wan2respack Documentation

Release 1.0.0

wan2respack team

Feb 03, 2023

CONTENTS

CHAPTER

ONE

HOW TO USE WAN2RESPACK

1.1 Prerequisite

Required packages

Install the following three packages

1. Quantum Espresso (6.6)
2. Wannier90 (3.0.0)
3. RESPACK (20200113)

The versions in parentheses have been tested.

About Python

Python 3.6 or higher version is required.

Python requires tomli library. This tomli library is the standard library for python 3.11. Execute the following command.

```
pip install tomli
```

1.2 Structure

wan2respack has the following directory structure

```
|--CMakeLists.txt
|--LICENSE
|--README.md
|
|--config
|    |
|    |--intel.cmake
|    |--gcc.cmake
|
|--docs
|
|--samples
|    |
|    |--README.md
|    |--Al.fcc.666
```

(continues on next page)

(continued from previous page)

```
|    |--La2CuO4.bct.666
|    |--SrVO3.sc.666
|
|--util
     |
     |--wan2respack
```

1.3 Installation

wan2respack can be downloaded from the following GitHub page. https://github.com/respack-dev/wan2respack

Users can complie *wan2respack* by using CMake. An example for the installtion of *wan2respack* is as follows

```
cd $PATH_to_wan2respack
mkdir build
cd build
cmake ../ -DCONFIG=$Type_of_Configure -DCMAKE_INSTALL_PREFIX=$PATH_to_Install
make
make install
```

, where `$PATH_to_wan2respack` is the path to *wan2respack* directory and `$PATH_to_Install` is the path to the directory for the installation. By replacing `$Type_of_Configure` for a name of CMake configure files, users can specify compilers they want to use. In α version, the following `$Type_of_Configure` are available

```
intel: Intel compiler + MKL
gcc: GCC compiler
```

The details of compiler options can be found in CMake configure files in `$PATH_to_wan2respack/config` directory.

All the binary files and the Python script are installed to `$PATH_to_Install/bin`. Their details are as follows:

wan2respack.py

- Main Python script including two modes: pre-process and core-process. This script requires a configuration toml file as an argument.

gen_mk.x

- Fortran90 code for calculating k-points mesh for Wannier90. This binary is called by wan2respack_pre.py.

gen_wan.x

- Fortran90 code for converting the Wannier90 results into the RESPACK Wannier format. This binary is called by wan2respack_core.py.

wan2respack_pre.py

- Python script for saving the QE results and exporting k-points with gen_mk.x and qe2respack.py.

wan2respack_core.py

- Python script for preparing files about Wannier functions in RESPACK format with gen_wan.x and qe2respack.py.

qe2respack.py

- Python script for generating input files of RESPACK from QE band calculations. This script is originally distributed under GNU GPL ver.3 by open-source software RESPACK ver. 20200113.

`init.py`

- Python module in which common functions are defined.

1.4 Basic usage

1. Perform first principles calculations by QE
   - Only norm-conserving pseudopotentials can work in RESPACK.
   - Perform the calculation at the irreducible k-points.
2. Running wan2respack.py in pre-process mode
   - Export k-points to be calculated to nscf-input and wannier90-input.
3. Generate Wannier functions by QE & Wannier90
   - Use the input files made by the previous step.
4. Convert Wannier functions to RESPACK format by running wan2respack.py

1.4.1 Prepare input files

Prepare the following files. See *Input files* for details.

- QE scf input.
- QE nscf input.
  - Be sure to use *{automatic}* to set the k-point.
- wannier90 input file.
  - Do not write kpoints-block.
- pw2wannier90 input file.
- RESPACK input file.
- configuration toml file.

1.4.2 Run *wan2respack*

After the calculations at the irreducible k-points,

```
python wan2respack.py conf.toml -pp
```

The above command generates `new_nscf` and `new_win` files with the k-points list to be calculated.

After the Wannier functions are generated,

```
python wan2respack.py conf.toml
```

The `dir-wan` directory and four files inside it are generated.

CHAPTER

TWO

TUTORIAL

In this tutorial, we demonstrate the following calculations: (i) performing the first principle calculations using *QE* , (ii) calculating wannier functinos using *Wannier90* , (iii) generating input files for *RESPACK* using *wan2respack*, and (iV) calculating coulomb interactions using *RESPACK* . The sample files are located in `samples/Al.fcc.666`. In this directory, there are following four directories:

1. PP
   - Including files of pseudopotentials.
2. inputs
   - Including input files.
3. inputs_selfk
   - Including input files when setting k-points by yourself.
4. reference
   - Including reference input files for generating wannier functions using `calc_wannier` in *RESPACK* .

2.1 Al.fcc

The sample files of this tutorial are located in `samples/Al.fcc.666/inputs` . First, change the directory to `samples/Al.fcc.666/inputs`:

```
$cd samples/Al.fcc.666/inputs
```

In this directory, the following input files are included:

- QE
  - Al.scf.in: Input file for scf calculation.
  - Al.nscf.in: Input file for nscf calculation.
  - Al.pw2wan.in: Input file for generating `mmn` and `amn` files.
- Wannier90
  - Al.win.ref: Reference file of *wan2respack* for generating an input file of *Wannier90* .
- wan2respack
  - conf.toml: Input file of *wan2respack* for setting a path to the output directry of *QE* and seed names et al.
- RESPACK
  - input.in : Input file *RESPACK* .

2.1.1 First principle calculations for the irreducible k-points using *QE*.

Typing following commands, first princeple calculations of scf and nscf using *QE* will be performed:

```
$QE/bin/pw.x < Al.scf.in > Al.scf.out
$QE/bin/pw.x < Al.nscf.in > Al.nscf.out
```

Here, `$QE` indicates a path to an installing directory of *QE*. **It is noted that target k-points must be irreducible.**

2.1.2 Export k-points to be calculated by *Wannier90*.

Next, as a preprocess, export k-points to be calculated to input files of nscf calculation and wannier90 by typing a following command.

```
$python $PATH_to_Install/bin/wan2respack.py -pp conf.toml
```

The contents of `conf.toml` is shown below

```
[base]
QE_output_dir = "./work/Al.save"
seedname = "Al"

[pre.ref]
nscf = "Al.nscf.in"
win = "Al.win.ref"

[pre.output]
nscf = "Al.nscf_wannier.in"
win = "Al.win"
```

In `[base]` section, an output directory of *QE* and seed name are indicated by `QE_output_dir` and `seedname` , respectively. In `[pre.ref]` section, reference files for generating input files are indicated by `nscf` and `win` , respectively. In `[pre.output]` section, names of output files generated by *wan2respack* are indicated by `nscf` and `win` , respectively.

After finishing a calculation, `dir-wfn` directory, `Al.nscf_wannier.in` and `Al.win` files will be generated (k-points will be added to `Al.nscf_wannier.in` and `Al.win`).

2.1.3 Generate Wannier functions with *QE* & *Wannier90*.

Using `Al.nscf_wannier.in` and `Al.win` , Wannier functions are generated using *QE* & *Wannier90* by typing following commands.

```
$QE/bin/pw.x < Al.nscf_wannier.in > Al.nscf_wannier.out
$Wanier90/wannier90.x -pp Al
$QE/bin/pw2wannier90.x < Al.pw2wan.in > Al.pw2wan.out
$Wanier90/wannier90.x Al
```

2.1.4 Convert Wannier functions to *RESPACK* format

Converting wannier functions to *RESPACK* format can be done by using `wan2respack.py` The execution command is described below.

```
$python $PATH_to_Install/bin/wan2respack.py conf.toml
```

After finishing calculations, 4 files are generated in `dir-wan` directory.

2.1.5 Calculation of Coulomb interactions using *RESPACK*

Input file of *RESPACK* is prepared as `input.in` . Using this file, we can calculate Coulomb interactions using constrained Random Phase Approximation by *RESPACK* .

The execution command is described below.

```
$RESPACK/bin/calc_chiqw < input.in > LOG.chiqw
$RESPACK/bin/calc_w3d < input.in > LOG.W3d
$RESPACK/bin/calc_j3d < input.in > LOG.J3d
```

The obtained results are shown in Fig. 2.1. The horizontal axis corresponds to the distance and the vertical axis to the screened Coulomb interaction. In these figures, we also plotted the numerical results obtained by using *RESPACK*. In this case, the calculation of Wannier functions were performed by `calc_wannier` in *RESPACK*. The input file `input.in` of `calc_wannier` is located in `reference` directory. We can see that we obtained qualitatively almost the same trend, indicating that the tool is working well. The difference shown in the inset of Fig. 2.1 is due to the fact that the Wannier function obtained with Wannier90 is not the maximally localized (num_iter =0).

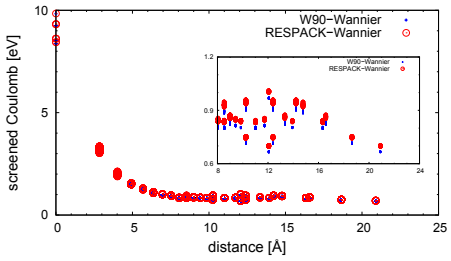

Fig. 2.1: Coulomb interactions obtained by constrained Random Phase Approximation. W90-Wannier indicates the numerical results obtained by this tutorial. RESAPCK-Wannier indicates the numerical results obtained by using *RESPACK* (the calculation of Wannier functions were performed by `calc_wannier` in *RESPACK*). The inset shows enlarged view of Coulomb interactions obtained by constrained Random Phase Approximation.

We also prepare sample files of La2CuO4 and SrVO3 in `samples` directory.

2.1.6 [Optional] Set k-points by yourself

In above tutorial, k-points are automatically exported. You can set k-points by yourself if you like, by typing the following command in the `inputs_selfk` directory.

```
$python $PATH_to_Install/bin/wan2respack.py -pp conf.toml
```

The contents of `conf.toml` is shown below

```
[base]
QE_output_dir = "./work/Al.save"
seedname = "Al"
selfk = true
```

`selfk` flag in `[base]` section must be `true` in this mode. The k-points list is written in `dat.sample_mk`. A k-points list in `Al.nscf_wannier.in` and `Al.win` is determined based on `dat.sample_mk`.

CHAPTER

THREE

FILE SPECIFICATION

3.1 Input files

This section explains all input files from SCF calculation to Coulomb interaction calculation.

- QE scf input file
- QE nscf input file
- **Wannier90 input file**
  Do not write k-points block. This file is used as only reference.
- pw2wannier90 input file
- RESPACK input file
- **conf.toml**
  The format is shown below.

```
[base]
QE_output_dir = "./work/prefix.save"
seedname = "seedname"
selfk = false            #(optional) Default: false

[pre.ref]
nscf = "nscf.in"
win = "seedname.win.ref"

[pre.output]
nscf = "nscf_wannier.in"
win = "seedname.win"
```

1. **base**
   - QE_output_dir : *QE* output directory
   - seedname : The same string as seedname used in *Wannier90*
   - selfk (optional, Default: false) : Flag to set k-points manually at the pre-process mode.
2. **pre.ref**
   - nscf : File name of the *QE* nscf input file you prepared
   - win : File name of the *Wannier90* input file you prepared
3. **pre.output**

  - `nscf` : File name of the new *QE* nscf input file that is automatically generated based on `[pre.ref]nscf`.
  - `win` : File name of the new *Wannier90* input that is automatically generated base on `[pre.ref]win`

3.2 Output files

The details of the output files are explained.

3.2.1 pre-process

- **`[pre.output]nscf`**
  The *QE* input file whose name is determined by `[pre.output]nscf` in conf.toml. This file is automatically made based on the reference file, `[pre.ref]nscf`.
- **`[pre.output]win`**
  The *Wannier90* input file whose name is determined by `[pre.output]win` in conf.toml. This file is automatically made based on the reference file, `[pre.ref]win`.
- **dat.sample_mk**
  The intermediate file for making input files of *QE* and *Wannier90*, including k-points. The first line gives the total number of k-points. The next block gives k-points in terms of the reciprocal lattice vectors.
- **dat.kg_respack**
  The intermediate file for making `dat.wan`, including G-vectors. This file consists of the number of blocks equal to the total number of k-points. The first line of each block gives the number of G-vectors. The remaining lines of each block give the G-vectors in terms of the reciprocal lattice vectors.
- **LOG.mk**
  Log file.

3.2.2 core-process

- **dat.ns-nb, dat.umat, dat.wan, dat.wan-center**
  These files includes information about Wannier functions. The file format is the same as *RESPACK*. See *Relations between expressions and file names* for details.
- **LOG.genwan**
  Log file.

CHAPTER

FOUR

ALGORITHM

In RESPACK, the i-th Wannier function is defined as

$$w_{i,0}(\mathbf{r}) = \frac{1}{\sqrt{N_k}} \sum_{\mathbf{k}} \sum_{m=1}^{N_w} \sum_{n=N_s}^{N_s+N_b} U_{im}^{\mathbf{k}} U_{mn}^{\mathbf{k},opt} \psi_{nk}(\mathbf{r}) \tag{4.1}$$

Here, U, U^{opt} are unitary matrix, $\psi_{n\mathbf{k}}(r)$ is n-th Bloch wave function, N_s and N_b are determined from the energy-window information. The Bloch wave function is defined as

$$\psi_{n\mathbf{k}} = \frac{1}{\sqrt{N_k}} \frac{1}{\sqrt{\Omega}} \sum_{\mathbf{G}} C_{\mathbf{G}n}(\mathbf{k}) e^{i(\mathbf{k}+\mathbf{G})\cdot\mathbf{r}} \tag{4.2}$$

Here, $C_{\mathbf{G}n}$ is the expansion coefficient of plane wave. By inserting this, the Wannier function is written as

$$w_{i,0}(\mathbf{r}) = \frac{1}{N_k} \sum_{\mathbf{k}} \sum_{\mathbf{G}} \tilde{C_{\mathbf{G}i}}(\mathbf{k}) \frac{1}{\sqrt{\Omega}} e^{i(\mathbf{k}+\mathbf{G})\cdot r} \tag{4.3}$$

Here, $\tilde{C_{\mathbf{G}i}}(\mathbf{k})$ is the expansion coefficient of the plane wave for the Wannier function:

$$\tilde{C_{\mathbf{G}i}}(\mathbf{k}) = \sum_{m=1}^{N_w} \sum_{n=N_s}^{N_s+N_b} U_{im}^{\mathbf{k}} U_{mn}^{\mathbf{k},opt} C_{\mathbf{G}n} \tag{4.4}$$

$U^{\mathbf{k},opt}$ is required at dielectric function calculation, and $\tilde{C_{\mathbf{G}i}}(\mathbf{k})$ is required at Coulomb calculation. This program generates $\tilde{C_{\mathbf{G}i}}(\mathbf{k})$ using $C_{\mathbf{G}n}$ made by QE and $U^{\mathbf{k}}, U^{\mathbf{k},opt}$ made by Wannier90.

K-points Order

RESPACK performs calculations based on the information at irreducible k-points ($\psi_{k\in irr}$). This is because the information at reduicble k-points ($\psi_{k\in reducible}$) can be generated by using the symmetry. On the other hand, Wannier90 calculates $U^{\mathbf{k}}$ and so on based on $\psi_{k\in reducible}$. The **k**-order of $\psi_{k\in reducible}$ used in Wannier90 must be the same as the **k**-order of $\psi_{k\in reducible}$ generated by the symmetry in RESPACK. In pre-process mode, an operation is performed to align the **k**-order.

4.1 Relations between expressions and file names

This program finally outputs the four files in `dir-wan` directory. The expressions and the corresponding files are shown below.

- $\tilde{C_{\mathbf{G}i}}(\mathbf{k})$ — `dat.wan`
- N_s, N_b — `dat.ns-nb`
- $U^{\mathbf{k}} U^{\mathbf{k},opt}$ — `dat.umat`
- $< w_{i0}|r|w_{i0} >$ — `dat.wan-center`

4.2 References

- K. Nakamura, Y. Yoshimoto, Y. Nomura, T. Tadano, M. Kawamura, T. Kosugi, K. Yoshimi, T. Misawa, and Y. Motoyama, Comput. Phys. Commun. 261, 107781 (2021)
- N. Marzari, A. A. Mostofi, J. R. Yates, I. Souza, and D. Vanderbilt, Rev. Mod. Phys. 84, 1419 (2012)

CHAPTER

FIVE

ACKNOWLEDGEMENT

We would like to thank TOYOTA MORTOR CORPORATION for their helpful comments in developing this tool.