H-wave Documentation

Release 1.0.1

ISSP, University of Tokyo

Jan 31, 2025

CONTENTS

CHAPTER

ONE

WHAT IS H-WAVE?

1.1 Introduction

H-wave is a program to perform unrestricted Hartree-Fock (UHF) approximation and random phase approximation (RPA) for itinerant electron systems. It is based on mean field approximation, and it enables calculations of complicated Hamiltonian and large lattices at low compuational cost.

1.2 License

The distribution of the program package and the source codes for H-wave follow GNU General Public License version 3 (GPL v3) or later.

1.3 Contributors

This software was developed by the following contributors.

- ver.1.0.0 (released on 2023/04/25)
  - Developers
    - Kazuyoshi Yoshimi (The Institute for Solid State Physics, The University of Tokyo)
    - Tatsumi Aoyama (The Institute for Solid State Physics, The University of Tokyo)
    - Yuichi Motoyama (The Institute for Solid State Physics, The University of Tokyo)
    - Takahiro Misawa (The Institute for Solid State Physics, The University of Tokyo)
    - Kota Ido (The Institute for Solid State Physics, The University of Tokyo)
    - Akito Kobayashi (Department of Physics, Nagoya University)
    - Taiki Kawamura (Department of Physics, Nagoya University)
  - Project coordinator
    - Takeo Kato (The Institute for Solid State Physics, The University of Tokyo)

1.4 Copyright

This software was developed with the support of “Project for advancement of software usability in materials science” of The Institute for Solid State Physics, The University of Tokyo.

1.5 Operating environment

H-wave was tested on the following platforms

- macOS + python3 (brew)
- Ubuntu Linux + python3 (miniconda)

1.6 Reference

“H-wave – A Python package for the Hartree-Fock approximation and the random phase approximation”, Tatsumi Aoyama, Kazuyoshi Yoshimi, Kota Ido, Yuichi Motoyama, Taiki Kawamura, Takahiro Misawa, Takeo Kato, and Akito Kobayashi, Computer Physics Communications, 298, 109087 (2024) (arXiv:2308.00324 [cond-mat.str-el]).

CHAPTER TWO

BASIC USAGE

- Prerequisite

  H-wave requires the following programs and libraries:

  - python 3.x
  - numpy module
  - scipy module
  - requests module
  - tomli module

  Note that numpy.fft is used for FFT calculations in H-wave UHFk and rpa modes.

- Official Page

  - GitHub repository
  - Sample/Tutorial

- Installation

  - From PyPI:

    H-wave is available from PyPI package repository as follows:

    ```
    $ pip install hwave
    ```

  - From source:

    H-wave source archive can be obtained from the release site:

    https://github.com/issp-center-dev/H-wave/releases

    The latest version is available from the development site using git:

    ```
    $ git clone https://github.com/issp-center-dev/H-wave.git
    ```

    Once the source files are obtained, you can install H-wave by running the following command. The required libraries will also be installed at the same time.

    ```
    $ cd ./H-wave
    $ pip install .
    ```

- Directory structure

```
.
|-- LICENSE
|-- README.md
|-- pyproject.toml
|-- docs/
|   |-- en/
|   |-- ja/
|   |-- tutorial/
|
|-- src/
|   |-- qlms.py
|   |-- hwave/
|       |-- __init__.py
|       |-- qlms.py
|       |-- qlmsio/
|       |   |-- __init__.py
|       |   |-- read_input.py
|       |   |-- read_input_k.py
|       |   |-- wan90.py
|       |-- solver/
|           |-- __init__.py
|           |-- base.py
|           |-- uhfr.py
|           |-- uhfk.py
|           |-- rpa.py
|           |-- perf.py
|-- tests/
```

- Basic usage

  1. Prepare input files

     First, you need to create input files for H-wave that are an input file that specify calculation conditions, and the definition files for the Hamiltonian. To generate the definition files, it will be convenient to use StdFace library. A brief description of these files is given in Tutorial section. You may consult File format sections for the details.

  2. Run

     Run the H-wave program by typing the following command in the directory where the input files are placed, and the calculation will be launched.

     ```
     $ hwave input.toml
     ```

     or

     ```
     $ python3 path_to_H-wave/qlms.py input.toml
     ```

     When the calculation is completed, the results will be written in the output directory. See File format sections for the details of the output files.

CHAPTER

THREE

COORDINATE-SPACE UHF (UHFR)

3.1 Tutorial

To use H-wave, you need to prepare the input files:

1. parameter file to set calculation conditions,
2. definition files of the Hamiltonian,
3. specifications to output the results,

before performing calculations. In the following, we will provide a tutorial using a sample in `docs/tutorial/Hubbard/UHFr` directory. The interaction definition files can be generated using StdFace library. See *Generation of interaction files using StdFace library* section for the details.

3.1.1 Create a parameter file

The parameter file contains information to control inputs and outputs of the program. An example is given in the directory `docs/tutorial/Hubbard/UHFr` by the name `input.toml`. The content of the file will be as follows:

```
[log]
print_level = 1
print_step = 20
[mode]
mode = "UHFr"
[mode.param]
Nsite = 8
2Sz = 0
Ncond = 8
IterationMax = 1000
EPS = 8
RndSeed = 123456789
T = 0.0
[file]
[file.input]
path_to_input = ""
OneBodyG = "greenone.def"
[file.input.interaction]
Trans = "trans.def"
CoulombIntra = "coulombintra.def"
[file.output]
path_to_output = "output"
```

(continues on next page)

(continued from previous page)

```
energy = "energy.dat"
eigen = "eigen"
green = "green.dat"
```

The parameter file is described in TOML format.

In the `log` section, `print_level` specifies the level of the standard output, and `print_step` specifies the number of steps between printing logs.

In the `mode` section, the calculation mode and the basic parameters are specified.

In the `file.input` section, `path_to_input` specifies the directory in which input files are located, `OneBodyG` specifies the definition file of the one-body Green's function to output, and `Initial` specifies the initial configuration. If `OneBodyG` is missing, the Green's function will not be exported. When `Initial` is not specified, a random configuration will be generated for the initial state.

In the `file.input.interaction` section, the input files to define Hamiltonian are specified.

In the `file.output` section, `path_to_output` specifies the directory to which the results will be written. `energy` specifies the filename for the energy, `eigen` specifies the filename for the eigenvalues and eigenvectors of the Hamiltonian, and `green` specifies the filename for the one-body Green's function. If these keywords are missing, the corresponding results will not be exported.

See *Parameter file* section for the details.

3.1.2 Create definition files for Hamiltonian

Next, we will create input files that define the Hamiltonian.

Transfer term

The input file associated with a keyword `Trans` (`trans.def` in this tutorial) provides definitions of Hamltonian for Transfer term of the electron system:

$$\mathcal{H} = -\sum_{ij\sigma_1\sigma_2} t_{ij\sigma_1\sigma_2} c^\dagger_{i\sigma_1} c_{j\sigma_2}. \tag{3.1}$$

The content of the file is as follows:

```
========================
NTransfer      64
========================
========i_j_s_tijs======
========================
    4     0     0     0       1.000000000000000       0.000000000000000
    0     0     4     0       1.000000000000000      -0.000000000000000
    4     1     0     1       1.000000000000000       0.000000000000000
    0     1     4     1       1.000000000000000      -0.000000000000000
    2     0     0     0       1.000000000000000       0.000000000000000
    0     0     2     0       1.000000000000000      -0.000000000000000
    2     1     0     1       1.000000000000000       0.000000000000000
    0     1     2     1       1.000000000000000      -0.000000000000000
...
```

See *Trans file* for the details.

Two-body interaction term

In this tutorial, we consider a two-body interaction Hamiltonian of the electron system of the form:

$$\mathcal{H} = \sum_i U_i n_{i\uparrow} n_{i\downarrow}. \tag{3.2}$$

The definition is given in the file associated with the keyword `CoulombIntra` (`coulombintra.def` in the present case). The content of the file is as follows:

```
=============================================
NCoulombIntra          8
=============================================
================== CoulombIntra ================
=============================================
    0         8.000000000000000
    1         8.000000000000000
    2         8.000000000000000
    3         8.000000000000000
    4         8.000000000000000
 ...
```

There are a number of keywords provided to concicely describe the Hamiltonian, besides `CoulombIntra`. See sections *InterAll file* - *PairLift file* for the details.

3.1.3 Specify output components

Next, we will provide the files that describe the output components.

Setting indices of one-body Green's functions

A file associated with the keyword `OneBodyG` (`greenone.def` in this tutorial) specifies the indices of one-body Green's functions to be calculated $\langle c^\dagger_{i\sigma_1} c_{j\sigma_2} \rangle$. The content of the file will be as follows:

```
===============================
NCisAjs         16
===============================
======== Green functions ======
===============================
    0     0     0     0
    0     0     1     0
    0     0     2     0
    0     0     3     0
    0     0     4     0
 ...
```

See *OneBodyG file* for the details of the file format to specify indices of the one-body Green's functions.

3.1.4 Run

All the input files have been created, and we are ready to run the program. Type in the command with the parameter file (`input.toml` in this tutorial) as an argument:

```
$ hwave input.toml
```

The calculation is launched, and the logs will be shown as follows:

```
2022-12-01 09:37:30,114 INFO qlms: Read def files
2022-12-01 09:37:30,116 INFO qlms: Get Hamiltonian information
2022-12-01 09:37:30,116 INFO qlms: Get Green function information
2022-12-01 09:37:30,116 INFO qlms.uhfr: Show input parameters
  Nsite               : 8
  Ncond               : 8
  2Sz                 : 0
  Mix                 : 0.5
  EPS                 : 1e-08
  IterationMax        : 1000
  RndSeed             : 123456789
  T                   : 0.0
  ene_cutoff          : 100.0
  threshold           : 1e-12
2022-12-01 09:37:30,117 INFO qlms: Start UHF calculation
2022-12-01 09:37:30,117 INFO qlms.uhfr: Set Initial Green's functions
2022-12-01 09:37:30,117 INFO qlms.uhfr: Initialize green function by random numbers
2022-12-01 09:37:30,117 INFO qlms.uhfr: Start UHFr calculations
2022-12-01 09:37:30,117 INFO qlms.uhfr: step, rest, energy, NCond, Sz
2022-12-01 09:37:30,119 INFO qlms.uhfr: 0, 0.022144468, -27.16081+0j, 8, -7.425e-16
2022-12-01 09:37:30,134 INFO qlms.uhfr: 20, 1.2083848e-05, -3.399532+0j, 8, -1.055e-15
2022-12-01 09:37:30,145 INFO qlms.uhfr: UHFr calculation is succeeded: rest=5.
↪7552848630056134e-09, eps=1e-08.
2022-12-01 09:37:30,145 INFO qlms: Save calculation results.
2022-12-01 09:37:30,146 INFO qlms: All procedures are finished.
--------------------------------------------------------------------------------
Statistics
  function                         :  total elapsed  : average elapsed : ncalls
--------------------------------------------------------------------------------
  hwave.solver.uhfr.__init__       :      0.357 msec :      0.357 msec :      1
  hwave.solver.uhfr._initial_G     :      0.090 msec :      0.090 msec :      1
  hwave.solver.uhfr._makeham_const :      0.839 msec :      0.839 msec :      1
  hwave.solver.uhfr._makeham_mat   :      0.309 msec :      0.309 msec :      1
  hwave.solver.uhfr._makeham       :      6.001 msec :      0.176 msec :     34
  hwave.solver.uhfr._diag          :      2.468 msec :      0.073 msec :     34
  hwave.solver.uhfr._green         :      3.107 msec :      0.091 msec :     34
  hwave.solver.uhfr._calc_energy   :      1.990 msec :      0.059 msec :     34
  hwave.solver.uhfr._calc_phys     :     12.929 msec :      0.380 msec :     34
  hwave.solver.uhfr.solve          :     28.290 msec :     28.290 msec :      1
  hwave.solver.uhfr.save_results   :      0.852 msec :      0.852 msec :      1
--------------------------------------------------------------------------------
```

The log messages on reading the input files are presented, followed by the information on the process of UHF calculations. The results are written in the `output` directory, according to the settings in `file.output` section of the input toml file: `energy.dat` for the eigenvalues, `spin-up_eigen.npz` and `spin-down_eigen.npz` for the eigenvectors, and `green.dat` for the one-body Green's functions. See *Output files for UHFr* section for the details of the output files.

3.2 File specifications

3.2.1 Parameter file

The parameter file specifies calculation conditions of H-wave in TOML format. This file consists of these three sections:

1. `mode` section to set calculation mode,
2. `log` section to set standard log output,
3. `file` section to set file and directory paths. This section consists of two subsections, `input` and `output`.

A sample file reads as follows:

```
[log]
print_level = 1
print_step = 20
[mode]
mode = "UHFr"
[mode.param]
Nsite = 8
2Sz = 0
Ncond = 8
IterationMax = 1000
EPS = 8
RndSeed = 123456789
T = 0.0
[file]
[file.input]
path_to_input = ""
OneBodyG = "greenone.def"
[file.input.interaction]
Trans = "trans.def"
CoulombIntra = "coulombintra.def"
[file.output]
path_to_output = "output"
energy = "energy.dat"
eigen = "eigen"
green = "green.dat"
```

File format

TOML

Parameters

`mode` section

- `mode`

  Type : String

  Description : This parameter specifies calculation mode. Set to `UHFr` when using coordinate-space UHF.

- `flag_fock`

  Type : Boolean (default value is `true`)

  Description : When this parameter is `true`, the Fock term is considered. If it is `false`, only the Hartree term is considered.

`mode.param` section

`mode.param` section sets parameters of the calculation.

- `T`

  Type : Float (default value is `0`)

  Description : This parameter specifies temperature. It must be greater than or equal to zero.

- `2Sz`

  Type : Integer, String, or None (default value is None)

  Description : Twice the size of the z compoonent of the total spin is specified when it is set to a fixed value. In this case, the up and down spin components are calculated separately. If this parameter is not given, or it is set to `"free"`, the spin space is not separated in the calculation.

  This parameter should not be specified when `Sz` is not conserved (e.g. when the spin-orbital interaction is present).

  `2Sz` takes a value between `-Nsite` and `Nsite`.

- `Nsite`

  Type : Integer

  Description : This parameter specifies the number of sites. It must be greater than or equal to one.

- `Ncond`

  Type : Integer

  Description : This parameter specifies the number of conduction electrons. It must be greater than or equal to one.

- `filling`

  Type : Float

  Description : This parameter specifies the filling ratio of electrons with respect to the number of states. It must be between 0 and 1. Both `Ncond` and `filling` are specified, the program will be terminated with error.

- `Ncond_round_mode`

  Type : String (default value is `"strict"`)

Description : This parameter specifies how the number of electrons calculated from the `filling` parameter is rounded to an integer value. The parameter must take one of the following values.

- `as-is`: the value is not rounded to an integer. (returns a floating-point number)
- `round-up`: the value is rounded up.
- `round-down`: the value is rounded down.
- `round-off`: the value is rounded to the closest integer. (0.5 is rounded up.)
- `round`: the value is rounded by `round` function. (0.5 is rounded down.)
- `strict`: if the value is not an integer value, the program terminates with error.
- `exact`: if the value is not an integer value, a warning message will be shown and the value is rounded to an integer as `round`.

- `IterationMax`

  Type : Integer (default value is `20000`)

  Description : This parameter specifies the maximum number of iterations. It must be greater than or equal to zero.

- `EPS`

  Type : Integer (default value is `6`)

  Description : This parameter specifies the convergence criterion. The solver iteration will be terminated when the norm of the difference between the previous and new Green's function falls below $10^{-\mathrm{EPS}}$. The residue is defined by $R = \sum_{i,j}^{N} \sqrt{\left|G_{ij}^{\mathrm{new}} - G_{ij}^{\mathrm{old}}\right|^2}/2N^2$. It must be greater than or equal to zero.

- `Mix`

  Type : Float (default value is `0.5`)

  Description : This parameter specifies the ratio α of simple-mixing when the Green's function is updated by the previous and the new one. It must be between 0 and 1. If it is set to 1, the previous value will not be mixed. See *Algorithms* section for simple-mixing algorithm.

- `RndSeed`

  Type : Integer (default value is `1234`)

  Description : This parameter specifies the seed of random numbers.

- `ene_cutoff`

  Type : Float (default value is `100.0`)

  Description : This parameter specifies a cut-off to avoid overflow when the Fermi distribution function is calculated.

- `strict_hermite`

  Type : Boolean (default value is `false`)

  Description : This parameter specifies strictness of Hermiticity checks when the interaction definitions are read from files. If it is set to `true`, the program stops when the deviation larger than `hermite_tolerance` is detected. If it is set to `false`, a warning message will be shown and the program execution continues.

- `hermite_tolerance`

  Type : Float (default value is 10^{-8})

  Description : This parameter specifies the tolerance of the deviation from Hermiticity condition $|t_{ij} - t_{ji}^*| < \varepsilon$.

`log` section

- `print_level`

  Type : Integer (default value is 1)

  Description : This parameter specifies verbosity of the standard log output. When it is set to 1, the detailed information will be printed.

- `print_step`

  Type : Integer (default value is 1)

  Description : This parameter specifies the interval between outputs of calculation logs to the standard output during iterations. It must be greater than or equal to one.

- `print_check`

  Type : String

  Description : This parameter specifies the output logfile to which the calculation logs are written during the iterations besides the standard output. If it is not given, the logs are not exported to files.

`file` section

This section consists of `input` and `output` subsections. The former specifies settings on input files (e.g. locations and names of files), while the latter on output files, as described below.

`file.input` section

- `path_to_input`

  Type : String (default value is `""`)

  Description : This parameter specifies the directory in which the input files are located.

- `Initial`

  Type : String (default value is `""`)

  Description : This parameter specifies the input file for the initial configuration.

- `OneBodyG`

  Type : String (default value is `""`)

  Description : This parameter specifies the input file that contains a list of indices of one-body Green's function to export.

`file.input.interaction` section

- `Trans`

  Type : String (default value is `""`)

  Description : This parameter specifies the input file for general one-body interaction term.

- `InterAll`

  Type : String (default value is `""`)

  Description : This parameter specifies the input file for generalized two-body interaction term.

- `CoulombIntra`

  Type : String (default value is `""`)

  Description : This parameter specifies the input file for on-site Coulomb interaction term.

- `CoulombInter`

  Type : String (default value is `""`)

  Description : This parameter specifies the input file for inter-site Coulomb interaction term.

- `Hund`

  Type : String (default value is `""`)

  Description : This parameter specifies the input file for Hund interaction term.

- `PairHop`

  Type : String (default value is `""`)

  Description : This parameter specifies the input file for pair-hopping term.

- `Exchange`

  Type : String (default value is `""`)

  Description : This parameter specifies the input file for exchange interaction term.

- `Ising`

  Type : String (default value is `""`)

  Description : This parameter specifies the input file for Ising interaction term.

- `PairLift`

  Type : String (default value is `""`)

  Description : This parameter specifies the input file for pair-lift interaction term.

`file.output` section

- `path_to_output`

  Type : String (default value is `"output"`)

  Description : This parameter specifies the directory to store the output files.

- `energy`

  Type : String

  Description : This parameter specifies the output file for energies. If it is not given, the output is not exported.

- `eigen`

  Type : String

  Description : This parameter specifies the output file for eigenvalues of Hamiltonian. If it is not given, the output is not exported.

- `green`

  Type : String

  Description : This parameter specifies the output file for one-body Green's function. If it is not given, the output is not exported.

- `initial`

  Type : String

  Description : This parameter specifies the output file for one-body Green's function in a format suitable for the initial configuration. If it is not given, the output is not exported.

- `fij`

  Type : String

  Description : This parameter specifies the output file for pair-orbital factor f_{ij}. If it is not given, the output is not exported.

3.2.2 Input files for UHFr

In this section, the input files (*.def) used in H-wave are described. They are divided into two types.

(1) Hamiltonian

The Hamiltonian is given in the form of interactions of the electron system. They are defined in these files.

Trans gives the one-body part expressed by $c^\dagger_{i\sigma_1} c_{j\sigma_2}$.
InterAll gives the generalized two-body interactions expressed by $c^\dagger_{i\sigma_1} c_{j\sigma_2} c^\dagger_{k\sigma_3} c_{l\sigma_4}$.

Besides, we can specify by the following keywords the interactions that are frequently used.

CoulombIntra gives the on-site Coulomb interaction expressed by $n_{i\uparrow} n_{i\downarrow}$, where $n_{i\sigma} = c^\dagger_{i\sigma} c_{i\sigma}$.
CoulombInter gives the inter-site Coulomb interaction expressed by $n_i n_j$, where $n_i = n_{i\uparrow} + n_{i\downarrow}$.
Hund gives the Hund interaction expressed by $n_{i\uparrow} n_{j\uparrow} + n_{i\downarrow} n_{j\downarrow}$.
PairHop gives the pair-hop interaction expressed by $c^\dagger_{i\uparrow} c_{j\uparrow} c^\dagger_{i\downarrow} c_{j\downarrow}$.
Exchange gives the exchange interaction expressed by $S^+_i S^-_j$.
Ising gives the Ising interaction expressed by $S^z_i S^z_j$.
PairLift gives the pair-lift interaction expressed by $c^\dagger_{i\uparrow} c_{i\downarrow} c^\dagger_{j\uparrow} c_{j\downarrow}$.

(2) Green's functions

Initial specifies the one-body Green's function for the initial configuration, $\langle c^\dagger_{i\sigma_1} c_{j\sigma_2} \rangle$.
OneBodyG specifies the indices of the one-body Green's function $\langle c^\dagger_{i\sigma_1} c_{j\sigma_2} \rangle$ to be exported.

The details of the file format for each input file is described in the following subsections.

Trans file

This file determines the coefficients of the transfer integrals $t_{ij\sigma_1\sigma_2}$ in the Hamiltonian

$$\mathcal{H} = -\sum_{ij\sigma_1\sigma_2} t_{ij\sigma_1\sigma_2} c^\dagger_{i\sigma_1} c_{j\sigma_2}. \tag{3.3}$$

An example of the file format is presented below.

```
========================
NTransfer      24
========================
========i_j_s_tijs======
========================
    0     0     2     0   1.000000  0.000000
    2     0     0     0   1.000000  0.000000
    0     1     2     1   1.000000  0.000000
    2     1     0     1   1.000000  0.000000
    2     0     4     0   1.000000  0.000000
    4     0     2     0   1.000000  0.000000
    2     1     4     1   1.000000  0.000000
    4     1     2     1   1.000000  0.000000
    4     0     6     0   1.000000  0.000000
    6     0     4     0   1.000000  0.000000
    4     1     6     1   1.000000  0.000000
    6     1     4     1   1.000000  0.000000
    6     0     8     0   1.000000  0.000000
    8     0     6     0   1.000000  0.000000
...
```

File format

- Line 1: Header
- Line 2: `[ntransfer] [count]`
- Lines 3-5: Header
- Lines 6-: `[i] [s1] [j] [s2] [v.real] [v.imag]`

Parameters

- `[ntransfer]`

  Type : String (blank is not allowed)

  Description : An arbitrary keyword for the total number of transfer integrals.

- `[count]`

  Type : Integer (blank is not allowed)

  Description : An integer giving the total number of transfer integrals.

- `[i]`, `[j]`

  Type : Integer (blank is not allowed)

  Description : An integer giving a site index ($0 \leq i, j < \mathrm{Nsite}$).

- `[s1]`, `[s2]`

  Type : Integer (blank is not allowed)

  Description : An integer giving a spin index: either 0 (up-spin) or 1 (down-spin).

- `[v.real]`, `[v.imag]`

  Type : Float (blank is not allowed)

  Description : Values for the real and imaginary parts of $t_{ij\sigma_1\sigma_2}$, respectively.

Usage rules

- Headers cannot be omitted.
- Since the Hamiltonian should be hermite, the coefficients must satisfy the relation $t_{ij\sigma_1\sigma_2} = t^{\dagger}_{ji\sigma_2\sigma_1}$. Otherwise, the program is terminated or a warning is reported, depending on the `strict_hermite` parameter.
- The program is terminated with error if there are duplicated entries.
- The program is terminated with error when the number of entries is different from `[count]`.
- The program is terminated with error if `[i]`, `[j]`, `[s1]`, or `[s2]` are outside the range of the defined values.

InterAll file

This file determines the coefficients of the generalized two-body interaction integrals $I_{ijkl\sigma_1\sigma_2\sigma_3\sigma_4}$ of the Hamiltonian

$$\mathcal{H} = \sum_{i,j,k,l} \sum_{\sigma_1,\sigma_2,\sigma_3,\sigma_4} I_{ijkl\sigma_1\sigma_2\sigma_3\sigma_4} c^\dagger_{i\sigma_1} c_{j\sigma_2} c^\dagger_{k\sigma_3} c_{l\sigma_4}. \tag{3.4}$$

An example of the file format is presented below.

```
======================
NInterAll      36
======================
========zInterAll=====
======================
0    0    0    1    1    1    1    0    0.50   0.0
0    1    0    0    1    0    1    1    0.50   0.0
0    0    0    0    1    0    1    0    0.25   0.0
0    0    0    0    1    1    1    1   -0.25   0.0
0    1    0    1    1    0    1    0   -0.25   0.0
0    1    0    1    1    1    1    1    0.25   0.0
2    0    2    1    3    1    3    0    0.50   0.0
2    1    2    0    3    0    3    1    0.50   0.0
2    0    2    0    3    0    3    0    0.25   0.0
2    0    2    0    3    1    3    1   -0.25   0.0
2    1    2    1    3    0    3    0   -0.25   0.0
2    1    2    1    3    1    3    1    0.25   0.0
4    0    4    1    5    1    5    0    0.50   0.0
4    1    4    0    5    0    5    1    0.50   0.0
4    0    4    0    5    0    5    0    0.25   0.0
4    0    4    0    5    1    5    1   -0.25   0.0
4    1    4    1    5    0    5    0   -0.25   0.0
4    1    4    1    5    1    5    1    0.25   0.0
...
```

File format

- Line 1: Header
- Line 2: `[ninterall] [count]`
- Lines 3-5: Header
- Lines 6-: `[i] [s1] [j] [s2] [k] [s3] [l] [s4] [v.real] [v.imag]`

Parameters

- `[ninterall]`

  Type : String (blank is not allowed)

  Description : An arbitrary keyword for the total number of the two-body interactions.

- `[count]`

  Type : Integer (blank is not allowed)

  Description : An integer giving the total number of the two-body interactions.

- [i], [j], [k], [l]

  Type : Integer (blank is not allowed)

  Description : An integer giving a site index ($0 \leq i, j, k, l < \mathrm{Nsite}$).

- [s1], [s2], [s3], [s4]

  Type : Integer (blank is not allowed)

  Description : An integer giving a spin index: either 0 (up-spin) or 1 (down-spin).

- [v.real], [v.imag]

  Type : Float (blank is not allowed)

  Description : Values for the real and imaginary parts of $I_{ijkl\sigma_1\sigma_2\sigma_3\sigma_4}$, respectively.

Usage ruels

- Headers cannot be omitted.
- Since the Hamiltonian should be hermite, the coefficients must satisfy the relation $I_{ijkl\sigma_1\sigma_2\sigma_3\sigma_4} = I^{\dagger}_{lkji\sigma_4\sigma_3\sigma_2\sigma_1}$. Otherwise, the program is terminated or a warning is reported, depending on the strict_hermite parameter. It is noted that the term of the Hermite conjugate for $I_{ijkl\sigma_1\sigma_2\sigma_3\sigma_4} c^{\dagger}_{i\sigma_1} c_{j\sigma_2} c^{\dagger}_{k\sigma_3} c_{l\sigma_4}$ should be read $I_{lkji\sigma_4\sigma_3\sigma_2\sigma_1} c^{\dagger}_{l\sigma_4} c_{k\sigma_3} c^{\dagger}_{j\sigma_2} c_{i\sigma_1}$.
- The program is terminated with error if there are duplicated entries.
- The program is terminated with error when the number of entries is different from [count].
- The program is terminated with error if [i], [j], [k], [l], [s1], [s2], [s3], or [s4] are outside the range of the defined values.

CoulombIntra file

This file determines the coefficients of the on-site Coulomb interactions given by

$$\mathcal{H} = \sum_i U_i n_{i\uparrow} n_{i\downarrow}. \tag{3.5}$$

An example of the file format is presented below.

```
======================
NCoulombIntra 6
======================
==== CoulombIntra ====
======================
   0  4.000000
   1  4.000000
   2  4.000000
   3  4.000000
   4  4.000000
   5  4.000000
```

File format

- Line 1: Header
- Line 2: `[ncoulombintra] [count]`
- Lines 3-5: Header
- Lines 6-: `[i] [val]`

Parameters

- `[ncoulombintra]`

  Type : String (blank is not allowed)

  Description : An arbitrary keyword for the total number of the on-site Coulomb interactions.

- `[count]`

  Type : Integer (blank is not allowed)

  Description : An integer giving the total number of the on-site Coulomb interactions.

- `[i]`

  Type : Integer (blank is not allowed)

  Description : An integer giving a site index ($0 \leq i <$ Nsite).

- `[val]`

  Type : Float (blank is not allowed)

  Description : A value for U_i.

Usage rules

- Headers cannot be omitted.
- The program is terminated with error if there are duplicated entries.
- The program is terminated with error when the number of entries is different from `[count]`.
- The program is terminated with error if `[i]` is outside the range of the defined value.

CoulombInter file

This file determines the coefficients of the off-site Coulomb interactions given by

$$\mathcal{H} = \sum_{i,j} V_{ij} n_i n_j. \tag{3.6}$$

An example of the file format is presented below.

```
======================
NCoulombInter 6
======================
========CoulombInter ======
======================
   0     1 -0.125000
   1     2 -0.125000
   2     3 -0.125000
   3     4 -0.125000
   4     5 -0.125000
   5     0 -0.125000
```

File format

- Line 1: Header
- Line 2: `[ncoulombinter] [count]`
- Lines 3-5: Header
- Lines 6-: `[i] [j] [val]`

Parameters

- `[ncoulombinter]`

  Type : String (blank is not allowed)

  Description : An arbitrary keyword for the total number of the off-site Coulomb interactions.

- `[count]`

  Type : Integer (blank is not allowed)

  Description : An integer giving the total number of the off-site Coulomb interactions.

- `[i]`, `[j]`

  Type : Integer (blank is not allowed)

  Description : An integer giving a site index ($0 \leq i, j <$ Nsite).

- `[val]`

  Type : Float (blank is not allowed)

  Description : A value for V_{ij}.

Usage rules

- Headers cannot be omitted.
- The program is terminated with error if there are duplicated entries.
- The program is terminated with error when the number of entries is different from `[count]`.
- The program is terminated with error if `[i]` or `[j]` are outside the range of the defined values.

Hund file

This file determines the coefficients of the Hund couplings given by

$$\mathcal{H} = -\sum_{i,j} J_{ij}^{\mathrm{Hund}} (n_{i\uparrow} n_{j\uparrow} + n_{i\downarrow} n_{j\downarrow}). \tag{3.7}$$

An example of the file format is presented below.

```
======================
NHund 6
======================
========Hund ======
======================
   0     1 -0.250000
   1     2 -0.250000
   2     3 -0.250000
   3     4 -0.250000
   4     5 -0.250000
   5     0 -0.250000
```

File format

- Line 1: Header
- Line 2: `[nhund] [count]`
- Lines 3-5: Header
- Lines 6-: `[i] [j] [val]`

Parameters

- `[nhund]`

  Type : String (blank is not allowed)

  Description : An arbitrary keyword for the total number of the Hund couplings.

- `[count]`

  Type : Integer (blank is not allowed)

  Description : An integer giving the total number of the Hund couplings.

- `[i]`, `[j]`

  Type : Integer (blank is not allowed)

  Description : An integer giving a site index ($0 \le i, j < \mathrm{Nsite}$).

- `[val]`

  Type : Float (blank is not allowed)

  Description : A value for J_{ij}^{Hund}.

Usage rules

- Headers cannot be omitted.
- The program is terminated with error if there are duplicated entries.
- The program is terminated with error when the number of entries is different from `[count]`.
- The program is terminated with error if `[i]` or `[j]` are outside the range of the defined values.

PairHop file

This file determines the coefficients of the pair-hopping couplings given by

$$\mathcal{H} = \sum_{i,j} J_{ij}^{\mathrm{Pair}} (c_{i\uparrow}^{\dagger} c_{j\uparrow} c_{i\downarrow}^{\dagger} c_{j\downarrow} + h.c.). \tag{3.8}$$

An example of the file format is presented below.

```
======================
NPairhop 6
======================
========Pairhop ======
======================
   0     1  0.50000
   1     2  0.50000
   2     3  0.50000
   3     4  0.50000
   4     5  0.50000
   5     0  0.50000
```

File format

- Line 1: Header
- Line 2: `[npairhop] [count]`
- Lines 3-5: Header
- Lines 6-: `[i] [j] [val]`

Parameters

- `[npairhop]`

  Type : String (blank is not allowed)

  Description : An arbitrary keyword for the total number of the pair-hopping couplings.

- `[count]`

  Type : Integer (blank is not allowed)

  Description : An integer giving the total number of the pair-hopping couplings.

- `[i]`, `[j]`

  Type : Integer (blank is not allowed)

  Description : An integer giving a site index ($0 \leq i, j < \mathrm{Nsite}$).

- `[val]`

  Type : Float (blank is not allowed)

  Description : A value for J_{ij}^{Pair}.

Usage rules

- Headers cannot be omitted.
- The program is terminated with error if there are duplicated entries.
- The program is terminated with error when the number of entries is different from `[count]`.
- The program is terminated with error if `[i]` or `[j]` are outside the range of the defined values.

Exchange file

This file determines the coefficients of the exchange couplings given by

$$\mathcal{H} = \sum_{i,j} J_{ij}^{\mathrm{Ex}} (c_{i\uparrow}^{\dagger} c_{j\uparrow} c_{j\downarrow}^{\dagger} c_{i\downarrow} + c_{i\downarrow}^{\dagger} c_{j\downarrow} c_{j\uparrow}^{\dagger} c_{i\uparrow}). \tag{3.9}$$

An example of the file format is presented below.

```
======================
NExchange 6
======================
========Exchange ======
======================
   0     1  0.50000
   1     2  0.50000
   2     3  0.50000
   3     4  0.50000
   4     5  0.50000
   5     0  0.50000
```

File format

- Line 1: Header
- Line 2: `[nexchange] [count]`
- Lines 3-5: Header
- Lines 6-: `[i] [j] [val]`

Parameters

- `[nexchange]`

  Type : String (blank is not allowed)

  Description : An arbitrary keyword for the total number of the exchange couplings.

- `[count]`

  Type : Integer (blank is not allowed)

  Description : An integer giving the total number of the exchange couplings.

- `[i]`, `[j]`

  Type : Integer (blank is not allowed)

  Description : An integer giving a site index ($0 \leq i, j < \mathrm{Nsite}$).

- `[val]`

  Type : Float (blank is not allowed)

  Description : A value for J_{ij}^{Ex}.

Usage rules

- Headers cannot be omitted.
- The program is terminated with error if there are duplicated entries.
- The program is terminated with error when the number of entries is different from `[count]`.
- The program is terminated with error if `[i]` or `[j]` are outside the range of the defined values.

Ising file

This file determines the coefficients of the Ising interactions given by

$$\mathcal{H} + = \sum_{i,j} J^z_{ij} (n_{i\uparrow} - n_{i\downarrow})(n_{j\uparrow} - n_{j\downarrow}) \tag{3.10}$$

An example of the file format is presented below.

```
======================
NIsing 6
======================
========Ising ======
======================
   0      1  0.50000
   1      2  0.50000
   2      3  0.50000
   3      4  0.50000
   4      5  0.50000
   5      0  0.50000
```

File format

- Line 1: Header
- Lines 2: `[nising] [count]`
- Lines 3-5: Header
- Lines 6-: `[i] [j] [val]`

Parameters

- `[nising]`

  Type : String (blank is not allowed)

  Description : An arbitrary keyword for the total number of the Ising interactions.

- `[count]`

  Type : Integer (blank is not allowed)

  Description : An integer giving the total number of the Ising interactions.

- `[i]`, `[j]`

  Type : Integer (blank is not allowed)

  Description : An integer giving a site index ($0 \leq i, j <$ Nsite).

- `[val]`

  Type : Float (blank is not allowed)

  Description : A value for J^z_{ij}.

Usage rules

- Headers cannot be omitted.
- The program is terminated with error if there are duplicated entries.
- The program is terminated with error when the number of entries is different from `[count]`.
- The program is terminated with error if `[i]` or `[j]` are outside the range of the defined values.

PairLift file

This file determines the coefficients of the pair-lift couplings given by

$$\mathcal{H} = \sum_{i,j} J_{ij}^{\mathrm{PairLift}} (c_{i\uparrow}^{\dagger} c_{i\downarrow} c_{j\uparrow}^{\dagger} c_{j\downarrow} + c_{i\downarrow}^{\dagger} c_{i\uparrow} c_{j\downarrow}^{\dagger} c_{j\uparrow}). \tag{3.11}$$

An example of the file format is presented below.

```
======================
NPairLift 6
======================
========NPairLift ======
======================
   0     1  0.50000
   1     2  0.50000
   2     3  0.50000
   3     4  0.50000
   4     5  0.50000
   5     0  0.50000
```

File format

- Line 1: Header
- Line 2: `[npairlift] [count]`
- Lines 3-5: Header
- Lines 6-: `[i] [j] [val]`

Parameters

- `[npairlift]`

  Type : String (blank is not allowed)

  Description : An arbitrary keyword for the total number of the pair-lift couplings.

- `[count]`

  Type : Integer (blank is not allowed)

  Description : An integer giving the total number of the pair-lift couplings.

- `[i]`, `[j]`

  Type : Integer (blank is not allowed)

  Description : An integer giving a site index ($0 \leq i, j < \mathrm{Nsite}$).

- `[val]`

  Type : Double (blank is not allowed)

  Description : A value for $J_{ij}^{\mathrm{PairLift}}$.

Usage rules

- Headers cannot be omitted.
- The program is terminated with error if there are duplicated entries.
- The program is terminated with error when the number of entries is different from `[count]`.
- The program is terminated with error if `[i]` or `[j]` are outside the range of the defined values.

Initial file

This file contains the values of Green's function $G_{ij\sigma_1\sigma_2} \equiv \langle c^{\dagger}_{i\sigma_1} c_{j\sigma_2} \rangle$ to be read for an initial configuration. The unspecified elements are assumed to be zero. The file format is the same as that of the `green` output file.

An example of the file format is presented below.

```
0 0 0 0  0.9517526553947047  0.0
0 0 1 0 -0.03971951040016314 0.0
0 0 2 0  0.09202884754223833 0.0
0 0 3 0 -0.039719448981075135 0.0
0 0 4 0  0.09202884754219534 0.0
0 0 5 0 -0.03971947216145664 0.0
0 0 6 0  0.09202884753253462 0.0
0 0 7 0  0.09202884754259735 0.0
0 1 0 1  0.04824734460529617 0.0
0 1 1 1  0.03971951040016307 0.0
...
```

File format

- `[i] [s1] [j] [s2] [v.real] [v.imag]`

Parameters

- `[i]`, `[j]`

  Type : Integer

  Description : An integer giving a site index ($0 \leq i, j <$ Nsite).

- `[s1]`, `[s2]`

  Type : Integer

  Description : An integer giving a spin index: either 0 (up-spin) or 1 (down-spin).

- `[v.real]`, `[v.imag]`

  Type : Float

  Description : Values for the real and imaginary parts of $\langle c^{\dagger}_{i\sigma_1} c_{j\sigma_2} \rangle$, respectively.

OneBodyG file

This file determines the list of indices of the one-body Green's function $\langle c^\dagger_{i\sigma_1} c_{j\sigma_2} \rangle$ to be exported. An example of the file format is presented below.

```
===============================
NCisAjs         24
===============================
======== Green functions ======
===============================
    0     0     0     0
    0     1     0     1
    1     0     1     0
    1     1     1     1
    2     0     2     0
    2     1     2     1
    3     0     3     0
    3     1     3     1
    4     0     4     0
    4     1     4     1
    5     0     5     0
    5     1     5     1
    6     0     6     0
    6     1     6     1
    7     0     7     0
    7     1     7     1
    8     0     8     0
    8     1     8     1
    9     0     9     0
    9     1     9     1
   10     0    10     0
   10     1    10     1
   11     0    11     0
   11     1    11     1
```

File format

- Line 1: Header
- Line 2: `[ncisajs] [count]`
- Lines 3-5: Header
- Lines 6-: `[i] [s1] [j] [s2]`

Parameters

- [ncisajs]

  Type : String (blank is not allowed)

  Description : An arbitrary keyword for the total number of components of the one-body Green's functions.

- [count]

  Type : Integer (blank is not allowed)

  Description : An integer giving the total number of components of the one-body Green's functions.

- [i], [j]

  Type : Integer (blank is not allowed)

  Description : An integer giving a site index ($0 \leq i, j < \mathrm{Nsite}$).

- [s1], [s2]

  Type : Integer (blank is not allowed)

  Description : An integer giving a spin index: either 0 (up-spin) or 1 (down-spin).

Usage rules

- Headers cannot be omitted.
- The program is terminated with error if there are duplicated entries.
- The program is terminated with error when the number of entries is different from [count].
- The program is terminated with error if [i], [j], [s1], or [s2] are outside the range of the defined values.

3.2.3 Output files for UHFr

In the coordinate-space UHF (UHFr) calculation, the energy, the eigenvalues and eigenvectors, and the one-body Green's functions are exported. In this section, the file formats of these outputs are described.

Energy

The values of energy, number of particles, and spin obtained by the calculations of the UHF method are written to this file. The filename is specified by a keyword `energy` in `file.output` section of the input parameter file. An example of the output is presented below.

```
Energy_total = -5.88984624257707
Energy_band = -0.9265413257740396
Energy_interall = -4.963304916803031
NCond = 8.000000000000007
Sz = 3.2822430107160017e-07
```

File format

- Energy_total = `[energy_total]`
- Energy_band = `[energy_band]`
- Energy_interall = `[energy_interall]`
- NCond = `[ncond]`
- Sz = `[sz]`

Parameters

- `[energy_total]`

  Type : Float

  Description : The value of the total energy which is calculated using the eigenvectors obtained by the UHF method.

- `[energy_band]`

  Type : Float

  Description : The value of the energy which is derived from the eigenvalues of the Hamiltonian obtained by the UHF mothod.

- `[energy_interall]`

  Type : Float

  Description : The value of the energy of the interaction terms.

- `[ncond]`

  Type : Float

  Description : The expectation value of the number of particles, $\sum_i \langle n_i \rangle$.

- `[sz]`

  Type : Float

  Description : The expectation value of the z component of the total spin, $S_z = \sum_i \langle (n_{i\uparrow} - n_{i\downarrow}) \rangle / 2$.

eigen

The eigenvalues and eigenvectors of the Hamiltonian obtained by the UHF method are exported in npz (numpy zip archive) format.

The filename is chosen, with the string specified for the `eigen` keyword in `file.output` section (indicated by *eigen_str* hereafter), as `{key}_`*eigen_str*`.npz`, where `{key}` turns to be:

- `sz-free` if the parameter `2Sz` is not specified in `mode.param` section,
- `spin-up` and `spin-down` otherwise. (yields two files)

The code shown below is an example for reading data from the file using Python.

```
import numpy as np
data = np.load("key_eigen_str.npz")
eigenvalue = data["eigenvalue"]
eigenvector = data["eigenvector"]
```

`eigenvalue` holds a list of eigenvalues in ascending order. The number of eigenvalues is `N` if `2Sz` is specified, or `2N` otherwise, for `N` being the total number of sites.

`eigenvector` holds the corresponding eigenvectors as a two-dimensional array: The first index refers to the site index `i_site` and the spin index `s_spin` (0 for up-spin, and 1 for down-spin) by:

- `i_site` if *2Sz* is specified, or
- `i_site + s_spin * N` otherwise.

The second index refers to the index of the eigenvalues.

green

The one-body Green's function $\langle c^\dagger_{i\sigma_1} c_{j\sigma_2} \rangle$ obtained by the UHF calculation is to be exported. The indices of the component to be written are specified by `OneBodyG`. The filename is associated with a keyword `green` in `file.output` section of the input parameter file. An example of the output is presented below.

```
0 0 0 0  0.9517526553947047  0.0
0 0 1 0 -0.03971951040016314 0.0
0 0 2 0  0.09202884754223833 0.0
0 0 3 0 -0.039719448981075135 0.0
0 0 4 0  0.09202884754219534 0.0
0 0 5 0 -0.03971947216145664 0.0
0 0 6 0  0.09202884753253462 0.0
0 0 7 0  0.09202884754259735 0.0
0 1 0 1  0.04824734460529617 0.0
0 1 1 1  0.03971951040016307 0.0
...
```

File format

- `[i] [s1] [j] [s2] [v.real] [v.imag]`

Parameters

- `[i]`, `[j]`

  Type : Integer

  Description : An integer giving a site index ($0 \leq i, j < \mathrm{Nsite}$).

- `[s1]`, `[s2]`

  Type : Integer

  Description : An integer giving a spin index: either 0 (up-spin) or 1 (down-spin).

- `[v.real]`, `[v.imag]`

  Type : Float

  Description : Values for the real and imaginary parts of $\langle c^\dagger_{i\sigma_1} c_{j\sigma_2} \rangle$, respectively.

CHAPTER

FOUR

WAVE-NUMBER SPACE UHF (UHFK)

4.1 Tutorial

To use H-wave in wave-number space (UHFk), you need to prepare input files:

1. an input parameter file,
2. interaction definition files,

before running the program. For the latter, you can use the outputs of external programs such as RESPACK, or you may create the files using the StdFace library.

In the following, we provide a tutorial based on a sample in `docs/tutorial/Hubbard/UHFk` directory. The interaction definition files can be generated using StdFace library. See *Generation of interaction files using StdFace library* section for the details.

4.1.1 Create a parameter file

We prepare an input parameter file that contains basic parameters as well as settings on inputs and outputs. A sample file can be found in `docs/tutorial/Hubbard/UHFk` directory by a filename `input.toml`. The content of the file is as follows:

```
[log]
  print_level = 1
  print_step = 10
[mode]
  mode = "UHFk"
[mode.param]
  # 2Sz = 0
  Ncond = 16
  IterationMax = 1000
  EPS = 8
  Mix = 0.5
  RndSeed = 123456789
  # ene_cutoff = 1.0e+2
  T = 0.0
  CellShape = [ 4, 4, 1 ]
  SubShape = [ 2, 2, 1 ]
[file]
[file.input]
  path_to_input = ""
  # initial = "green_init.dat.npz"
```

(continues on next page)

(continued from previous page)

```
[file.input.interaction]
  path_to_input = "./"
  Geometry = "geom.dat"
  Transfer = "transfer.dat"
  CoulombInter = "coulombinter.dat"
[file.output]
  path_to_output = "output"
  energy = "energy.dat"
  eigen = "eigen"
  green = "green"
```

The file is written in TOML format, and organized by the following sections.

`[log]` section

This section contains settings on log outputs. `print_level` sets the verbosity of the standard output, and `print_step` sets the interval of the outputs.

`[mode]` section

This section contains settings on calculation mode and basic parameters. We choose `mode` to be either the coordinate space UHF (`UHFr`) or the wave-number space UHF (`UHFk`). The calculation parameters are specified in `[mode.param]` subsection.

`[file]` section

`[file.input]` subsection contains settings on the directory for the input files by `path_to_input`, and the filename for the initial configuration by `initial`. If the latter is not specified, a random configuration will be generated.

`[file.input.interaction]` subsection contains a list of files associated with the geometry information and the interactions distinguished by the keywords.

`[file.output]` subsection contains filenames for the physical observables such as the energy by `energy`, for the eigenvalues and eigenvectors of the Hamiltonian by `eigen`, and for the one-body Green's functions by `green`. If they are not specified, the corresponding data will not be outputted.

See *Parameter files* section for the details.

4.1.2 Create interaction definition files

You need to prepare data files on the geometry information of the lattice, and the coefficients of the interactions required to construct the Hamiltonian. The association between the types of information and the filenames is provided in `[file.input.interaction]` section.

Geometry

The file associated with Geometry provides the geometrical information of the lattice. An example of the file is shown below.

```
   1.000000000000   0.000000000000   0.000000000000
   0.000000000000   1.000000000000   0.000000000000
   0.000000000000   0.000000000000   1.000000000000
1
     0.000000000000000e+00     0.000000000000000e+00     0.000000000000000e+00
```

It contains the primitive vectors (lines 1–3), the number of orbitals (line 4), and the Wannier centers of the orbitals (line 5 onwards).

Transfer, CoulombIntra, CoulombInter, Hund, etc

The file associated with Transfer contains the coefficients of Hamiltonian corresponding to the transfer term of the electron systems. The coefficients of the two-body interaction terms are stored in the files associated with the respective keywords. The defined types include CoulombIntra, CoulombInter, Hund, Ising, Exchange, PairLift, and PairHop, in accordance with the coordinate-space UHF. These files are written in Wannier90(-like) format, as exemplified below.

```
Transfer in wannier90-like format for uhfk
1
9
 1 1 1 1 1 1 1 1 1
  -1    0    0    1    1  -1.000000000000  -0.000000000000
   0   -1    0    1    1  -1.000000000000  -0.000000000000
   0    1    0    1    1  -1.000000000000   0.000000000000
   1    0    0    1    1  -1.000000000000   0.000000000000
```

It contains a comment (line 1), the number of orbitals (line 2), the number of cells `nrpts` of the rectangular cuboid that accommodates translation vectors (line 3), the multiplicity factors (`nrpts` elements, with 15 elements per line), and the elements of the coefficient matrix.

Each element of the matrix consists of translation vector r_x, r_y, r_z, indices of orbitals α, β, and the real and imaginary part of the coefficient.

4.1.3 Run

Once you prepare all the input files, you can perform the calculation by running H-wave with the input parameter file (`input.toml` in this tutorial) as an argument.

```
$ hwave input.toml
```

The calculation starts with the logs as shown below:

```
2022-12-02 13:48:11,641 INFO qlms: Read definitions from files
2022-12-02 13:48:11,641 INFO qlms.read_input: QLMSkInput: read Gemoetry from ./geom.dat
2022-12-02 13:48:11,641 INFO qlms.read_input: QLMSkInput: read interaction Transfer from␣
↪./transfer.dat
2022-12-02 13:48:11,641 INFO qlms.read_input: QLMSkInput: read interaction CoulombInter␣
↪from ./coulombinter.dat
2022-12-02 13:48:11,641 INFO qlms: Get Hamiltonian information
```

(continues on next page)

(continued from previous page)

```
2022-12-02 13:48:11,641 INFO qlms: Get Green function information
2022-12-02 13:48:11,667 INFO qlms.uhfk: Show parameters
2022-12-02 13:48:11,668 INFO qlms.uhfk:     Cell Shape     = (4, 4, 1)
2022-12-02 13:48:11,668 INFO qlms.uhfk:     Sub Shape      = (2, 2, 1)
2022-12-02 13:48:11,668 INFO qlms.uhfk:     Block          = (2, 2, 1)
2022-12-02 13:48:11,668 INFO qlms.uhfk:     Block volume   = 4
2022-12-02 13:48:11,668 INFO qlms.uhfk:     Num orbit      = 1
2022-12-02 13:48:11,668 INFO qlms.uhfk:     Num orbit eff  = 4
2022-12-02 13:48:11,668 INFO qlms.uhfk:     nspin          = 2
2022-12-02 13:48:11,668 INFO qlms.uhfk:     nd             = 8
2022-12-02 13:48:11,668 INFO qlms.uhfk:     Ncond          = 16
2022-12-02 13:48:11,669 INFO qlms.uhfk:     T              = 0.0
2022-12-02 13:48:11,669 INFO qlms.uhfk:     E_cutoff       = 100.0
2022-12-02 13:48:11,669 INFO qlms.uhfk:     Mix            = 0.5
2022-12-02 13:48:11,669 INFO qlms.uhfk:     RndSeed        = 123456789
2022-12-02 13:48:11,669 INFO qlms.uhfk:     IterationMax   = 1000
2022-12-02 13:48:11,669 INFO qlms.uhfk:     EPS            = 1e-08
2022-12-02 13:48:11,669 INFO qlms.uhfk:     strict_hermite = False
2022-12-02 13:48:11,669 INFO qlms.uhfk:     hermite_tol    = 1e-08
2022-12-02 13:48:11,669 INFO qlms: Start UHF calculation
2022-12-02 13:48:11,670 INFO qlms.uhfk: Start UHFk calculations
2022-12-02 13:48:11,670 INFO qlms.uhfk: step, rest, energy, NCond, Sz
2022-12-02 13:48:11,671 INFO qlms.uhfk: initialize green function with random numbers
2022-12-02 13:48:11,673 INFO qlms.uhfk: 0, 0.015588886, -139.86928, 16, 1.732e-14
2022-12-02 13:48:11,684 INFO qlms.uhfk: 10, 0.00043101981, 91.751578, 16, -1.029e-11
2022-12-02 13:48:11,690 INFO qlms.uhfk: 20, 0.00097917933, 92.129093, 16, -0.0001693
2022-12-02 13:48:11,694 INFO qlms.uhfk: 30, 0.0002328601, -0.49699902, 16, -2.492e-09
2022-12-02 13:48:11,697 INFO qlms.uhfk: 40, 8.9087396e-07, -2.2626401, 16, -2.354e-14
2022-12-02 13:48:11,699 INFO qlms.uhfk: UHFk calculation succeeded: rest=9.
↪905239155412216e-09, eps=1e-08.
2022-12-02 13:48:11,699 INFO qlms: Save calculation results.
2022-12-02 13:48:11,699 INFO qlms.uhfk: save_results: save energy in file output/energy.
↪dat
2022-12-02 13:48:11,699 INFO qlms.uhfk: save_results: save eigenvalues and eigenvectors␣
↪in file output/eigen.dat
2022-12-02 13:48:11,699 INFO qlms.uhfk: save_results: save green function to file output/
↪green.dat
2022-12-02 13:48:11,700 INFO qlms: All procedures are finished.
--------------------------------------------------------------------------------
Statistics
  function                         :  total elapsed  : average elapsed : ncalls
--------------------------------------------------------------------------------
  hwave.solver.uhfk._init_param    :      0.004 msec :      0.004 msec :      1
  hwave.solver.uhfk._init_lattice  :      0.004 msec :      0.004 msec :      1
  hwave.solver.uhfk._init_orbit    :      0.001 msec :      0.001 msec :      1
  hwave.solver.uhfk._check_interaction :      0.176 msec :      0.176 msec :      1
  hwave.solver.uhfk._reshape_geometry :     23.000 msec :     23.000 msec :      1
  hwave.solver.uhfk._reshape_interaction :      0.222 msec :      0.111 msec :      2
  hwave.solver.uhfk._init_interaction :     23.313 msec :     23.313 msec :      1
  hwave.solver.uhfk._show_param    :      2.149 msec :      2.149 msec :      1
  hwave.solver.uhfk.__init__       :     28.129 msec :     28.129 msec :      1
  hwave.solver.uhfk._make_ham_trans :      0.501 msec :      0.501 msec :      1
```

(continues on next page)

(continued from previous page)

```
  hwave.solver.uhfk._make_ham_inter :      0.414 msec :      0.414 msec :      1
  hwave.solver.uhfk._reshape_green :      0.202 msec :      0.202 msec :      1
  hwave.solver.uhfk._initial_green :      0.494 msec :      0.494 msec :      1
  hwave.solver.uhfk._make_ham      :      6.999 msec :      0.143 msec :     49
  hwave.solver.uhfk._diag          :      3.533 msec :      0.072 msec :     49
  hwave.solver.uhfk._green         :      8.698 msec :      0.178 msec :     49
  hwave.solver.uhfk._calc_energy   :      3.960 msec :      0.081 msec :     49
  hwave.solver.uhfk._calc_phys     :      3.559 msec :      0.073 msec :     49
  hwave.solver.uhfk.solve          :     29.349 msec :     29.349 msec :      1
  hwave.solver.uhfk._deflate_green :      0.035 msec :      0.035 msec :      1
  hwave.solver.uhfk._save_green    :      0.202 msec :      0.202 msec :      1
  hwave.solver.uhfk.save_results   :      0.559 msec :      0.559 msec :      1
--------------------------------------------------------------------------------
```

The logs on the input files are shown, followed by the logs on the iterations of the wave-number space UHF calculation. The program will yield, according to the settings in `[file.output]` section, the output files `energy.dat`, `eigen.npz`, and `green.npz` in `output` directory.

See *Output files of UHFk* section for the details of the output files.

4.1.4 Calculate density of state (`hwave_dos`)

You can calculate the density of states (DOS) using the post-processing tool, `hwave_dos`.

`hwave_dos` uses the `libtetrabz` library to integrating over the Brillouin zone. `libtetrabz` can be installed by `pip`:

```
$ python3 -m pip install libtetrabz
```

The tool requires the input parameter file used in the calculation. The following example shows how to calculate the DOS using the sample input file:

```
$ hwave_dos input.toml
```

`hwave_dos` outputs the DOS file, `dos.dat` in the directory specified by the `file.output.path_to_output` of the input file. The filename can be changed by `--output` option:

```
$ hwave_dos input.toml --output dos.dat
```

The DOS is calculated in the energy range specified by `--ene-window` option. If omitted, the energy range is set to $[E_{\rm min} - 0.2, E_{\rm max} + 0.2]$ where $E_{\rm min}$ and $E_{\rm max}$ are the minimum and maximum energies obtained by `hwave`. The number of the energy points for the DOS calculation is specified by `--ene-num` option (default is 101):

```
$ hwave_dos input.toml --ene-window -10.0 5.0 --ene-num 201
```

The `--plot` option plots the DOS. `matplotlib` is required:

```
$ hwave_dos input.toml --plot dos.png
```

4.2 File specifications

4.2.1 Parameter files

The parameter file specifies calculation conditions and parameters for H-wave in TOML format. It is composed of the following three sections.

1. `mode` section for specifying calculation conditions,
2. `log` section for setting standard outputs,
3. `file` section for setting file paths: It contains `input` and `output` subsections.

An example of the file is shown below:

```
[log]
  print_level = 1
  print_step = 10
[mode]
  mode = "UHFk"
[mode.param]
  # 2Sz = 0
  Ncond = 16
  IterationMax = 1000
  EPS = 8
  Mix = 0.5
  RndSeed = 123456789
  # ene_cutoff = 1.0e+2
  T = 0.0
  CellShape = [ 4, 4, 1 ]
  SubShape = [ 2, 2, 1 ]
[file]
[file.input]
  path_to_input = ""
  # initial = "green_init.dat.npz"
[file.input.interaction]
  path_to_input = "./"
  Geometry = "geom.dat"
  Transfer = "transfer.dat"
  CoulombInter = "coulombinter.dat"
[file.output]
  path_to_output = "output"
  energy = "energy.dat"
  eigen = "eigen"
  green = "green"
```

File format

TOML format

Parameters

`mode` section

- `mode`

  Type : String

  Description : This parameter specifies the calculation mode. Set to `"UHFk"` for calculations of the wave-number space UHF.

- `flag_fock`

  Type : Boolean (Default value is `true`)

  Description : If `true`, include the Fock term in the Hamiltonian, otherwise, exclude it (Hartree approximation).

- `enable_spin_orbital` (default value is `false`)

  Type : Boolean

  Description : This parameter specifies whether to allow spin-orbital interaction. If it is set to true, the orbital indices in Transfer term are interpreted in the way that they include the orbital index α and the spin index s by $\alpha + N_{\mathrm{orb}} \cdot s$.

`mode.param` section

`mode.param` section contains the parameters for the calculation.

- `CellShape`

  Type : Integer array

  Description : This parameter specifies the shape of the lattice Lx, Ly, Lz.

- `SubShape`

  Type : Integer array (default value is `[` Lx, Ly, Lz `]`)

  Description : This parameter specifies the shape of the sublattice Bx, By, Bz.

- `T`

  Type : Float (default value is 0)

  Description : This parameter specifies the temperature. It must be greater than or equal to zero.

- `Ncond`

  Type : Integer

  Description : This parameter specifies the number of conduction electrons. It must be greater than or equal to one.

- `filling`

  Type : Float

Description : This parameter specifies the filling ratio of electrons with respect to the number of states. It must be between 0 and 1. Both `Ncond` and `filling` are specified, the program will be terminated with error.

- `Ncond_round_mode`

  Type : String (default value is `"strict"`)

  Description : This parameter specifies how the number of electrons calculated from the `filling` parameter is rounded to an integer value. The parameter must take one of the following values.

  - `as-is`: the value is not rounded to an integer. (returns a floating-point number)
  - `round-up`: the value is rounded up.
  - `round-down`: the value is rounded down.
  - `round-off`: the value is rounded to the closest integer. (0.5 is rounded up.)
  - `round`: the value is rounded by `round` function. (0.5 is rounded down.)
  - `strict`: if the value is not an integer value, the program terminates with error.
  - `exact`: if the value is not an integer value, a warning message will be shown and the value is rounded to an integer as `round`.

- `IterationMax`

  Type : Integer (default value is 20000)

  Description : This parameter specifies the maximum number of iterations. It must be greater than or equal to zero.

- `EPS`

  Type : Integer (default value is 6)

  Description : This parameter specifies the convergence criterion. It is considered convergent when the norm of the difference between the previous and the new Green's function falls below $10^{-\mathrm{EPS}}$. The residue is defined by $R = \sum_{i,j}^{N} \sqrt{\left|G_{ij}^{\mathrm{new}} - G_{ij}^{\mathrm{old}}\right|^2}/2N^2$. It must be greater than or equal to zero.

- `Mix`

  Type : Float (default value is 0.5)

  Description : This parameter specifies the ratio α of simple-mixing when the Green's function is updated by the previous and the new one. It must be between 0 and 1. If it is set to 1, the previous value will not be mixed.

  See *Algorithms* section for the simple-mixing algorithm.

- `RndSeed`

  Type : Integer (default value is 1234)

  Description : This parameter specifies the seed of the random number generator.

- `ene_cutoff`

  Type : Float (default value is 100.0)

  Description : This parameter specifies the cutoff to avoid overflow in the calculations of the Fermi distribution function.

- `strict_hermite`

  Type : Boolean (default value is false)

Description : This parameter specifies the strictness of checking Hermiticity of the interaction definitions when they are read from files. If it is true, the program will be terminated with error when there are deviations greater than `hermite_tolerance`. If it is false, only the warning messages will be shown, and the calculation continues.

- `hermite_tolerance`

  Type : Float (default value is 10^{-8})

  Description : This parameter specifies the tolerance for the Hermiticity condition $|t_{ij} - t_{ji}^*| < \varepsilon$.

- `trustme_interaction_range`

  Type : Boolean (default value is false)

  Description :

  Relax the checks that the distances of hopping and interaction are within the halves of `CellShape`; if true, warn and continue execution, otherwise stop.

`log` section

- `print_level`

  Type : Integer (default value is 1)

  Description : This parameter specifies the verbosity of the standard output. If it is set to 1, a detailed information will be shown.

- `print_step`

  Type : Integer (default value is 1)

  Description : This parameter specifies the interval to write calculation logs to the standard output during the iteration. It must be greater than or equal to one.

- `print_check`

  Type : String (default value is None)

  Description : This parameter specifies the filename to which the calculation logs are written during the iteration besides the standard output. If it is not set, no output file will be generated.

`file` section

This section consists of `input` and `output` subsections. The former specifies the settings on the input files (such as locations and name of the files), and the latter on the output files (such as locations to store).

`file.input` section

- `path_to_input`

  Type : String (default value is `""`)

  Description : This parameter specifies the directory in which the input files are located.

- `initial`

  Type : String

  Description : This parameter specifies the filename of the initial configuration of the one-body Green's function. The input file is in NumPy binary format that corresponds to the output format of `green` in `file.output` section.

- `initial_mode`

  Type : String

  Description : This parameter specifies how the initial value of the Green's function is set when the initial configuration is not given from files. It takes one of the following:

  - `zero`: set the initial configuration to be zero. (default)
  - `one` or `unity`: set the initial configuration as $G_{\alpha\sigma,\beta\sigma'}(\vec{r}) = \delta_{\vec{r},0}\delta_{\alpha\beta}\delta_{\sigma\sigma'}$.
  - `random`: set the initial configuration by random numbers.

`file.input.interaction` section

This section describes the relation of the interaction types and geometry information to the definition files.

- `path_to_input`

  Type : String

  Description : This parameter specifies the directory in which the input files are located. It is independent from `path_to_input` in `file.input` section.

- `Geometry`

  Type : String

  Description : This parameter specifies the filename for the geometry information.

- `Transfer`, `CoulombIntra`, `CoulombInter`, `Hund`, `Ising`, `Exchange`, `PairLift`, `PairHop`

  Type : String

  Description : These parameters specify the filenames for the definitions of the corresponding interaction terms.

`file.output` section

- `path_to_output`

  Type : String (default value is `"output"`)

  Description : This parameter specifies the directory in which the output files are stored.

- `energy`

  Type : String

  Description :

  This parameter specifies the name of the file to store the energy values. If it is not set, no output will be generated.

- `eigen`

  Type : String

  Description : This parameter specifies the name of the file to store the eigenvalues and eigenvectors of the Hamiltonian. If it is not set, no output will be generated.

- `green`

  Type : String (default value is `"green"`)

  Description : This parameter specifies the name of the file to store the one-body Green's functions. If it is set to an empty string, no output will be generated.

- `rpa`

  Type : String

  Description : This parameter specifies the name of the file to store the one-body term involving the approximated two-body interaction term by UHF method. The output may be used as an initial configuration of the RPA calculation by specifying it in the parameter `file.input.trans_mod`. If it is not set, no output will be generated.

4.2.2 Input files for UHFk

In this section, the input files for the wave-number space UHF are described. They are classified into two categories, and written in Wannier90 format.

(1) Geometry information

  Geometry defines the geometrical information of the lattice.

(2) Interaction definitions

  These files defines the Hamiltonian for UHF in the form of electron systems. They provide the coefficients of the interaction terms associated with the specified keywords.

  The following keywords adopted in $\mathcal{H}\Phi$ and mVMC in the Expert Mode are accepted.

  Transfer corresponds to one-body term denoted by $c_{i\sigma_1}^{\dagger} c_{j\sigma_2}$.

  CoulombIntra corresponds to the interaction denoted by $n_{i\uparrow} n_{i\downarrow}$, where $n_{i\sigma} = c_{i\sigma}^{\dagger} c_{i\sigma}$.

  CoulombInter corresponds to the interaction denoted by $n_i n_j$, where $n_i = n_{i\uparrow} + n_{i\downarrow}$.

  Hund corresponds to the interaction denoted by $n_{i\uparrow} n_{j\uparrow} + n_{i\downarrow} n_{j\downarrow}$.

  Ising corresponds to the interaction denoted by $S_i^z S_j^z$.

  Exchange corresponds to the interaction denoted by $S_i^+ S_j^-$.

  PairLift corresponds to the interaction denoted by $c_{i\uparrow}^{\dagger} c_{i\downarrow} c_{j\uparrow}^{\dagger} c_{j\downarrow}$.

  PairHop corresponds to the interaction denoted by $c_{i\uparrow}^{\dagger} c_{j\uparrow} c_{i\downarrow}^{\dagger} c_{j\downarrow}$.

The data formats are described in the following sectoins.

Geometry input file

The Geometry input file describes the geometry information of the lattice. An example of the file is shown as follows.

```
3.7599302871   0.0000000000   0.0000000000
0.0000000000   3.7599302871   0.0000000000
0.0000000000   0.0000000000   5.4822004186
     10
-7.179835091886330E-003 -3.812050198019962E-002  1.639284152926924E-003
 1.346463812592166E-002  6.709778405878775E-003 -6.812442303544219E-003
 0.495705070884200      -0.457955704941170      -4.077818544354700E-003
-1.577970702078565E-004 -2.999005205319096E-004 -1.190284144276225E-004
-1.302397074478660E-003 -5.021621895411691E-003 -3.514564279609852E-004
 0.504124376959700       0.457760356450585      -2.634809811615298E-003
 0.499384075989520      -0.494227365093439       6.927730957590197E-003
-5.164444920392309E-003  3.667887236852975E-002  4.972296517752579E-003
```

(continues on next page)

(continued from previous page)

```
0.500170586121734       0.499747448247510       2.760670734661295E-003
0.500734036298328       0.494793997305026      -2.212377045150314E-003
```

File format

- Lines 1-3: `[ax_i] [ay_i] [az_i]`
- Line 4: `[Norbit]`
- Lines 5-: `[vx_i] [vy_i] [vz_i]`

Parameters

- `[ax_i]`, `[ay_i]`, `[az_i]`

  Type : Float

  Description : These parameters for `i` from 1 to 3 specify the primitive vectors $\vec{a}_1, \vec{a}_2, \vec{a}_3$.

- `[Norbit]`

  Type : Integer

  Description : This parameter specifies the number of orbitals N_{orbit} in a unitcell.

- `[vx_i]`, `[vy_i]`, `[vz_i]`

  Type : Float

  Description : These parameters specify the Wannier center $\vec{v}_i$ of each orbital in the fractional coordinates.

Usage rules

- The indices of the orbitals are implicitly assigned from 1 to N_{orbit} in the order of the Wannier centers.

Interaction definition files

The interaction definition files describe the coefficients $T_{\alpha\beta}(r_{ij})$, $J_{\alpha\beta}(r_{ij})$, $V_{\alpha\beta}(r_{ij})$, or U_α of the one-body and two-body Hamiltonian denoted by the following expressions. They are given in Wannier90(-like) format. It is noted that the generalized two-body interaction term (InterAll) is not supported in the wave-number space UHF.

Transfer: $\sum_{ij\alpha\beta\sigma} T_{\alpha\beta}(r_{ij})\, c_{i\alpha\sigma}^\dagger c_{j\beta\sigma}$

CoulombIntra: $\sum_{i\alpha} U_\alpha\, n_{i\alpha\uparrow} n_{i\alpha\downarrow}, \quad (n_{i\alpha\sigma} = c_{i\alpha\sigma}^\dagger c_{i\alpha\sigma})$

CoulombInter: $\sum_{ij\alpha\beta} V_{\alpha\beta}(r_{ij})\, n_{i\alpha} n_{j\beta}, \quad (n_{i\alpha} = n_{i\alpha\uparrow} + n_{i\alpha\downarrow})$

Hund: $\sum_{ij\alpha\beta} J_{\alpha\beta}^{\text{Hund}}(r_{ij})\, (n_{i\alpha\uparrow} n_{j\beta\uparrow} + n_{i\alpha\downarrow} n_{j\beta\downarrow})$

Ising: $\sum_{ij\alpha\beta} J_{\alpha\beta}^{\text{Ising}}(r_{ij}) S_{i\alpha}^z S_{j\beta}^z, \quad (S_{i\alpha}^z = \frac{1}{2}(n_{i\alpha\uparrow} - n_{i\alpha\downarrow}))$

PairHop: $\sum_{ij\alpha\beta} J_{\alpha\beta}^{\text{PH}}(r_{ij})\, c_{i\alpha\uparrow}^\dagger c_{j\beta\uparrow} c_{i\alpha\downarrow}^\dagger c_{j\beta\downarrow} + h.c.$

Exchange: $\sum_{ij\alpha\beta} J_{\alpha\beta}^{\text{Ex}}(r_{ij})\, c_{i\alpha\uparrow}^\dagger c_{j\beta\uparrow} c_{j\beta\downarrow}^\dagger c_{i\alpha\downarrow}$

PairLift: $\sum_{ij\alpha\beta} J_{\alpha\beta}^{\text{PairLift}}(r_{ij})\, c_{i\alpha\uparrow}^\dagger c_{i\alpha\downarrow} c_{j\beta\uparrow}^\dagger c_{j\beta\downarrow}$

An example of the file is shown below.

```
wannier90 format for vmcdry.out or HPhi -sdry
    10
   245
 1    1    1    1    1    1    1    1    1    1    1    1    1    1    1
 1    1    1    1    1    1    1    1    1    1    1    1    1    1    1
...
 1    1    1    1    1
-3   -3   -2    1    1  -0.0000269645  -0.0000000000
-3   -3   -2    1    2  -0.0000071722  -0.0000018600
-3   -3   -2    1    3  -0.0000083990   0.0000010972
-3   -3   -2    1    4  -0.0000000990   0.0000000427
-3   -3   -2    1    5  -0.0000018628  -0.0000003609
-3   -3   -2    1    6  -0.0000129504  -0.0000014047
-3   -3   -2    1    7  -0.0000189169   0.0000024697
-3   -3   -2    1    8   0.0000238115   0.0000014316
-3   -3   -2    1    9   0.0000036708  -0.0000003266
-3   -3   -2    1   10   0.0000361752   0.0000003247
-3   -3   -2    2    1  -0.0000071722   0.0000018600
-3   -3   -2    2    2   0.0000105028  -0.0000000000
...
```

File format

- Line 1: Header
- Line 2: `[Norbit]`
- Line 3: `[Npts]`
- Lines 4 - $\lceil N_{\text{pts}}/15 \rceil + 3$:

  `[n1] [n2] ...`

- Line $\lceil N_{\text{pts}}/15 \rceil + 4$ onwards:

  `[rx] [ry] [rz] [alpha] [beta] [J.real] [J.imag]`

Parameters

- `[Norbit]`

  Type : Integer

  Description : This parameter specifies the number of orbitals N_{orbit} in a unit cell.

- `[Npts]`

  Type : Integer

  Description : This parameter specifies the number of cells in a rectangular cuboid that accommodates entire translation vectors.

- `[n1]`, `[n2]`, ...

  Type : Integer

  Description : These parameters specify the multiplicity of cells (ordinary 1), with 15 points in a line.

- `[rx]`, `[ry]`, `[rz]`

  Type : Integer

  Description : These parameters specify the translation vector.

- `[alpha]`, `[beta]`

  Type : Integer

  Description : These parameters specify the indices of the orbitals. `[alpha]` corresponds to the orbital α in the original cell, and `[beta]` corresponds to the orbital β in the cell displaced by $\vec{r}$.

- `[J.real]`, `[J.imag]`

  Type : Float

  Description : These parameters specify the real and imaginary parts of the coefficient $J_{\alpha\beta}(\vec{r})$.

Usage rules

- Header cannot be omitted.
- The unspecified elements of the coefficient matrix are assumed to be zero.
- The translation vectors need to be enclosed within the CellShape. If the range of `r_x`, `r_y`, or `r_z` exceeds the extent of `x`, `y`, or `z` dimension of CellShape, the program terminates with an error.
- When `mode.enable_spin_orbital` is set to `true`, the orbital indices of Transfer term are interpreted as the extended orbital indices including spin degree of freedom that ranges from 1 to $2N_{\text{orbital}}$, in which the indices $1 \ldots N_{\text{orbital}}$ correspond to spin-up, and the indices $N_{\text{orbital}} + 1 \ldots 2N_{\text{orbital}}$ correspond to spin-down. Otherwise, only the entries with the orbital indices from 1 to N_{orbital} are taken into account.

4.2.3 Output files of UHFk

This section describes the file formats of the outputs of UHFk.

energy

The values of the energy, the number of electrons, and the spin obtained by the wave-number space UHF method are outputted. The filename is specified by the keyword `energy` in `file.output` section in the parameter file.

An example of the file is shown as follows.

```
Energy_Total = -5.88984624257707
Energy_Band = -0.9265413257740396
Energy_Coulomb = -4.963304916803031
NCond = 8.000000000000007
Sz = 3.2822430107160017e-07
```

File format

- `Energy_Total = [energy_total]`
- `Energy_Band = [energy_band]`
- `Energy_{type} = [energy_type ]`
- `NCond = [ncond]`
- `Sz = [sz]`

Parameters

- `[energy_total]`

  Type : Float

  Description : The value of the energy calculated from the eigenvectors obtained by the UHF mothod.

- `[energy_band]`

  Type : Float

  Description : The value of the energy derived only from the eigenvalues of the Hamiltonian obtained by the UHF method.

- `[energy_type ]`

  Type : Float

  Description : The value of the energy calculated separately for each interaction type.

- `[ncond]`

  Type : Float

  Description : The expectation value of the total number of electrons denoted by $\sum_i \langle n_i \rangle$.

- `[sz]`

  Type : Float

  Description : The expectation value of the z component of total spin S_z denoted by $\sum_i \langle (n_{i\uparrow} - n_{i\downarrow}) \rangle / 2$.

eigen

The eigenvalues and eigenvectors of the Hamiltonian at the convergence are exported in NumPy zip (npz) format. Using the string (referred to as *eigen_str*) specified by the keyword `eigen` in `file.output` section in the parameter file, the filename is chosen as *eigen_str*`.npz`.

The following code is an example for reading the data from the output file.

```
import numpy as np
data = np.load("eigen_str.npz")
eigenvalue = data["eigenvalue"]
eigenvector = data["eigenvector"]

wavevector_unit = data["wavevector_unit"]
wavevector_index = data["wavevector_index"]
```

`eigenvalue` contains the eigenvalues $\lambda_l(\vec{k})$ for each wave number. The wave number is taken in unit of sublattice when the sublattice is considered. The data format is a numpy ndarray with the layout as `eigenvalue[k][l]`, where `k` refers to the linearlized index of the wave number vector $\vec{k}$ (see below), and `l` refers to the index of eigenvalue. When `Sz` is fixed, `l` is given by `l = l' + Norb * s` where `l'` is the index of the eigenvalue in a cell, and `s` refers to the spin index (0 for up-spin, and 1 for down-spin).

`eigenvector` contains the corresponding eigenvectors. The data format is a numpy ndarray with the layout as `eigenvector[k][j][l]`, where `k` and `l` refer to the indices of the corresponding wave number and eigenvalue, and `j` refers to the index of the orbital and spin in a cell.

`wavevector_unit` and `wavevector_index` refer to the information of the wave number vectors. `wavevector_unit` contains the unit wave number vectors given by $2\pi\vec{b}_i/N_i$ with $\vec{b}_i$ being reciprocal lattice vectors. `wavevector_index` contains the map from the index `k` to the indices of the wave number vector `(kx, ky, kz)`. The wave number vector that corresponds to the index `k` is obtained by

```
k_vec = np.dot(wavevector_index[k], wavevector_unit)
```

green

The one-body Green's function $\langle c^\dagger_{i\sigma_1} c_{j\sigma_2}\rangle$ calculated by the wave-number space UHF method is exported in NumPy zip (npz) format. Using the string (referred to as *green_str*) specified by the keyword `green` in `file.output` section in the parameter file, the filename is chosen as *green_str*`.npz`.

The data is bound to the key `green`. The data format is a numpy ndarray with the layout `ndarray(r, s, a, t, b)`, where

- `r` denotes a linearlinzed index of translation vector $[r_x\ r_y\ r_z]$, where the indices are packed into `r` by $\mathtt{r} = r_z + N_z \cdot (r_y + N_y r_x)$.
- `a`, `b` denote the indices of the orbitals α, β,
- `s`, `t` denote the indices of the spins σ_1, σ_2.

The output can be used as an initial configuration of the Green's function specified by the keyword `initial` in `file.input` section.

When the sublattice is considerd, the Green's function in unit of the sublattice is also stored with the key `green_sublattice`. The indices of the data are regarded as those of the sublattice.

The following code is an example for reading the data from the output file.

```
import numpy as np
data = np.load("green.dat.npz")
green = data["green"]
```

CHAPTER

FIVE

RANDOM PHASE APPROXIMATION (RPA)

5.1 Tutorial

To use H-wave for the calculation of Random Phase Approximation (RPA), you need to prepare input files:

1. an input parameter file,
2. interaction definition files,

before running the program.

In the following, we provide a tutorial based on a sample in `docs/tutorial/Hubbard/RPA` directory. The interaction definition files can be generated using StdFace library. See *Generation of interaction files using StdFace library* section for the details.

5.1.1 Create a parameter file

We prepare an input parameter file that contains basic parameters as well as settings on inputs and outputs. A sample file can be found in `docs/tutorial/Hubbard/RPA` directory by a filename `input.toml`. The content of the file is as follows:

```
[log]
  print_level = 1

[mode]
  mode = "RPA"

[mode.param]
  T = 0.5
  # mu = 0.0
  CellShape = [32,32,1]
  SubShape = [1,1,1]
  nmat = 1024
  # Ncond = 1024
  filling = 0.5
  matsubara_frequency = "all"

[file]
[file.input]
  path_to_input = "input"
  # initial = "initial.dat"
  # chi0q_init = "chi0q.npz"
```

(continues on next page)

(continued from previous page)

```
[file.input.interaction]
  path_to_input = "."
  Geometry = "geom.dat"
  Transfer = "transfer.dat"
  CoulombIntra = "coulombintra.dat"
  CoulombInter = "coulombinter.dat"

[file.output]
  path_to_output = "output"
  chiq = "chiq"
  chi0q = "chi0q"
```

The file is written in TOML format, and organized by the following sections.

[log] section

This section contains settings on log outputs. `print_level` sets the verbosity of the standard output, and `print_step` sets the interval of the outputs.

[mode] section

This section contains settings on calculation mode and basic parameters. We choose `mode` to be `"RPA"`. The calculation parameters are specified in `[mode.param]` subsection.

[file] section

`[file.input]` subsection contains settings on the directory for the input files by `path_to_input`. It is possible to read the pre-calculated irreducible susceptibility $\chi_0(\vec{q})$ from a file specified by `chi0q_init` and calculate the susceptibility.

`[file.input.interaction]` subsection contains a list of files associated with the geometry information and the interactions distinguished by the keywords.

`[file.output]` subsection contains filenames for the physical observables such as the energy by `energy`, for the eigenvalues and eigenvectors of the Hamiltonian by `eigen`, and for the one-body Green's functions by `green`. If they are not specified, the corresponding data will not be outputted.

See *Parameter files* section for the details.

5.1.2 Create interaction definition files

You need to prepare data files on the geometry information of the lattice, and the coefficients of the interactions required to construct the Hamiltonian. The association between the types of information and the filenames is provided in `[file.input.interaction]` section.

Geometry

The file associated with Geometry provides the geometrical information of the lattice. An example of the file is shown below.

```
  1.000000000000   0.000000000000   0.000000000000
  0.000000000000   1.000000000000   0.000000000000
  0.000000000000   0.000000000000   1.000000000000
1
    0.000000000000000e+00     0.000000000000000e+00     0.000000000000000e+00
```

It contains the primitive vectors (lines 1–3), the number of orbitals (line 4), and the Wannier centers of the orbitals (line 5 onwards).

Transfer, CoulombIntra, CoulombInter, Hund, etc

The file associated with Transfer contains the coefficients of Hamiltonian corresponding to the transfer term of the electron systems. The coefficients of the two-body interaction terms are stored in the files associated with the respective keywords. The defined types include CoulombIntra, CoulombInter, Hund, Ising, Exchange, PairLift, and PairHop, in accordance with the UHF calculation. These files are written in Wannier90(-like) format, as exemplified below.

```
Transfer in wannier90-like format for uhfk
1
9
 1 1 1 1 1 1 1 1 1
  -1   -1    0    1    1  0.500000000000  -0.000000000000
  -1    0    0    1    1  1.000000000000  -0.000000000000
   0   -1    0    1    1  1.000000000000  -0.000000000000
   0    1    0    1    1  1.000000000000   0.000000000000
   1    0    0    1    1  1.000000000000   0.000000000000
   1    1    0    1    1  0.500000000000   0.000000000000
```

It contains a comment (line 1), the number of orbitals (line 2), `nrpts` (line 3), the multiplicity factors (`nrpts` elements, with 15 elements per line), and the elements of the coefficient matrix. `nrpts` denotes the number of cells of the rectangular cuboid that is spanned by the lower and upper ends of the translation vectors along x, y, and z axes.

Each element of the matrix consists of translation vector r_x, r_y, r_z, indices of orbitals α, β, and the real and imaginary part of the coefficient.

5.1.3 Run

Once you prepare all the input files, you can perform the calculation by running H-wave with the input parameter file (`input.toml` in this tutorial) as an argument.

```
$ hwave input.toml
```

The calculation starts with the logs as shown below:

```
2023-03-07 18:55:44,682 INFO qlms: RPA mode
2023-03-07 18:55:44,682 INFO qlms: Read interaction definitions from files
2023-03-07 18:55:44,682 INFO qlms.read_input: QLMSkInput: read Gemoetry from input/geom.
↪dat
2023-03-07 18:55:44,682 INFO qlms.read_input: QLMSkInput: read interaction Transfer from␣
↪input/transfer.dat
```

(continues on next page)

(continued from previous page)

```
2023-03-07 18:55:44,682 INFO qlms.read_input: QLMSkInput: read interaction CoulombIntra␣
↪from input/coulombintra.dat
2023-03-07 18:55:44,682 INFO qlms.read_input: QLMSkInput: read interaction CoulombInter␣
↪from input/coulombinter.dat
2023-03-07 18:55:44,682 INFO hwave.solver.rpa: Lattice parameters:
2023-03-07 18:55:44,682 INFO hwave.solver.rpa:     CellShape       = (32, 32, 1)
2023-03-07 18:55:44,682 INFO hwave.solver.rpa:     cell volume     = 1024
2023-03-07 18:55:44,682 INFO hwave.solver.rpa:     cell dimension  = 3
2023-03-07 18:55:44,682 INFO hwave.solver.rpa:     SubShape        = (1, 1, 1)
2023-03-07 18:55:44,682 INFO hwave.solver.rpa:     subshape volume = 1
2023-03-07 18:55:44,682 INFO hwave.solver.rpa:     Shape           = (32, 32, 1)
2023-03-07 18:55:44,682 INFO hwave.solver.rpa:     shape volume    = 1024
2023-03-07 18:55:44,682 INFO hwave.solver.rpa:     has_sublattice  = False
2023-03-07 18:55:44,683 INFO hwave.solver.rpa: RPA parameters:
2023-03-07 18:55:44,683 INFO hwave.solver.rpa:     norbit          = 1
2023-03-07 18:55:44,683 INFO hwave.solver.rpa:     nspin           = 2
2023-03-07 18:55:44,683 INFO hwave.solver.rpa:     nd              = 2
2023-03-07 18:55:44,683 INFO hwave.solver.rpa:     Nmat            = 1024
2023-03-07 18:55:44,683 INFO hwave.solver.rpa:     mu              = 0.0
2023-03-07 18:55:44,683 INFO hwave.solver.rpa:     T               = 0.5
2023-03-07 18:55:44,683 INFO hwave.solver.rpa:     E_cutoff        = 1.000000e+02
2023-03-07 18:55:44,683 INFO qlms: Start UHF calculation
2023-03-07 18:55:44,683 INFO hwave.solver.rpa: Start RPA calculations
2023-03-07 18:55:47,726 INFO hwave.solver.rpa: End RPA calculations
2023-03-07 18:55:47,726 INFO qlms: Save calculation results.
2023-03-07 18:55:47,726 INFO hwave.solver.rpa: Save RPA results
2023-03-07 18:55:47,925 INFO hwave.solver.rpa: save_results: save chiq in file output/
↪chiq
2023-03-07 18:55:48,294 INFO hwave.solver.rpa: save_results: save chi0q in file output/
↪chi0q
2023-03-07 18:55:48,294 INFO qlms: All procedures are finished.
--------------------------------------------------------------------------------
Statistics
  function                         :  total elapsed  : average elapsed : ncalls
--------------------------------------------------------------------------------
  hwave.solver.rpa.__init__        :      1.037 msec :      1.037 msec :      1
  hwave.solver.rpa.read_init       :      0.001 msec :      0.001 msec :      1
  hwave.solver.rpa._calc_epsilon_k :     22.587 msec :     22.587 msec :      1
  hwave.solver.rpa._calc_green     :    130.035 msec :    130.035 msec :      1
  hwave.solver.rpa._calc_chi0q     :   1886.201 msec :   1886.201 msec :      1
  hwave.solver.rpa._solve_rpa      :   1003.617 msec :   1003.617 msec :      1
  hwave.solver.rpa.solve           :   3042.926 msec :   3042.926 msec :      1
  hwave.solver.rpa.save_results    :    567.897 msec :    567.897 msec :      1
--------------------------------------------------------------------------------
```

The logs on the input files are shown, followed by the logs on the process of RPA calculations. The program will yield, according to the settings in `[file.output]` section, the output files `chi0q.npz` and `chiq.npz` in `output` directory.

See *Output files of RPA* section for the details of the output files.

A tool is prepared in `sample/RPA/view.py` for visualizing the calculation results as a post-process. Let us copy the script file to the current directory, and run the script as follows:

```
$ python3 view.py
```

The script reads `output/chi0q.npz` and `output/chiq.npz`, and writes the values of the charge susceptibility $\chi_c(\vec{q})$ and the spin susceptibility $\chi_s(\vec{q})$ at Matsubara frequency $\mathrm{i}\omega_m = 0$ for each $\vec{q}$ to the standard output. It also produces the figures of these quantities in PNG format shown as below:

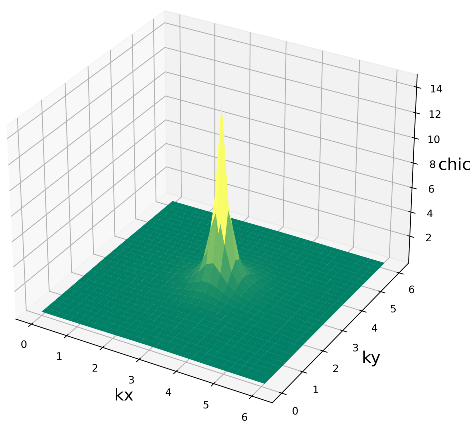

Fig. 5.1: $\chi_c(\vec{q})$

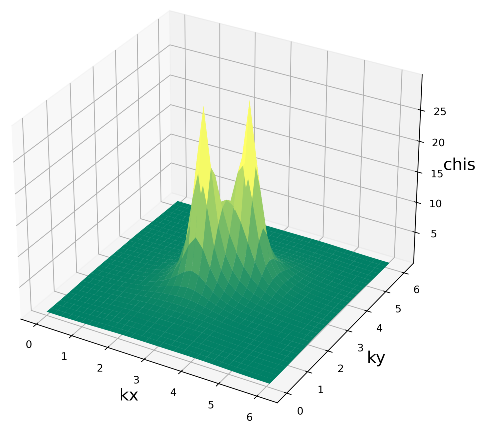

Fig. 5.2: $\chi_s(\vec{q})$

5.2 File specifications

5.2.1 Parameter files

The parameter file specifies calculation conditions and parameters for H-wave in TOML format. It is composed of the following three sections.

1. `mode` section for specifying calculation conditions,
2. `log` section for setting standard outputs,
3. `file` section for setting file paths: It contains `input` and `output` subsections.

An example of the file is shown below:

```
[log]
  print_level = 1

[mode]
  mode = "RPA"

[mode.param]
  T = 0.5
  # mu = 0.0
  CellShape = [32,32,1]
  SubShape = [1,1,1]
```

(continues on next page)

(continued from previous page)

```
  nmat = 1024
  # Ncond = 1024
  filling = 0.5
  matsubara_frequency = "all"

[file]
[file.input]
  path_to_input = "input"
  # initial = "initial.dat"
  # chi0q_init = "chi0q.npz"

[file.input.interaction]
  path_to_input = "."
  Geometry = "geom.dat"
  Transfer = "transfer.dat"
  CoulombIntra = "coulombintra.dat"
  CoulombInter = "coulombinter.dat"

[file.output]
  path_to_output = "output"
  chiq = "chiq"
  chi0q = "chi0q"
```

File format

TOML format

Parameters

`mode` section

- `mode`

  Type : String

  Description : This parameter specifies the calculation mode. Set to `"RPA"` for calculations of the Random Phase Approximation.

- `enable_spin_orbital` (default value is `false`)

  Type : Boolean

  Description : This parameter specifies whether to allow spin-orbital interaction. If it is set to true, the orbital indices in Transfer term are interpreted in the way that they include the orbital index α and the spin index s by $\alpha + N_{\mathrm{orb}} \cdot s$.

- `calc_scheme` (default value is `"auto"`)

  Type : String

  Description : This parameter specifies how the spin and orbitals are treated in the calculation. The parameter takes one of the following options.

- `general`: Generalized orbitals combining spins and orbitals are considered. The susceptibility matrix takes the most general form, with the size of $N_{\text{orb}}^4 N_{\text{spin}}^4 N_k N_\omega$.
- `reduced`: Generalized orbitals combining spins and orbitals are considered. The components of the susceptibility matrix with $\alpha = \alpha'$ and $\beta = \beta'$ are considered. The size of the matrix turns to $N_{\text{orb}}^2 N_{\text{spin}}^2 N_k N_\omega$. For the two-body interaction terms, only CoulombIntra, CoulombInter, Ising and Hund are allowed.
- `squashed`: Spins and orbitals are separately treated, and for the orbitals $\alpha = \alpha'$ and $\beta = \beta'$ are considered. The size of the susceptibility matrix becomes $N_{\text{orb}}^2 N_{\text{spin}}^4 N_k N_\omega$. See *Algorithms* for details.
- `auto`: scheme is automatically chosen according to the specifications of interaction terms. This option is not available when only `chi0q` is to be calculated.

`mode.param` section

`mode.param` section contains the parameters for the calculation.

- `CellShape`

  Type : Integer array

  Description : This parameter specifies the shape of the lattice Lx, Ly, Lz.

- `SubShape`

  Type : Integer array (default value is `[` Lx, Ly, Lz `]`)

  Description : This parameter specifies the shape of the sublattice Bx, By, Bz.

- `T`

  Type : Float (default value is 0)

  Description : This parameter specifies the temperature. It must be greater than or equal to zero.

- `mu`

  Type : Float or None (default value is None)

  Description : This parameter specifies the chemical potential μ. If it is not specified, the value of μ will be calculated so that the expectation value of the number of electrons equals to `Ncond`. If both `mu` and `Ncond` or `filling` are specified, the program terminates with error.

- `Ncond`

  Type : Integer

  Description : This parameter specifies the number of conduction electrons. It must be greater than or equal to one.

- `filling`

  Type : Float

  Description : This parameter specifies the filling ratio of electrons with respect to the number of states. Both `Ncond` and `filling` are specified, the program will be terminated with error.

- `Ncond_round_mode`

  Type : String (default value is `"strict"`)

  Description : This parameter specifies how the number of electrons calculated from the `filling` parameter is rounded to an integer value when the temperature is zero. The parameter must take one of the following values.

  - `as-is`: the value is not rounded to an integer. (returns a floating-point number)

  - `round-up`: the value is rounded up.
  - `round-down`: the value is rounded down.
  - `round-off`: the value is rounded to the closest integer. (0.5 is rounded up.)
  - `round`: the value is rounded by `round` function. (0.5 is rounded down.)
  - `strict`: if the value is not an integer value, the program terminates with error.
  - `exact`: if the value is not an integer value, a warning message will be shown and the value is rounded to an integer as `round`.

- `Nmat`

  Type : Integer (default value is 1024)

  Description : This parameter specifies the cut-off of Matsubara frequency. It must be an even number greater than zero. Matsubara frequency is defined as follows:

  - Boson: $\omega_n = \dfrac{2\pi(n - \texttt{Nmat}/2)}{\beta}$
  - Fermion: $\omega_n = \dfrac{\pi(2n + 1 - \texttt{Nmat})}{\beta}$

  with the indices n between 0 and `Nmat-1`.

- `coeff_tail`

  Type : Float (default value is 0.0)

  Description : This parameter specifies the magnitude of the correction when correcting the tails of the Fourier transformation. After Fourier transforming the diagonalized one-body Green function to the imaginary time representation by subtracting `coeff_tail`$/(i\omega_n)$, the term $-\beta/2 \cdot$ `coeff_tail` is added to the one-body Green function.

- `matsubara_frequency`

  Type : Integer, List of Integers, or String (default value is `"all"`)

  Description : This parameter specifies the indices of Matsubara frequency for which the susceptibility matrix $\chi(\vec{q})$ is calculated. The value must be one of the following:

  - *an integer value* : a single index value.
  - `[` *min*, *max* (, *step*) `]` : every *step* index from *min* to *max*. If *step* is omitted, it is assumed to be 1.
  - all : all indices
  - center : corresponds to `Nmat/2`.
  - none : nothing will be calculated.

  When the susceptibility matrix $\chi(\vec{q})$ or the irreducible susceptibility matrix $\chi_0(\vec{q})$ are stored to files, the values at the specified freqneucy are exported.

- `coeff_extern`

  Type : Float (default value is 0.0)

  Description : This parameter specifies the coefficient h of the external field given by the form $\pm h H_{\alpha\beta}(r_{ij})$. The definition of the matrix $H_{\alpha\beta}(r_{ij})$ will be provided by an input file. The sign $+$ and $-$ correspond to spin up and down, respectively.

- `RndSeed`

  Type : Integer (default value is 1234)

Description : This parameter specifies the seed of the random number generator.

- `ene_cutoff`

  Type : Float (default value is 100.0)

  Description : This parameter specifies the upper cutoff of the exponent in the Fermi distribution function to avoid overflow during the calculation.

`log` section

- `print_level`

  Type : Integer (default value is 1)

  Description : This parameter specifies the verbosity of the standard output. If it is set to 1, a detailed information will be shown.

`file` section

This section consists of `input` and `output` subsections. They specify the settings of the input and output files, respectively, on the types of files, the directories to be located or stored, and the names of the files.

`file.input` section

- `path_to_input`

  Type : String (default value is `""` (blank string))

  Description : This parameter specifies the directory in which the input files are located.

- `chi0q_init`

  Type : String

  Description : This parameter specifies the filename of the pre-calculated irreducible susceptibility $\chi_0(\vec{q})$ to be used for the calculation of the susceptibility matrix. The input file is in NumPy binary format that corresponds to the output format of `chi0q` in `file.output` section.

- `trans_mod`

  Type : String

  Description : This parameter specifies the filename of the initial configuration exported from UHFk by the parameter `file.output.rpa`. It contains the one-body interaction term involving the approximated two-body interaction terms via UHF method.

- `green_init`

  Type : String

  Description : This parameter specifies the filename of the initial Green's function for RPA calculation. The file format corresponds to the output file of `green` of UHFk. When `trans_mod` is specified, `green_init` is not used.

`file.input.interaction` section

This section describes the relation of the interaction types and geometry information to the definition files.

- `path_to_input`

  Type : String

  Description : This parameter specifies the directory in which the input files are located. It is independent from `path_to_input` in `file.input` section.

- `Geometry`

  Type : String

  Description : This parameter specifies the filename for the geometry information.

- `Transfer`, `CoulombIntra`, `CoulombInter`, `Hund`, `Ising`, `Exchange`, `PairLift`, `PairHop`, `Extern`

  Type : String

  Description : These parameters specify the filenames for the definitions of the corresponding interaction terms. If none of two-body interaction term (CoulombIntra, CoulombInter, Hund, Ising, Exchange, PairLift, or PairHop) is specified, the program only calculates `chi0q` and exits.

`file.output` section

- `path_to_output`

  Type : String (default value is `"output"`)

  Description : This parameter specifies the directory in which the output files are stored.

- `chi0q`

  Type : String

  Description :

  This parameter specifies the name of the file to store the irreducible susceptibility matrix $\chi_0(\vec{q})$. If it is not set, no output will be generated.

- `chiq`

  Type : String

  Description : This parameter specifies the name of the file to store the susceptibility matrix $\chi(\vec{q})$. If it is not set, no output will be generated.

5.2.2 Input files for RPA

In this section, the input files for the random phase approximation (RPA) are described. They are classified into two categories, and written in Wannier90 format.

(1) Geometry information

  Geometry defines the geometrical information of the lattice.

(2) Interaction definitions

  These files defines the Hamiltonian for UHF in the form of electron systems. They provide the coefficients of the interaction terms associated with the specified keywords.

The following keywords adopted in $\mathcal{H}\Phi$ and mVMC in the Expert Mode are accepted.

Transfer corresponds to one-body term denoted by $c_{i\sigma_1}^\dagger c_{j\sigma_2}$.

Extern corresponds to external field for the one-body term denoted by $\sigma^z_{\sigma_1\sigma_2} c_{i\sigma_1}^\dagger c_{j\sigma_2}$.

CoulombIntra corresponds to the interaction denoted by $n_{i\uparrow}n_{i\downarrow}$, where $n_{i\sigma} = c_{i\sigma}^\dagger c_{i\sigma}$.

CoulombInter corresponds to the interaction denoted by $n_i n_j$, where $n_i = n_{i\uparrow} + n_{i\downarrow}$.

Hund corresponds to the interaction denoted by $n_{i\uparrow}n_{j\uparrow} + n_{i\downarrow}n_{j\downarrow}$.

Ising corresponds to the interaction denoted by $S_i^z S_j^z$.

Exchange corresponds to the interaction denoted by $S_i^+ S_j^-$.

PairLift corresponds to the interaction denoted by $c_{i\uparrow}^\dagger c_{i\downarrow} c_{j\uparrow}^\dagger c_{j\downarrow}$.

PairHop corresponds to the interaction denoted by $c_{i\uparrow}^\dagger c_{j\uparrow} c_{i\downarrow}^\dagger c_{j\downarrow}$.

The data formats are described in the following sectoins.

Geometry input file

The Geometry input file describes the geometry information of the lattice. An example of the file is shown as follows.

```
3.7599302871   0.0000000000   0.0000000000
0.0000000000   3.7599302871   0.0000000000
0.0000000000   0.0000000000   5.4822004186
     10
-7.179835091886330E-003 -3.812050198019962E-002  1.639284152926924E-003
 1.346463812592166E-002  6.709778405878775E-003 -6.812442303544219E-003
 0.495705070884200      -0.457955704941170      -4.077818544354700E-003
-1.577970702078565E-004 -2.999005205319096E-004 -1.190284144276225E-004
-1.302397074478660E-003 -5.021621895411691E-003 -3.514564279609852E-004
 0.504124376959700       0.457760356450585      -2.634809811615298E-003
 0.499384075989520      -0.494227365093439       6.927730957590197E-003
-5.164444920392309E-003  3.667887236852975E-002  4.972296517752579E-003
 0.500170586121734       0.499747448247510       2.760670734661295E-003
 0.500734036298328       0.494793997305026      -2.212377045150314E-003
```

File format

- Lines 1-3: `[ax_i] [ay_i] [az_i]`
- Line 4: `[Norbit]`
- Lines 5-: `[vx_i] [vy_i] [vz_i]`

Parameters

- `[ax_i]`, `[ay_i]`, `[az_i]`

  Type : Float

  Description : These parameters for `i` from 1 to 3 specify the primitive vectors $\vec{a}_1, \vec{a}_2, \vec{a}_3$.

- `[Norbit]`

  Type : Integer

  Description : This parameter specifies the number of orbitals N_{orbit} in a unitcell.

- `[vx_i]`, `[vy_i]`, `[vz_i]`

  Type : Float

  Description : These parameters specify the Wannier center $\vec{v}_i$ of each orbital in the fractional coordinates.

Usage rules

- The indices of the orbitals are implicitly assigned from 1 to N_{orbit} in the order of the Wannier centers.

Interaction definition files

The interaction definition files describe the coefficients $T_{\alpha\beta}(r_{ij})$, $J_{\alpha\beta}(r_{ij})$, $V_{\alpha\beta}(r_{ij})$, or U_α of the one-body and two-body Hamiltonian denoted by the following expressions. They are given in Wannier90(-like) format. It is noted that the generalized two-body interaction term (InterAll) is not supported in the random phase approximation,

Transfer: $\sum_{ij\alpha\beta\sigma} T_{\alpha\beta}(r_{ij})\, c_{i\alpha\sigma}^\dagger c_{j\beta\sigma}$

Extern: $\sum_{ij\alpha\beta\sigma_1\sigma_2} H_{\alpha\beta}(r_{ij})\, \sigma^z_{\sigma_1\sigma_2}\, c_{i\alpha\sigma_1}^\dagger c_{j\beta\sigma_2}, \quad \sigma^z = \text{diag}(1, -1)$

CoulombIntra: $\sum_{i\alpha} U_\alpha\, n_{i\alpha\uparrow} n_{i\alpha\downarrow}, \quad n_{i\alpha\sigma} = c_{i\alpha\sigma}^\dagger c_{i\alpha\sigma}$

CoulombInter: $\sum_{ij\alpha\beta} V_{\alpha\beta}(r_{ij})\, n_{i\alpha} n_{j\beta}, \quad n_{i\alpha} = n_{i\alpha\uparrow} + n_{i\alpha\downarrow}$

Hund: $\sum_{ij\alpha\beta} J_{\alpha\beta}^{\text{Hund}}(r_{ij})\, (n_{i\alpha\uparrow} n_{j\beta\uparrow} + n_{i\alpha\downarrow} n_{j\beta\downarrow})$

Ising: $\sum_{ij\alpha\beta} J_{\alpha\beta}^{\text{Ising}}(r_{ij}) S^z_{i\alpha} S^z_{j\beta}, \quad S^z_{i\alpha} = \frac{1}{2}(n_{i\alpha\uparrow} - n_{i\alpha\downarrow})$

PairHop: $\sum_{ij\alpha\beta} J_{\alpha\beta}^{\text{PH}}(r_{ij})\, c_{i\alpha\uparrow}^\dagger c_{j\beta\uparrow} c_{i\alpha\downarrow}^\dagger c_{j\beta\downarrow} + h.c.$

Exchange: $\sum_{ij\alpha\beta} J_{\alpha\beta}^{\text{Ex}}(r_{ij})\, c_{i\alpha\uparrow}^\dagger c_{j\beta\uparrow} c_{j\beta\downarrow}^\dagger c_{i\alpha\downarrow}$

PairLift: $\sum_{ij\alpha\beta} J_{\alpha\beta}^{\text{PairLift}}(r_{ij})\, c_{i\alpha\uparrow}^\dagger c_{i\alpha\downarrow} c_{j\beta\uparrow}^\dagger c_{j\beta\downarrow}$

An example of the file is shown below.

```
wannier90 format for vmcdry.out or HPhi -sdry
    10
   245
 1    1    1    1    1    1    1    1    1    1    1    1    1    1    1
 1    1    1    1    1    1    1    1    1    1    1    1    1    1    1
...
 1    1    1    1    1
-3   -3   -2    1    1  -0.0000269645  -0.0000000000
-3   -3   -2    1    2  -0.0000071722  -0.0000018600
```

(continues on next page)

(continued from previous page)

```
-3   -3   -2    1    3  -0.0000083990   0.0000010972
-3   -3   -2    1    4  -0.0000000990   0.0000000427
-3   -3   -2    1    5  -0.0000018628  -0.0000003609
-3   -3   -2    1    6  -0.0000129504  -0.0000014047
-3   -3   -2    1    7  -0.0000189169   0.0000024697
-3   -3   -2    1    8   0.0000238115   0.0000014316
-3   -3   -2    1    9   0.0000036708  -0.0000003266
-3   -3   -2    1   10   0.0000361752   0.0000003247
-3   -3   -2    2    1  -0.0000071722   0.0000018600
-3   -3   -2    2    2   0.0000105028  -0.0000000000
...
```

File format

- Line 1: Header
- Line 2: `[Norbit]`
- Line 3: `[Npts]`
- Lines 4 - $\lceil N_{\text{pts}}/15 \rceil + 3$:

  `[n1] [n2] ...`
- Line $\lceil N_{\text{pts}}/15 \rceil + 4$ onwards:

  `[rx] [ry] [rz] [alpha] [beta] [J.real] [J.imag]`

Parameters

- `[Norbit]`

  Type : Integer

  Description : This parameter specifies the number of orbitals N_{orbit} in a unit cell.
- `[Npts]`

  Type : Integer

  Description : This parameter specifies the number of cells in a rectangular cuboid that accommodates entire translation vectors.
- `[n1]`, `[n2]`, ...

  Type : Integer

  Description : These parameters specify the multiplicity of cells (ordinary 1), with 15 points in a line.
- `[rx]`, `[ry]`, `[rz]`

  Type : Integer

  Description : These parameters specify the translation vector.
- `[alpha]`, `[beta]`

  Type : Integer

Description : These parameters specify the indices of the orbitals. `[alpha]` corresponds to the orbital α in the original cell, and `[beta]` corresponds to the orbital β in the cell displaced by $\vec{r}$.

- `[J.real]`, `[J.imag]`

  Type : Float

  Description : These parameters specify the real and imaginary parts of the coefficient $J_{\alpha\beta}(\vec{r})$.

Usage rules

- Header cannot be omitted.
- The unspecified elements of the coefficient matrix are assumed to be zero.
- The translation vectors need to be enclosed within the CellShape. If the range of `r_x`, `r_y`, or `r_z` exceeds the extent of `x`, `y`, or `z` dimension of CellShape, the program terminates with an error.
- When `mode.enable_spin_orbital` is set to `true`, the orbital indices of Transfer term are interpreted as the extended orbital indices including spin degree of freedom that ranges from 1 to $2N_{\text{orbital}}$, in which the indices $1 \ldots N_{\text{orbital}}$ correspond to spin-up, and the indices $N_{\text{orbital}} + 1 \ldots 2N_{\text{orbital}}$ correspond to spin-down. Otherwise, only the entries with the orbital indices from 1 to N_{orbital} are taken into account.

5.2.3 Output files of RPA

This section describes the file formats of the outputs of RPA.

chiq and chi0q

The susceptibility matrix and the irreducible susceptibility matrix are exported in NumPy zip (npz) format. Using the string (referred to as *chiq_str*) specified by the keyword `chiq` or `chi0q` in `file.output` section in the parameter file, the filename is chosen as *chiq_str*`.npz`.

The file contains several arrays bound to the following keys:

- `chiq` or `chi0q`:

  The susceptibility matrix or the irreducible susceptibility matrix. Their data layout is described in the following sections.

- `freq_index`:

  The value or the range of Matsubara frequency is specified by `matsubara_frequency` parameter. The array bound to `freq_index` relates the index of the output data and the label of the actual Matsubara frequency.

- `wavevector_unit` and `wavevector_index`:

  These arrays refer to the information of the wave number vectors. See *Output files of UHFk* for details.

When the sublattice is considered, the indices of the wave numbers and the orbitals are regarded as those of the sublattice.

The output file of `chi0q` can be used as a pre-calculated input of the irreducible susceptibility by specifying the file to `chi0q_init` in `file.input` section.

Data format of chi0q

Data format of `chi0q` relies on the presence of spin-orbital interaction and external field, and the value of `mode.calc_scheme` parameter, and takes one of the following:

- "spin-free" case:

  If `enable_spin_orbital` parameter is set to `false`, or even if it is set to `true` when $T_{\tilde{\alpha}\tilde{\beta}}(k)$ is diagonal and symmetric with respect to spin degree of freedom, while the external field is not present, the spin-independent irreducible susceptibility matrix is exported.

  - When `calc_scheme = general`, the array format takes the form of `ndarray(l,q,a,ap,b,bp)` whose indices are given as follows:

    * `l`: label of Matsubara frequency. The map from the label to the index is provided by the aforementioned array `freq_index`.

    * `q`: linearlized index of wave-number indices $[q_x\ q_y\ q_z]$, where $q = q_z + N_z \cdot (q_y + N_y \cdot q_x)$.

    * `a`, `ap`, `b`, `bp`: indices of the orbitals not including spin degree of freedom. They correspond to α, α', β, β'.

  - When `calc_scheme = reduced` or `squashed`, the array format takes the form of `ndarray(l,q,a,b)` whose indices are same as the above.

- "spin-diagonal" case:

  If `enable_spin_orbital` parameter is set to `false` and the external field is present, or it is set to `true` while $T_{\tilde{\alpha}\tilde{\beta}}(k)$ is diagonal with respect to spin degree of freedom, the spin-up and spin-down components of the irreducible susceptibility matrix are exported.

  - When `calc_scheme = general`, the array format takes the form of `ndarray(s,l,q,a,ap,b,bp)`, where `s = 0` denotes spin-up component and `s = 1` does spin-down component. The other indices are same as the above.

  - When `calc_scheme = reduced` or `squashed`, the array format takes the form of `ndarray(s,l,q,a,b)`. The indices are same as above.

- "spinful" case:

  If `enable_spin_orbital` parameter is set to `true`, and $T_{\tilde{\alpha}\tilde{\beta}}(k)$ takes a general form, the irreducible susceptibility matrix with the generalized orbital indices is exported.

  - When `calc_scheme = general`, the array format takes the form of `ndarray(l,q,a,ap,b,bp)`, where `a`, `ap`, `b`, and `bp` corresponding to the generalized orbital indices including spin degree of freedom denoted by $\tilde{\alpha}$, $\tilde{\alpha}'$, $\tilde{\beta}$, and $\tilde{\beta}'$, respectively.

  - When `calc_scheme = reduced`, the array format takes the form of `ndarray(l,q,a,b)`, where `a` and `b` corresponding to the generalized orbital indices $\tilde{\alpha}$ and $\tilde{\beta}$, respectively.

Data format of chiq

Data format of `chiq` takes the following form depending on the value of `calc_scheme` parameter:

- When `calc_scheme = general`, the array format takes the form of `ndarray(l,q,a,ap,b,bp)`, where `a`, `ap`, `b`, and `bp` correspond to the generalized orbital indices including spin degree of freedom denoted by $\tilde{\alpha}$, $\tilde{\alpha}'$, $\tilde{\beta}$, and $\tilde{\beta}'$, respectively.

- When `calc_scheme = reduced`, the array format takes the form of `ndarray(l,q,a,b)`, where `a` and `b` correspond to the generalized orbital indices $\tilde{\alpha}$ and $\tilde{\beta}$, respectively.

- When `calc_scheme = squashed`, the array format takes the form of `ndarray(l,q,s1,s2,a,s3,s4,b)`, where `a` and `b` correspond to the orbital indices α and β, respectively, and `s1`, `s2`, `s3`, `s4` denote spin indices σ, σ', σ_1, σ_1', respectively. See *Algorithm* section for the notation.

Example for reading data

The following code is an example for reading the data from the output file.

```
import numpy as np
data = np.load("chiq_str.npz")

chiq = data["chiq"]
freq_index = data["freq_index"]
```

CHAPTER

SIX

ALGORITHMS

6.1 Unrestricted Hartree-Fock method

6.1.1 Overview

The unrestricted Hartree-Fock approximation is a method to approximate two-body interactions into one-body terms by taking account of the fluctuation of the one-body operators up to first order. For a general two-body interactions, it leads to the following approximation:

$$\begin{aligned} c_i^\dagger c_j^\dagger c_k c_l \sim & \langle c_i^\dagger c_l\rangle c_j^\dagger c_k + c_i^\dagger c_l \langle c_j^\dagger c_k\rangle - \langle c_i^\dagger c_k\rangle c_j^\dagger c_l - c_i^\dagger c_k \langle c_j^\dagger c_l\rangle \\ & - (\langle c_i^\dagger c_l\rangle \langle c_j^\dagger c_k\rangle - \langle c_i^\dagger c_k\rangle \langle c_j^\dagger c_l\rangle). \end{aligned}$$

In H-wave, the two-body interaction terms are defined as

$$\begin{aligned} \mathcal{H}_{\text{InterAll}} &= \sum_{ijkl\alpha\beta\gamma\delta} \sum_{\sigma_1\sigma_2\sigma_3\sigma_4} I_{ijkl\alpha\beta\gamma\delta} c_{i\alpha\sigma_1}^\dagger c_{j\beta\sigma_2} c_{k\gamma\sigma_3}^\dagger c_{l\delta\sigma_4} \\ &= \sum_{ijkl\alpha\beta\gamma\delta} \sum_{\sigma_1\sigma_2\sigma_3\sigma_4} I_{ijkl\alpha\beta\gamma\delta} (c_{i\alpha\sigma_1}^\dagger c_{k\gamma\sigma_3}^\dagger c_{l\gamma\sigma_4} c_{j\beta\sigma_2} + c_{i\alpha\sigma_1}^\dagger c_{l\delta\sigma_4} \delta_{j,k}\delta_{\beta,\gamma}\delta_{\sigma_2,\sigma_3}). \end{aligned}$$

It is noted that there is a one-body term as depicted in the second term of the above expression. Then, the Hamiltonian given by the one-body terms is generally denoted as

$$\mathcal{H}_{\text{UHF}} = \sum_{ij} H_{ij} c_i^\dagger c_j = \hat{c}^\dagger H \hat{c} \tag{6.1}$$

where we adopt a notation $i \equiv (i, \alpha, \sigma_1), j \equiv (j, \beta, \sigma_2)$ for brevity, H denotes a matrix whose elements are H_{ij}, and $\hat{c}$ denotes a column vector whose elements are c_i.

As H is an Hermite matrix, the Hamiltonian can be transformed into $H = U\hat{\xi}U^\dagger$ where $\hat{\xi}$ is a matrix whose diagonal elements are the eigenvalues of H, and U is a matrix composed of the corresponding eigenvectors.

Then, let $\hat{d} = U^\dagger \hat{c}$, and $\mathcal{H}_{\text{UHF}}$ leads to

$$\mathcal{H}_{\text{UHF}} = \hat{d}^\dagger \hat{\xi} \hat{d} = \sum_k \xi_k d_k^\dagger d_k. \tag{6.2}$$

Therefore, the energy derived from the one-body interaction term of the UHF approximation is obtained by

$$E_{\text{UHF}} = \langle \mathcal{H}_{\text{UHF}} \rangle = \sum_k \xi_k \langle d_k^\dagger d_k \rangle. \tag{6.3}$$

In the numerical calculation, as H depends on the one-body Green's function $\langle c_i^\dagger c_j\rangle$ through the UHF approximation, the equation is iteratively solved to satisfy the self-consistency. Starting from a one-body Green's function given as an initial value, it is updated through the relation

$$\langle c_i^\dagger c_j \rangle = \sum_l U_{il}^* U_{jl} \langle d_l^\dagger d_l \rangle = \sum_l \frac{U_{il}^* U_{jl}}{1 + \exp^{\beta(\xi_l - \mu)}} \tag{6.4}$$

until the one-body Green's function converges. Here, β denotes the inverse temperature $1/k_B T$, and μ denotes the chemical potential. In the canonical calculation in which the number of particles is fixed, μ is determined to satisfy the relation

$$N = \sum_i \langle c_i^\dagger c_i \rangle \tag{6.5}$$

for the number of particles N at every step.

In H-wave, the simple-mixing algorithm is employed to update the configuration. If we denote the one-body Green's function at *n*-th step by $\langle c_i^\dagger c_j \rangle^{(n)}$, the Green's function at *n+1*-th step is chosen by mixing that of *n*-th step with the new one obtained in *n+1*-th step as

$$\langle c_i^\dagger c_j \rangle^{(n+1)} := (1-\alpha) \langle c_i^\dagger c_j \rangle^{(n)} + \alpha \langle c_i^\dagger c_j \rangle^{(n+1)}, \tag{6.6}$$

where α is a parameter between 0 and 1. There are other update algorithms such as Anderson mixing, though they are not supported in the present version of H-wave.

In the coordinate-space UHF mode of H-wave, all interactions are mapped to InterAll form. The free energy at finite temperature is given by

$$F = \mu N - \frac{1}{\beta} \sum_k \ln\left[1 + \exp(-\beta(\xi_k - \mu))\right] - \sum_{ijkl} I_{ijkl} (\langle c_i^\dagger c_j \rangle \langle c_k^\dagger c_l \rangle - \langle c_i^\dagger c_l \rangle \langle c_k^\dagger c_j \rangle). \tag{6.7}$$

6.1.2 Extension to wave-number space

The Hamiltonian given by the one-body terms is rewritten in the wave-number representation by the Fourier transform $c_i = \frac{1}{\sqrt{V}} \sum_k e^{ikr_i} c_k$ as

$$\mathcal{H}_{\mathrm{UHF}} = \sum_{k\alpha\beta\sigma\sigma'} h_{\alpha\beta\sigma\sigma'}(k)\, c_{k\alpha\sigma}^\dagger c_{k\beta\sigma'} \tag{6.8}$$

Here, the interaction is assumed to have translational symmetry so that the coefficients depend only on the translation vectors $r_{ij} = r_j - r_i$. It is noted that InterAll type of interaction is not considered in the wave-number space UHF mode.

As the Hamiltonian is diagonal with respect to the wave number k, the calculation of the eigenvalues and eigenvectors reduces from diagonalization of a matrix of the size $N_{\mathrm{site}} N_{\mathrm{orbit}} \times N_{\mathrm{site}} N_{\mathrm{orbit}}$ to that of N_{site} matrices of the size $N_{\mathrm{orbit}} \times N_{\mathrm{orbit}}$, which lowers the calculation costs. Here, N_{site} denotes the number of sites, and N_{orbit} denotes the number of orbitals including the spin degree of freedom.

6.2 Random Phase Approximation

The random phase approximation (RPA) is a method to detect the response to the fluctuations of one-body operators by the effect of electron correlations, starting from the non-interacting state. While in the UHF approximation an initial guess of the configuration is required, the RPA method enables to infer the ordered phase that emerges from the second-order transition. H-wave implements RPA method using Matsubara frequency, and allows to compare with the dynamical observables measured in the experiments by analytical continuation.

In the following, the algorithm is described. In the RPA mode of H-wave, the Hamiltonian given below will be considered:

$$\begin{aligned}
\mathcal{H} &= \mathcal{H}_0 + \mathcal{H}_{\mathrm{int}}, \\
\mathcal{H}_0 &= \sum_{\langle i\alpha;j\beta\rangle} (t_{ij}^{\alpha\beta} c_{i\alpha}^\dagger c_{j\beta} + \mathrm{H.c.}), \\
\mathcal{H}_{\mathrm{int}} &= \sum_{ij} \sum_{\alpha,\alpha',\beta,\beta'} W_{ij}^{\beta\beta',\alpha\alpha'} \left(c_{i\alpha}^\dagger c_{i\alpha'} c_{j\beta'}^\dagger c_{j\beta} + \mathrm{H.c.} \right)
\end{aligned} \tag{6.9}$$

Applying the Fourier transformation

$$c_{i\alpha} = \frac{1}{\sqrt{N_L}} \sum_{\mathbf{k}} e^{i\mathbf{k}\cdot\mathbf{r_i}} c_{\mathbf{k},\alpha}, \tag{6.10}$$

the Hamiltonian is rewritten in the following form

$$\begin{aligned}
\mathcal{H} = &\sum_{\mathbf{k}\alpha\beta} (\varepsilon_{\alpha\beta}(\mathbf{k}) c_{\mathbf{k}\alpha}^\dagger c_{\mathbf{k}\beta} + \mathrm{H.c.}) \\
&+ \frac{1}{2N_L} \sum_{\mathbf{k}\mathbf{k}'\mathbf{q}} \sum_{\alpha\beta\alpha'\beta'} W_{\mathbf{q}}^{\beta\beta',\alpha\alpha'} c_{\mathbf{k}+\mathbf{q},\alpha}^\dagger c_{\mathbf{k},\alpha'} c_{\mathbf{k}'-\mathbf{q},\beta'}^\dagger c_{\mathbf{k}',\beta}
\end{aligned}$$

In the random phase approximation, the density fluctuation by the effect of electron correlation is detected with respect to $\mathcal{H}_0$. The scattering by the interaction must therefore be considered on the basis where $\mathcal{H}_0$ is diagonalize, and thus the interaction term is approximated as

$$\begin{aligned}
&W_{\mathbf{q}}^{\beta\beta',\alpha\alpha'} c_{\mathbf{k}+\mathbf{q},\alpha}^\dagger c_{\mathbf{k},\alpha'} c_{\mathbf{k}'-\mathbf{q},\beta'}^\dagger c_{\mathbf{k}',\beta} \\
&\sim W_{\mathbf{q}}^{\beta\beta',\alpha\alpha'} \sum_{\gamma,\gamma'} u_{\alpha\gamma,\mathbf{k}+\mathbf{q}}^* d_{\mathbf{k}+\mathbf{q},\gamma}^\dagger u_{\alpha'\gamma,\mathbf{k}} d_{\mathbf{k},\gamma} u_{\beta'\gamma',\mathbf{k}'-\mathbf{q}}^* d_{\mathbf{k}'-\mathbf{q},\gamma'}^\dagger u_{\beta\gamma',\mathbf{k}'} d_{\mathbf{k}',\gamma'}.
\end{aligned}$$

Here,

$$c_{\mathbf{k},\alpha} = \sum_{\gamma} u_{\alpha\gamma,\mathbf{k}} d_{\mathbf{k},\gamma} \tag{6.11}$$

and $d_{\mathbf{k},\gamma}$ denotes the annihilation operator that diagonalizes $\mathcal{H}_0$. (γ refers to the index of the eigenvalue.) Then, the irreducible one-body Green's function is written as

$$G_{\gamma}^{(0)\alpha\beta}(\mathbf{k}, i\omega_n) = \frac{u^{\alpha\gamma}(\mathbf{k}) u^{*\beta\gamma}(\mathbf{k})}{i\epsilon_n - \xi^\gamma(\mathbf{k}) + \mu}. \tag{6.12}$$

The irreducible susceptibility is given in the following form, as it must be closed within the diagnoalized elements:

$$X^{(0)\alpha\alpha',\beta\beta'}(\mathbf{q}, i\omega_n) = -\frac{T}{N_L} \sum_{\gamma=1}^{n_{\mathrm{orb}}} \sum_{\mathbf{k},n} G_{\gamma}^{(0)\alpha\beta}(\mathbf{k}+\mathbf{q}, i\omega_m + i\epsilon_n) G_{\gamma}^{(0)\beta'\alpha'}(\mathbf{k}, i\epsilon_n), \tag{6.13}$$

By using the irreducible susceptibility, the susceptibility matrix from the RPA is obtained as follows:

$$X^{\alpha\alpha',\beta\beta'}(q) = X^{(0)\alpha\alpha',\beta\beta'}(q) - \sum_{\alpha_1'\beta_1'} X^{(0)\alpha\alpha',\beta_1\beta_1'}(q) W_{\mathbf{q}}^{\beta_1\beta_1',\alpha_1\alpha_1'} X^{\alpha_1\alpha_1',\beta\beta'}(q), \tag{6.14}$$

Combining indices such as $\alpha\alpha'$ into one index, they are expressed in the matrix form. Then finally it leads to the expression:

$$\begin{aligned}
\hat{X}(q) &= \hat{X}^{(0)}(q) - \hat{X}^{(0)}(q)\hat{W}(q)\hat{X}(q) \\
&= \left[\hat{I} + \hat{X}^{(0)}(q)\hat{W}(q)\right]^{-1} \hat{X}^{(0)}(q).
\end{aligned}$$

In the above formula, orbitals and spins were treated as unified generalised orbitals. Of the arrays needed to perform the calculations, the susceptibility ($X^{(0)\alpha\alpha',\beta\beta'}(\mathbf{q}, i\omega_n)$, $X^{\alpha\alpha',\beta\beta'}(\mathbf{q}, i\omega_n)$) is the largest multidimensional array, given by $N_{\rm orb}^4 N_{\rm spin}^4 N_k N_\omega$, where the memory cost and computational complexity increase as the size increases. As explained below, the size of the multidimensional array of susceptibilities can be reduced by separating orbits and spins: for the two-body interactions handled in H-wave's RPA mode, separating orbits and spins results in

$$W_{\mathbf{q}}^{\beta\sigma_1\sigma_1',\alpha\sigma\sigma'} c_{\mathbf{k}+\mathbf{q},\alpha\sigma}^\dagger c_{\mathbf{k},\alpha\sigma'} c_{\mathbf{k}'-\mathbf{q},\beta\sigma_1'}^\dagger c_{\mathbf{k}',\beta\sigma_1}. \tag{6.15}$$

Since the scattering is on the same diagonalized general orbital, the irreducible susceptibility becomes

$$X_{\sigma\sigma'\sigma_1\sigma_1'}^{(0)\alpha,\beta}(\mathbf{q}, i\omega_n) = -\frac{T}{N_L}\sum_{\gamma=1}^{n_{\rm orb}}\sum_{\mathbf{k},n} G_{\sigma\sigma_1',\gamma}^{(0)\alpha\beta}(\mathbf{k}+\mathbf{q}, i\omega_m + i\epsilon_n) G_{\sigma_1\sigma',\gamma}^{(0)\beta\alpha}(\mathbf{k}, i\epsilon_n). \tag{6.16}$$

The array size can be reduced to $N_{\rm orb}^2 N_{\rm spin}^4 N_k N_\omega$. Then susceptibility matrix by RPA is obtained as follows:

$$X_{\sigma\sigma'\sigma_1\sigma_1'}^{\alpha,\beta}(q) = X_{\sigma\sigma'\sigma_1\sigma_1'}^{(0)\alpha,\beta}(q) - \sum_{\alpha_1'\beta_1'} X_{\sigma\sigma'\sigma_2\sigma_2'}^{(0)\alpha,\alpha_2}(q) W_{\sigma_2\sigma_2',\sigma_3\sigma_3'}^{\alpha_2,\alpha_3}(\mathbf{q}) X_{\sigma_3\sigma_3',\sigma_1\sigma_1'}^{\alpha_3,\beta}(q). \tag{6.17}$$

If $\alpha\sigma\sigma'$ is regarded as a single index, it can be put into matrix form and, as in the case of generalised orbitals, can be used as a

$$\begin{aligned}\hat{X}(q) &= \hat{X}^{(0)}(q) - \hat{X}^{(0)}(q)\hat{W}(q)\hat{X}(q)\\ &= \left[\hat{I} + \hat{X}^{(0)}(q)\hat{W}(q)\right]^{-1}\hat{X}^{(0)}(q).\end{aligned}$$

The above formula is the general formula for the RPA method.

In the above formula, the calculation of the irreducible susceptibility is performed as follows:

$$X_{\sigma\sigma'\sigma_1\sigma_1'}^{(0)\alpha,\beta}(\mathbf{q}, i\omega_n) = -\frac{T}{N_L}\sum_{\gamma=1}^{n_{\rm orb}}\sum_{\mathbf{k},n} G_{\sigma\sigma_1',\gamma}^{(0)\alpha\beta}(\mathbf{k}+\mathbf{q}, i\omega_m + i\epsilon_n) G_{\sigma_1\sigma',\gamma}^{(0)\beta\alpha}(\mathbf{k}, i\epsilon_n)$$

In this case, the sum of the diagonalized components is required, which is computationally more expensive. In many previous studies, the one body Green's function is calculated as follows:

$$G_{\sigma\sigma'}^{(0)\alpha\beta}(\mathbf{k}, i\omega_n) = \sum_{\gamma=1}^{n_{\rm orb}} G_{\sigma\sigma',\gamma}^{(0)\alpha\beta}(\mathbf{k}, i\omega_n). \tag{6.18}$$

The irreducible susceptibility is calculated as follows:

$$X_{\sigma\sigma'\sigma_1\sigma_1'}^{(0)\alpha,\beta}(\mathbf{q}, i\omega_n) = -\frac{T}{N_L}\sum_{\mathbf{k},n} G_{\sigma\sigma_1'}^{(0)\alpha\beta}(\mathbf{k}+\mathbf{q}, i\omega_m + i\epsilon_n) G_{\sigma_1\sigma'}^{(0)\beta\alpha}(\mathbf{k}, i\epsilon_n).$$

Though this method may lead to poor accuracy when the diagonalized components are mixed, there is an advantage that there is no need for technical consideration for γ due to band intersections. In order to make comparisons with previous studies, H-Wave has adopted this approach (a mode for correctly handling the Green's functions and susceptibilities will also be implemented). It is noted that the vertex correction may be taken into account as a means to consider higher order correlations. See, for example, reference[1] for the details.

[1] K. Yoshimi, T. Kato, H. Maebashi, J. Phys. Soc. Jpn. 78, 104002 (2009).

CHAPTER

SEVEN

ACKNOWLEDGMENTS

H-wave was developed with the support of "Project for advancement of software usability in materials science" (FY2022) of The Institute for Solid State Physics, The University of Tokyo.

APPENDIX

A.1 Generation of interaction files using StdFace library

A.1.1 Compile StdFace library

The interaction definition files can be generated easily using StdFace library. We will provide a short instruction how to use it.

The source package of StdFace library that supports input formats of the Hwave is available from the repository as follows.

```
$ git clone https://github.com/issp-center-dev/StdFace.git
```

Then, the library is to be compiled with the commands:

```
$ cd StdFace
$ mkdir build && cd build
$ cmake -DHWAVE=ON ..
$ make
```

If the compilation is successful, you can find the executable module `hwave_dry.out` in `src` directory.

An input to `hwave_dry.out` can be found as `stan.in` in the sample directory, which reads:

```
model = "Hubbard"
lattice = "square"
W = 4
L = 4
t = 1.0
t' = 0.5
U = 4.0
V = 1.0
Ncond = 16
eps = 12
calcmode = "uhfk"
exportall = 0
```

- `model` is a keyword to choose the target model. Currently, only `Hubbard` is supported that denotes Hubbard model with the number of electrons fixed.
- `lattice` is a keyword to specify the lattice structure. In this example, the square lattice `square` is chosen. `W` and `L` denote the size of the lattice.

- `t` and `V` denote parameters of the hopping and the neighbor-site Coulomb interaction, respectively.
- `calcmode = "uhfk"` and `calcmode = "rpa"` specify the output to be in the Wannier90(-like) format. `calcmode = "uhfr"` specifies the output for input files of UHFr. The default is `calcmode = "uhfk"`. If `exportall = 0` is given, the outputs are compactified with zero components omitted.

See Section *Input files for UHFr* , *Input files for UHFk* , *Input files for RPA* for the details of input files.

A.1.2 Run StdFace library

Then, run `hwave_dry.out` with the file above as an input:

```
$ cd path_to_Hwave/docs/tutorial/Hubbard/RPA
$ ln -s path_to_Stdface/build/src/hwave_dry.out .
$ ./uhf_dry.out stan.in
```

When the program finishes, a geometry information file `geom.dat` and interaction definition files `transfer.dat` and `coulombinter.dat`, are generated in the current directory.

A.2 List of error messages

- `mode is not defined in [mode].`

  description : `mode` parameter is missing in `[mode]` section of the input parameter file.

  mode : main

- `Get_param:  key must be mod or ham or output.`

  description : unsupported keyword is given to `get_param()`.

  mode : UHFr (read_input)

- `duplicate items found in` *file*

  description : the file *file* contains duplicate entries.

  mode : UHFr (read_input)

- `incorrect number of lines in` *file* `:  expected=` *N* `, found=` *M*

  description : number of lines of the input file does not match the description in the file.

  mode : UHFr (read_input)

- `Unknown keyword` *keyword*

  description : unsupported keyword is found in `[file.input.interaction]`.

  mode : UHFk (read_input_k)

- `initial and initial_uhf can not be specified simultaneously.`

  description : `initial` and `initial_uhf` cannot be specified simultaneously.

  mode : UHFk (read_input_k)

- `read_input_k:  file` *file* `not found`

  description : the file *file* cannot be found.

  mode : UHFk (read_input_k)

- `Get_param: key must be mod or ham or output.`

  description : unsupported keyword is given to `get_param()`.

  mode : UHFk (read_input_k)

- `read_geom: file` *file* `not found`

  description : the file *file* specified by `Geometry` keyword cannot be found.

  mode : UHFk (wan90)

- `mode.param.2Sz must be even(odd) when Ncond is even(odd).`

  description : even/odd mismatch between `2Sz` and `Ncond`.

  mode : solver base

- `range check for` *type* `failed.`

  description : the value of *type* is not appropriate.

  mode : UHFk

- `_check_cellsize failed. interaction range exceeds cell shape.`

  description : some of translation vectors of the interaction description do not lie within the CellShape.

  mode : UHFk

- `Hermiticity check failed: |T_ba(-r)^* - T_ab(r)| =` *val*

  description : Transfer term is not Hermite.

  mode : UHFk

- `Parameter range check failed for param_mod.`

  description : the parameter value in [mode.param] is out of range.

  mode : solver base

- `Parameter check failed for param_mod.`

  description : the parameter value in [mode.param] is inappropriate.

  mode : solver base

- `Hermite check failed for` *type*

  description : *type* is not Hermite.

  mode : UHFr

- `Parameter check failed for info_mode.`

  description : the parameter value in [mode] is inappropriate.

  mode : solver base

- `value not integer`

  description : the parameter value is not an integer.

  mode : RPA

- `Lattice initialization failed: 'CellShape' not found.`

  description : `CellShape` is missing in [mode.param].

  mode : RPA

- `Ncond must be greater than zero: Ncond=` *Ncond*

  description : the value of `Ncond` is not appropriate.

  mode : RPA

- `Nmat must be greater than zero: Nmat=` *Nmat*

  description : the value of `Nmat` is not appropriate.

  mode : RPA

- `RPA._find_mu: not converged. abort`

  description : the calculation of `mu` does not converge.

  mode : RPA

- `SubShape is not compatible with CellShape.`

  description : the value of SubShape does not divide that of CellShape.

  mode : RPA

- `T must be greater than or equal to zero: T=` *T*

  description : the value of `T` is not appropriate.

  mode : RPA

- `both mu and Ncond or filling are specified`

  description : `` mu` and ``Ncond `` or `filling` should not specified simultaneously.

  mode : RPA

- `dimension of CellShape must be one, two, or three.`

  description : the dimension of `CellShape` is not appropriate.

  mode : RPA

- `dimension of SubShape does not match with that of CellShape.`

  description : the dimension of `SubShape` is not appropriate.

  mode : RPA

- `invalid CellShape.`

  description : the value of `CellShape` is not appropriate.

  mode : RPA

- `invalid SubShape.`

  description : the value of `SubShape` is not appropriate.

  mode : RPA

- `none of mu, Ncond, nor filling is specified`

  description : one of `mu`, `Ncond`, or `filling` should be specified.

  mode : RPA

- `read_chi0q failed:` *info*

  description : reading `chi0q` from file was not successful.

  mode : RPA

- `round_to_int: unknown mode` *mode*

  description : unsupported rounding mode is specified.

  mode : RPA

- `unexpected data size` *error*

  description : data size is not as expected.

  mode : RPA

- `mode is not defined in [mode].`

  description : the `mode` parameter is missing in `[mode]`.

  mode : RPA

- `orbital index check failed for` *type*

  description : the indices of the orbitals are inappropriate.

  mode : UHFk

- `initial green function in coord space requires geometry.dat`

  description : `geometry.dat` must also be specified when the coordinate space Green's function.

  mode : UHFk

- `CellShape is missing. abort`

  description : `CellShape` parameter is missing.

  mode : UHFk

- `Ncond or Nelec is missing. abort`

  description : `Ncond` or `Nelec` parameter is missing.

  mode : UHFk

- `SubShape is not compatible with CellShape. abort`

  description : the value of `SubShape` does not divide that of `CellShape`.

  mode : UHFk

- `_check_orbital_index failed. invalid orbital index found in interaction definitions.`

  description : the indices of the orbitals in interaction definition files are inappropriate.

  mode : UHFk

- `_save_greenone: onebodyg_uhf and geometry_uhf are required`

  description : `onebodyg_uhf` and `geometry_uhf` are not provided.

  mode : UHFk

- `find mu: not converged. abort`

  description : the calculation of `mu` does not converge.

  mode : UHFk

- `range check failed for Initial`

  description : the values of `Initial` are inappropriate.

  mode : UHFr

- `OneBodyG is required to output green function.`

  description : `OneBodyG` is missing for the output of Green's function.

  mode : UHFr

- `hermite check failed for Initial`

  description : `Initial` is not Hermite.

  mode : UHFr

- `Range check failed for Transfer`

  description : the indices of `Transfer` definition file are out of range.

  mode : UHFr

- `Range check failed for` *type*

  description : the indices of *type* definition file are out of range.

  mode : UHFr

- `parameter range check failed.`

  description : the value of the parameter is not appropriate.

  mode : UHFr

- `mode is incorrect:  mode=` *mode*

  description : `mode` parameter is not appropriate.

  mode : UHFr

- `mode.param.` *key* `must be greater than` *value*

  description : the value of parameter *key* in [mode.param] is inappropriate.

  mode : solver base [warning]

- `"mode.param.` *key* `must be smaller than` *value*

  description : the value of parameter *key* is [mode.param] is inappropriate.

  mode : solver base [warning]

- `mode.param.` *key* `is not defined.`

  description : parameter *key* is not found in [mode.param].

  mode : solver base [warning]

- `mode.` *key* `in mode section is incorrect:` *values*

  description : `mode` parameter in `[mode]` section is not valid.

  mode : solver base [warning]

- `mode.` *key* `is not defined.`

  description : `mode` parameter is not found in `[mode]` section.

  mode : solver base [warning]

- `TRUST-ME mode enabled. parameter checks are relaxed`

  description : `TRUST-ME` mode is enabled. the parameter checks will be omitted.

  mode : solver base [warning]

- `value not integer`

  description : the specified value is not an integer.

  mode : RPA [warning]

- `mode is incorrect:  mode=` *mode*

  description : `mode` parameter is not valid.

  mode : RPA [warning]

- `FATAL: 2Sz=` *value* `. 2Sz should be even for calculating fij`

  description : `2Sz` must be an even number for the calculation of f_{ij}.

  mode : UHFr [warning]

- `FATAL: Ne=` *value* `. Ne should be even for calculating fij`

  description : `Ne` must be an even number for the calculation of f_{ij}.

  mode : UHFr [warning]

- `NOT IMPLEMENTED: Sz even and Sz != 0:  this case will be implemented in near future`

  description : the calculation of f_{ij} is not yet supported when `Sz` is an even number except zero.

  mode : UHFr [warning]

- `key` *key* `is wrong!`

  description : the keyword *key* is invalid.

  mode : UHFr [warning]

- `UHFr calculation is failed:  rest=` *residue* `, eps=` *eps*

  description : the calculation of UHFr does not converge.

  mode : UHFr [warning]