Introduction

This tutorial describes how to calculate Heisenberg exchange parameters for a magnetic compound using Green's functions formalism. NiO – the charge transfer insulator with antiferromagnetic ordering will be considered as an example. A theoretical background can be found in the article Phys. Rrev. B **91**, 224405 (2015).

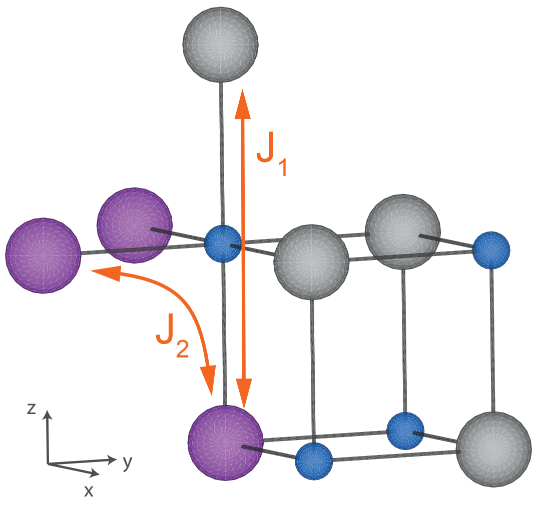

Figure 1: Schematic view of the NiO crystal structure. Blue spheres denote oxygen ions, gray and magenta spheres denote two magnetic types on Ni ions. Two types on Heisenberg exchanges between the Ni ions are shown.

NiO crystallizes in the rocksalt (NaCl) structure and exhibits an antiferromagnetic ordering of type-II with planes of opposite spins being repeated in alternating order along [111], see Fig. 1. The goal of the tutorial is to calculate the values of exchange parameters J_1 and J_2.

The calculation consists of three parts:

- Self consistent spin-polarized DFT calculation. Here and below we imply that you use Quantum ESPRESSO (QE) package for such calculation.
- Generation of the Hamiltonian in the Wannier functions basis. This basis should include d-states of magnetic ions and p-states of the nearest ligands. This could be done using ***wannier_ham.x*** code from QE postprocessing tools. The Hamiltonian should be stored in the $\mathcal{AMULET}$ file format. **Please use the latest stable QE version (5.3.0 and above) for the Hamiltonian production.**
- Calculation of the exchange parameters using ***exchanges.x*** code. The code uses as input only two files: ***system.am*** and ***hamilt.am*** from the previous step.

Step 1: Self consistent DFT calculation

It is assumed, that QE is already installed and you have an access to the standard QE pseudopotentials distribution. One needs to perform a spin-polarized calculations within LDA+U method with on-site Coulomb repulsion and intra-atomic Hund's coupling parameters, $U = 8.0$ eV and $J_H = 0.9$ eV. Below is an input file for Quantum ESPRESSO self-consistent run ***scf.in***:

```
&control
    calculation = 'scf'
    prefix='NiO',
    pseudo_dir='./'
/
&system
    ibrav=0, celldm(1)=7.8828134,
    nat=4, ntyp=3,
    ecutwfc = 45.0,
    ecutrho = 360
    occupations='smearing',
    degauss = 0.05
    nspin=2,
    starting_magnetization(1)= 0.5,
    starting_magnetization(2)=-0.5,
    lda_plus_u = .true.,
    U_projection_type = 'ortho-atomic'
    Hubbard_U(1) = 7.1,
    Hubbard_U(2) = 7.1
/
&electrons
/
CELL_PARAMETERS
0.50 0.50 1.00
0.50 1.00 0.50
1.00 0.50 0.50
ATOMIC_SPECIES
 Ni1  1.  Ni.pz-nd-rrkjus.UPF
 Ni2  1.  Ni.pz-nd-rrkjus.UPF
 O    1.  O.pz-rrkjus.UPF
ATOMIC_POSITIONS alat
 Ni1 0.0000000   0.0000000   0.0000000
 Ni2 0.0000000   0.0000000   1.0000000
 O   0.5000000   0.5000000   0.5000000
 O   1.5000000   1.5000000   1.5000000
K_POINTS automatic
 8 8 8 0 0 0
```

We have here two types on Ni ion (*Ni1* and *Ni2*). For both types of Ni ions we setup an opposite starting direction of magnetic moments (*starting_magnetization(i)*) and use the same value of the effective Hubbard interaction parameter $U_{eff} = U - J_H$ (*Hubbard_U(i)*). Please copy the pseudopotential files (***Ni.pz-nd-rrkjus.UPF*** and ***O.pz-rrkjus.UPF***) from ***exchanges/reference/*** folder to the same place where ***scf.in*** is located.

Then run Quantum ESRESSO on multiple processors or on single one (here we use 4 processors):

```
mpirun -n 4 $QE_PATH/pw.x < scf.in > scf.out &
```

After the calculation has been done, we have the ground state charge density (you can check the ***scf.out*** file to get the Ni d-states occupation, the Fermi energy, magnetization *etc*). Next one should obtain eigenvalues and eigenvectors of the total Hamiltonian on a regular k-points grid. To do that:

```
cp scf.in nscf.in
```

then edit ***nscf.in*** in the following way: change *calculation = 'scf'* → *calculation = 'nscf'*; add two more keys into the *&system* section:

```
&system
    ...
    nosym = .true.
    noinv = .true.
    ...
/
```

If you perform calculations in parallel mode, please set *wf_collect = .true.* in the *&control* section, since ***wannier_ham.x*** and ***exchanges.x*** codes are serial only.

Then run a non-selfconsistent calculation:

```
mpirun -n 4 $QE_PATH/pw.x < nscf.in > nscf.out &
```

Now we are ready to step into the next stage – construction of the Hamiltonian in the Wannier function basis.

Step 2: Generation of the Hamiltonian in Wannier functions basis.

The theoretical background for a Wannier functions generation and Hamiltonian production procedure is described in EPJB 65, 91 (2008). There is also an example in QE distributive (in ***PP/examples/WannierHam_example/*** directory).

Before start of the model Hamiltonian generation, you should know a symmetry of trial atomic orbitals that will be used for projection (typically these are transition metal d- plus, sometimes, the nearest ligands p-orbitals). And you should know numbers of bands (or energy interval) that you are going to reproduce with the Wannier Hamiltonian.

Since NiO is the charge transfer insulator with strong p-d hybridization, both Ni-d and O-p states are necessary to compute the exchange parameters. We construct a basis of 16 Wannier functions (WF): 5 WF are centered on the first Ni ion and have symmetry of d-orbitals, the next 5 WF are centered on the second Ni ion (have the same symmetry), other 6 WF are centered on two oxygen ions and have symmetry of O-p orbitals. Lets discuss the energy bands that one should choose to construct WF. The NiO is the simple compound. The pseudopotential files consider the following states as valence ones: $2s$, $2p$ for oxygen and $4s$, $3d$ for Ni. Therefore, the lowest (in energy) two bands are O-$2s$, then hybridized O-$2p$+Ni-$3d$ subbands are located, that crosses the Fermi level and the other energy bands are located above. It is clear that we are interested in 16 energy bands starting from the third, i.e. the bands from 3 to 18.

Construct the following input file (usually it is called ***hamilt.in***) for ***wannier_ham.x*** program:

```
&inputpp
    prefix='NiO'
    outdir='./'
    nwan = 16
    plot_bands = .true.
    form='amulet'
/
WANNIER_AC
Wannier# 1 3 18
atom 1
d 1
...
(text truncated  just copy hamilt.in file from the ./reference/ folder)
...
```

It is obvious that the *prefix* and *outdir* in ***hamilt.in*** should be the same as in the DFT calculation. The key *nwan* defines the number of Wannier functions, the string *form='amulet'* tells the code to export the Hamiltonian in format suitable for $\mathcal{AMULET}$. The key *plot_bands = .true.* allows one to export the eigenvalues in every **k**-point from the original DFT calculation and from transformed into the Wannier functions basis Hamiltonian. If the eigenvalues coincide for the chosen energy bands – the Hamiltonian construction was successful. The later part of the ***hamilt.in*** file contains *WANNIER_AC* section that describes the Wannier functions symmetry and location. The section *WANNIER_AC* has the following syntax:

```
    Wannier# 1 bands_from bands_to
    atom iatom
    l m
    Wannier# 2 bands_from bands_to
    atom iatom
    l m
    ...
    Spin#2:
    Wannier# 1 bands_from bands_to
    atom iatom
    l m
    Wannier# 2 bands_from bands_to
    atom iatom
    l m
    ...
```

bands_from, bands_to (real or integer)
Defines Bloch functions subspace for projection procedure. If *use_energy_interval=.true.* is set, these are energy values in eV. Otherwise these are bands numbers.

iatom (integer) – Number of site on that Wannier function centered.

l (character) – Angular channel for trial wavefunction. '*s*', '*p*' or '*d*'

m (integer) – Magnetic quantum number of trial orbital
(from 1 to 5 for d-orbitals, from 1 to 3 for p-orbitals)

Run the Hamiltonian construction on 1 processor in the directory with your scf calculation:

```
$QE_PATH/wannier_ham.x < hamilt.in > hamilt.out &
```

Please check obtained Wannier functions occupations. They should be reliable. If you obtain a lot of `wrong orthogonalization` warnings in output, it means that something is wrong with the energy bands selection.

As a result of Hamiltonian generation procedure at least four files will be produced: ***hamilt.am***, ***system.am***, ***original_bands.dat***, and ***wannier_bands.dat***.

The first two contains the Hamiltonian and some data about crystal structure of the compound **(If you didn't obtain `*.am` files - check *form = 'amulet'* parameter in the input file for *wannier_ham.x*)**. The last two contains eigenvalues of the original and model Hamiltonians. These files could be plotted with gnuplot and results should coincide within the energy window of interest. The coincidence of the eigenvalues is the main indicator of successful projection procedure.

All files that you need on the next step are ***hamilt.am*** and ***system.am***.

Step 3: Exchange parameters calculation with *exchanges.x*

The theoretical background for exchanges calculation is described in Phys. Rev. B 91, 224405 (2015). The ***exchanges.x*** code is available under the BSD license, the source could be downloaded from Github.

The code uses ***hamilt.am*** and ***system.am*** as input data. These two files could be produced by any DFT code or even generated by hands. Please make sure, that your starting Hamiltonian is spin-polarized. Most of input parameters are set by default and there is no need to change them in most cases. The main thing that could be tuned - *distance* parameter for the nearest neighbors search.

Input file for ***exchanges.x*** has similar to ***wannier_ham.x*** Fortran namelists form. Create file ***exchanges.in***:

```
&exchanges
 distance = 1.0
 mode = 'list'
/
ATOMS_LIST
3
 0.0  0.0  0.0
 0.0  0.0  1.0
 0.0 -0.5  0.5
```

The token *distance = 1.0* tells the code that we consider the exchange interaction between atoms located in a sphere centered on the first atom of the cell and with the radius equals 1.0*lattice parameter. But then, using *mode = 'list'* token we switch on the mode with manual atoms of interest definition (look at the *ATOMS_LIST*) below. These are coordinates of atoms, that correspond to J_1 and J_2 on the Fig. 1.

One can remove the *mode = 'list'* string. Then the code will find all 18 atoms within the 1.0*alat sphere and the calculation will take much more time.

Run the ***exchanges.x*** on one processor (it is serial code):

```
EX_PATH/exchanges.x < exchanges.in > exchanges.out &
```

Then look inside the exchanges.out to get the J_1 and J_2 exchange parameters:

```
...
Exchange interaction between atoms 1 and 2: -18.830 meV = -218 K (distance: 1.0)
...
Exchange interaction between atoms 1 and 3: -0.346 meV = -4 K (distance: 0.707)
...
```

It should be noted, that sign of the calculated exchange parameters is related to the spin ordering contained in the input Hamiltonian.