, \ldots)$.

```
import numpy as np
import matplotlib.pyplot as plt

qe = np.loadtxt("qe_bands.dat")
kqe = qe[:,0]
E_qe = qe[:,1:]

plt.figure(figsize=(7,4))
plt.plot(E_qe, "b-", lw=1, label="QE")

plt.plot(bands, "r--", lw=1, label="Wannier")
plt.legend()
plt.ylabel("Energy␣(eV)")
plt.title("QE␣vs␣Wannier␣band␣structure")
plt.tight_layout()
plt.show()
```

10 Interpreting the TB Model

The real-space Hamiltonian entries $H_{mn}(\mathbf{R})$ correspond to:

- on-site energies ($\mathbf{R} = 0$, $m = n$),
- local orbital hybridizations ($\mathbf{R} = 0$, $m \neq n$),
- hopping parameters between cells ($\mathbf{R} \neq 0$).

A simplified TB model is obtained by keeping only the largest nearest-neighbor hoppings indicated by the Python extractor.