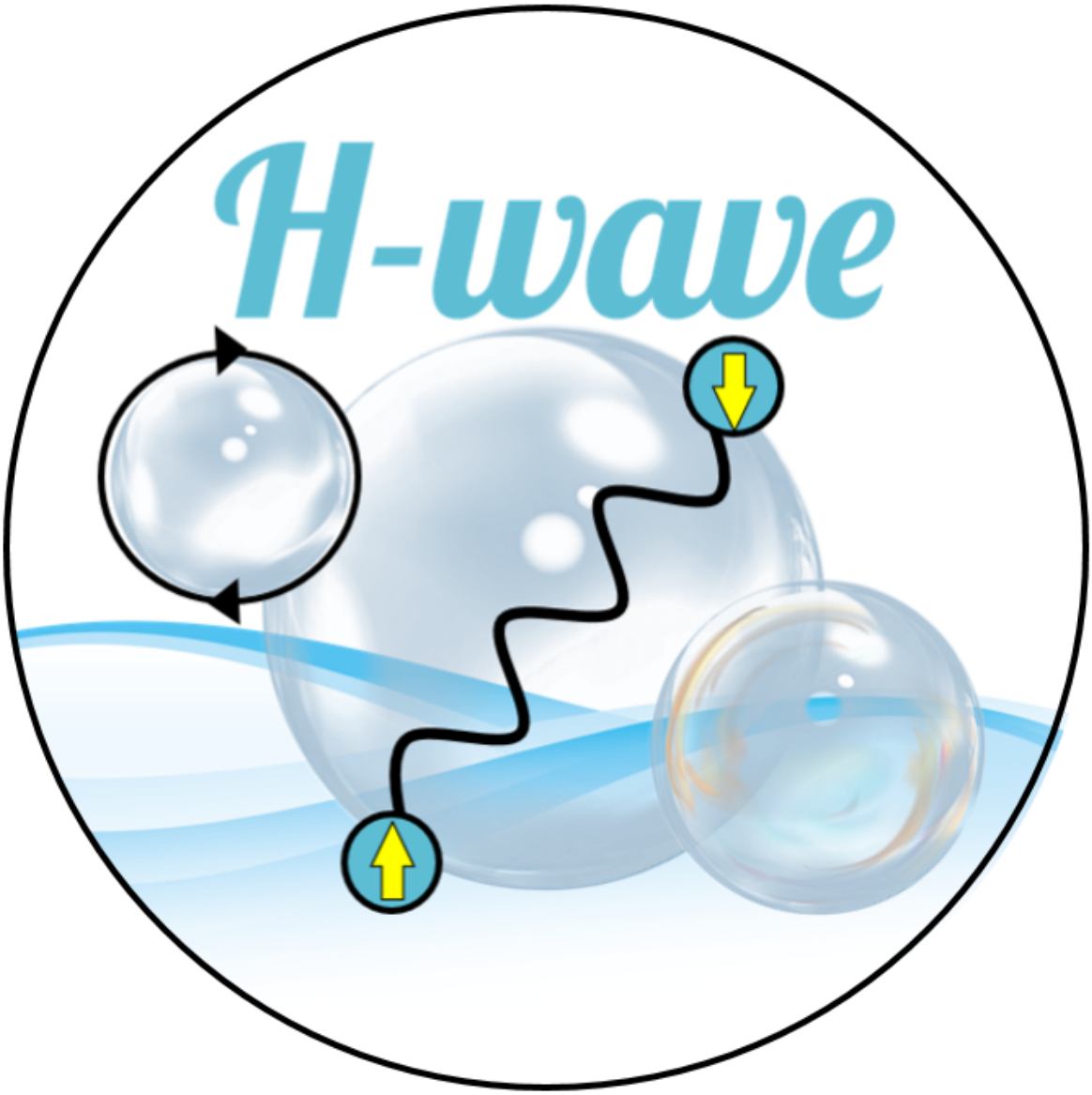

H-wave Documentation

リリース *1.0.1*

ISSP, University of Tokyo

2025 年 01 月 31 日

目 次

第1章　H-waveとは？

1.1 はじめに

H-wave は遍歴電子系を対象に非制限 Hartree-Fock(UHF) 近似計算および乱雑位相近似 (RPA) 計算を行うためのプログラムです。平均場近似をベースとした近似計算であるため、計算コストが軽く、複雑なハミルトニアンや大きいサイズでの計算が可能です。

1.2 ライセンス

1.3 開発貢献者

本ソフトウェアは以下の開発貢献者により開発されています。

- ver.1.0.0 (2023/04/25 リリース)
  - 開発者
    - 吉見 一慶 (東京大学 物性研究所)
    - 青山 龍美 (東京大学 物性研究所)
    - 本山 裕一 (東京大学 物性研究所)
    - 三澤 貴宏 (東京大学 物性研究所)
    - 井戸 康太 (東京大学 物性研究所)
    - 小林 晃人 (名古屋大学 理学研究科)
    - 川村 泰喜 (名古屋大学 理学研究科)
  - プロジェクトコーディネーター
    - 加藤 岳生 (東京大学 物性研究所)

1.4 コピーライト

本ソフトウェアは 2022 年度 東京大学物性研究所 ソフトウェア高度化プロジェクトの支援を受け開発されており、その著作権は東京大学が所持しています。

1.5 動作環境

以下の環境で動作することを確認しています。

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

1.6 参考文献

"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]).

第2章　基本的な使用方法

- 必要なライブラリ・環境

  H-wave を利用するには、以下のプログラムとライブラリが必要です。

  - python 3.x
  - numpy モジュール
  - scipy モジュール
  - requests モジュール
  - tomli モジュール

  なお、H-wave の UHFk と RPA モードでは numpy.fft を FFT 計算に利用しています。

- Official Page
  - GitHub リポジトリ
  - サンプル・チュートリアル
- ダウンロード方法

  git を利用できる場合は、以下のコマンドで H-wave をダウンロードすることができます。

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

- インストール方法
  - PyPI から

    H-wave は PyPI ソフトウェアリポジトリに登録されています。以下のコマンドで H-wave をインストールできます。

```
$ pip install hwave
```

  - ソースパッケージから

    H-wave のソースパッケージは配布サイトから取得できます。

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

    また、git を用いて最新版を開発サイトからダウンロードできます。

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

H-wave をダウンロード後、以下のコマンドを実行してインストールします。H-wave が利用するライブラリも必要に応じてインストールされます。

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

- ディレクトリ構成

```
.
|-- 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/
```

- 基本的な使用方法

1. 入力ファイルの作成

   最初に H-wave 用の入力ファイルを作成します。計算条件や入出力ファイル・ディレクトリなどの指定と、Hamiltonian の定義ファイルなどを作成する必要があります。後者は、StdFace ライブラリ の利用が便利です。各ファイルの簡単な紹介はチュートリアルの章に記載されています。詳細についてはファイルフォーマットの章を参照してください。

2. コマンドの実行

   入力ファイルのあるディレクトリで、以下のコマンドを実行することで、計算が行われます。

```
$ hwave input.toml
```

または、

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

計算終了後、計算結果が出力ディレクトリに出力されます。出力ファイルの詳細については、ファイルフォーマットの章を参照してください。

第3章　実空間UHF(UHFr)

3.1 チュートリアル

UHFr では、入力ファイルとして

1. 環境設定入力ファイル
2. Hamiltonian 作成用ファイル
3. 出力結果指定用ファイル

を用意した後、計算を行います。以下では、 docs/tutorial/Hubbard/UHFr ディレクトリにあるサンプルを例にチュートリアルを実施します。なお、相互作用定義ファイルは StdFace ライブラリを用いて生成することもできます。詳細は *StdFace* **を用いた相互作用ファイルの作成** の章をご覧ください。

3.1.1 環境設定入力ファイルの作成

環境設定入力ファイルでは、入出力を制御する情報を記載します。docs/tutorial/Hubbard/UHFr ディレクトリ内に input.toml というファイルがありますが、これが環境設定入力ファイルになります。以下、ファイルの中身を記載します。

```
[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]
```

(次のページに続く)

(前のページからの続き)

```
Trans = "trans.def"
CoulombIntra = "coulombintra.def"
[file.output]
path_to_output = "output"
energy = "energy.dat"
eigen = "eigen"
green = "green.dat"
```

このファイルは toml 形式で記述します。

log セクションに print_level で標準出力のレベル、 print_step でログファイルに出力するステップ間隔を指定します。

mode セクションに実行モードおよび基本パラメータを指定します。

file.input セクションに入力ファイルが格納されているディレクトリ path_to_input 、出力したい一体グリーン関数が定義されたファイル OneBodyG 及び初期配置 Initial を指定します。OneBodyG を指定しない場合にはグリーン関数の出力がされません。また、Initial を指定しない場合には初期配置はランダムな配置が設定されます。

file.input.interaction セクションに Hamiltonian を作成するための入力ファイルを指定します。

file.output セクションには出力ファイルを格納するディレクトリ path_to_output を指定します。また、エネルギーの値を出力するファイル名 energy 、ハミルトニアンの固有値を出力するファイル名 eigen 、一体グリーン関数の出力ファイル名 green を指定します。これらのキーワードがない場合には情報は出力されません。

詳細については **ファイルフォーマット** の章をご覧ください。

3.1.2 Hamiltonian の指定

基本パラメータを設定した後は、Hamiltonian を構築するためのファイルを作成します。

Transfer 部の指定

Trans でひも付けられるファイル (ここでは trans.def) で電子系の Transfer に相当する Hamiltonian

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

を指定します。ファイルの中身は下記の通りです。

```
========================
NTransfer      64
========================
========i_j_s_tijs======
========================
```

(次のページに続く)

(前のページからの続き)

```
    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
...
```

Trans ファイルの詳細はセクション *Trans* **指定ファイル** をご覧ください。

二体相互作用部の指定

このチュートリアルの例では CoulombIntra でひも付けられるファイル (ここでは coulombintra.def) で電子系の二体相互作用部に相当する Hamiltonian

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

を指定します。ファイルの中身は下記の通りです。

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

なお、 CoulombIntra 以外にも、Hamiltonian を簡易的に記載するための各種ファイル形式に対応しています。詳細はセクション *InterAll* **指定ファイル** - *PairLift* **指定ファイル** をご覧ください。

3.1.3 出力ファイルの指定

一体 Green 関数の計算する成分を、OneBodyG でひも付けられるファイルで指定します。

一体 Green 関数の計算対象の指定

OneBodyG でひも付けられるファイル (ここでは greenone.def) で計算する一体 Green 関数 $\langle c^{\dagger}_{i\sigma_1} c_{j\sigma_2} \rangle$ の成分を指定します。ファイルの中身は下記の通りです

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

一体 Green 関数計算対象成分の指定に関するファイル入力形式の詳細はセクション *OneBodyG* **指定ファイル** をご覧ください。

3.1.4 計算の実行

全ての入力ファイルが準備できた後、計算実行します。環境設定入力ファイル (ここでは input.toml) を引数とし、ターミナルから H-wave を実行します。

```
$ hwave input.toml
```

計算が開始されると以下のようなログが出力されます。

```
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
```

(次のページに続く)

(前のページからの続き)

```
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
-------------------------------------------------------------------------------
```

入力ファイル読み込みに関するログが出力されたあと、UHF 計算の計算過程に関する情報が出力されます。出力ファイルは `input.toml` の `file.output` セクションでの設定にしたがい、`output` ディレクトリに 固有値が記載された `energy.dat` , 固有ベクトルが記載された `spin-down_eigen.npz`, `spin-up_eigen.npz`, 一体グリーン関数の値が記載された `green.dat` ファイルが出力されます。出力ファイルの詳細については **ファイルフォーマット** の章をご覧ください。

3.2 ファイルフォーマット

3.2.1 環境設定ファイル

このファイルでは、TOML 形式で H-wave に関する環境を設定します。本ファイルは以下の 3 つのセクションから構成されます。

1. `mode` セクション: 計算モードに関する設定を指定するセクション。
2. `log` セクション: 標準出力に関する設定を指定するセクション。
3. `file` セクション: 入出力ファイルのパスなどを設定するセクションで、 `input` , `output` のサブセクションで構成される。

以下、ファイル例を記載します。

```
[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"
```

ファイル形式

TOML 形式

パラメータ

mode セクション

- mode

  形式 : string 型

  説明 : 計算モードを指定します。実空間版 UHF を利用する場合には UHFr と入力して下さい。

- flag_fock

  形式 : bool 型 (デフォルトは true)

  説明 : true の場合には Fock 項を考慮し、 false の場合には Hartree 項のみ取り扱います。

mode.param セクション

mode.param セクションでは計算用のパラメータを指定します。

- T

  形式 : float 型 (デフォルトは 0)

  説明 : 温度を指定します。0 以上の値を指定してください。

- 2Sz

  形式 : int 型, string 型, または None (デフォルトは None)

  説明 : スピンの z 成分 Sz を固定したい場合に使用し、固定する Sz の 2 倍の値を指定します。その場合はスピン空間を up と down に分けた上で計算を実施します。何も指定しない場合 (None) または "free" を指定した場合は、スピン空間を分けずに計算を実施します。Sz が保存しないような場合 (スピン軌道相互作用がある場合など) には指定しないようにしてください。-Nsite から Nsite の値を指定してください。

- Nsite

  形式 : int 型

  説明 : サイトの数を指定します。1 以上の値を指定してください。

- Ncond

  形式 : int 型

  説明 : 伝導電子の数を指定します。1 以上の値を指定してください。

- filling

  形式 : float 型

  説明 : 伝導電子の状態数に対する占有率を指定します。0 以上 1 以下の値を指定してください。 Ncond と同時に指定されている場合はエラーで終了します。

- Ncond_round_mode

  形式 : str 型 (デフォルトは "strict")

  説明 : filling から計算される伝導電子数を整数値に丸める方法を指定します。以下のいずれかの値をとります。

  - as-is : 丸めを行いません。(戻り値は float 型です)
  - round-up : 小数点以下を切り上げます。
  - round-down : 小数点以下を切り捨てます。
  - round-off : 小数点以下を四捨五入します。
  - round : 小数点以下を round 関数で丸めます。0.5 は 0 に丸められるので注意。
  - strict : 整数でない場合はエラーで終了します。
  - exact : 整数でない場合は warning を表示し、 round で丸めた整数値を返します。

- `IterationMax`

  形式 : int 型 (デフォルトは 20000)

  説明 : 反復回数の上限を指定します。0 以上の値を指定してください。

- `EPS`

  形式 : int 型 (デフォルトは 6)

  説明 : 収束条件を指定します。一つ前のステップとのグリーン関数の差のノルムが $10^{-\mathrm{EPS}}$ 以下になった場合に収束したと判定します。残差は $R = \sum_{i,j}^{N} \sqrt{\left|G_{ij}^{\mathrm{new}} - G_{ij}^{\mathrm{old}}\right|^2 / 2N^2}$ で定義されます。0 以上の値を指定してください。

- `Mix`

  形式 : float 型 (デフォルトは 0.5)

  説明 : Green 関数の更新時に、古い値と新しく得られた値を混ぜる (simple-mixing) 割合 α を指定します。0 以上から 1 以下の実数で指定してください。1 にすると古い値は使われません。simple-mixing については **アルゴリズムの章** をご覧ください。

- `RndSeed`

  形式 : int 型 (デフォルトは 1234)

  説明 : 乱数のシード (種) を指定します。

- `ene_cutoff`

  形式 : float 型 (デフォルトは 100.0)

  説明 : Fermi 分布関数を計算する際に overflow を避けるためのカットオフを指定します。

- `strict_hermite`

  形式 : bool 型 (デフォルトは false)

  説明 : 相互作用定義ファイルの読み込み時に Hermiticity を厳密にチェックします。true の場合、`hermite_tolerance` 以上のズレが見つかったときはエラーで終了します。false の場合は warning を表示して実行を継続します。

- `hermite_tolerance`

  形式 : float 型 (デフォルトは 10^{-8})

  説明 : Hermiticity の許容値 $|t_{ij} - t_{ji}^*| < \varepsilon$ を指定します。

log セクション

- print_level

  形式 : int 型 (デフォルトは 1)

  説明 : 標準出力のレベルを指定します。1 にすると詳細な情報が出力されます。

- print_step

  形式 : int 型 (デフォルトは 1)

  説明 : 反復計算の途中に計算ログを標準出力に書き出す間隔を指定します。1 以上の値を指定してください。

- print_check

  形式 : str 型

  説明 : 反復計算の途中に計算ログをファイルに書き出す場合、出力先ファイル名を指定します。無指定のときは出力しません。

file セクション

input と output のサブセクションからなります。前者は入力ファイルに関する情報 (格納場所やファイル名の指定など)、後者は出力ファイルに関する情報 (格納場所など) について指定します。以下、順に説明します。

file.input セクション

- path_to_input

  形式 : str 型 (デフォルトは "")

  説明 : 入力ファイルの格納されているディレクトリを指定します。

- Initial

  形式 : str 型 (デフォルトは "")

  説明 : 初期配置を指定する入力ファイル名を指定します。

- OneBodyG

  形式 : str 型 (デフォルトは "")

  説明 : 出力したい一体グリーン関数を指定する入力ファイル名を指定します。

file.input.interaction セクション

- Trans

  形式 : str 型 (デフォルトは "")

  説明 : 一般的一体相互作用を記述するファイルを指定します。

- InterAll

  形式 : str 型 (デフォルトは "")

  説明 : 一般的二体相互作用を記述するファイルを指定します。

- CoulombIntra

  形式 : str 型 (デフォルトは "")

  説明 : サイト内クーロン相互作用を記述するファイルを指定します。

- CoulombInter

  形式 : str 型 (デフォルトは "")

  説明 : サイト間クーロン相互作用を記述するファイルを指定します。

- Hund

  形式 : str 型 (デフォルトは "")

  説明 : フント結合を記述するファイルを指定します。

- PairHop

  形式 : str 型 (デフォルトは "")

  説明 : ペアホッピングを記述するファイルを指定します。

- Exchange

  形式 : str 型 (デフォルトは "")

  説明 : 交換相互作用を記述するファイルを指定します。

- Ising

  形式 : str 型 (デフォルトは "")

  説明 : イジング相互作用を記述するファイルを指定します。

- PairLift

  形式 : str 型 (デフォルトは "")

  説明 : ペアリフト相互作用を記述するファイルを指定します。

file.output セクション

- path_to_output

  形式 : str 型 (デフォルトは "output")

  説明 : 出力ファイルを格納するディレクトリを指定します。

- energy

  形式 : str 型

  説明 : エネルギーを出力するファイル名を指定します。このキーワードがない場合には情報は出力されません。

- eigen

  形式 : str 型

  説明 : ハミルトニアンの固有値を出力するファイル名を指定します。このキーワードがない場合には情報は出力されません。

- green

  形式 : str 型

  説明 : 一体グリーン関数の出力ファイル名を指定します。このキーワードがない場合には情報は出力されません。

- initial

  形式 : str 型

  説明 : 初期状態読み込み用の一体グリーン関数の出力ファイル名を指定します。このキーワードがない場合には情報は出力されません。

- fij

  形式 : str 型

  説明 : ペア軌道因子 fij の出力ファイル名を指定します。このキーワードがない場合には情報は出力されません。

3.2.2 UHFr 用入力ファイル

H-wave で使用する入力ファイル (*.def) に関して説明します。入力ファイルの種別は以下の 2 つで分類されます。

(1) Hamiltonian:

Hamiltonian を電子系の表式により指定します。具体的には以下のファイルで指定されます。

Trans: $c^\dagger_{i\sigma_1}c_{j\sigma_2}$ で表される一体項を指定します。
InterAll: $c^\dagger_{i\sigma_1}c_{j\sigma_2}c^\dagger_{k\sigma_3}c_{l\sigma_4}$ で表される一般二体相互作用を指定します。

なお、使用頻度の高い相互作用に関しては下記のキーワードで指定することも可能です。

CoulombIntra: $n_{i\uparrow}n_{i\downarrow}$ で表される相互作用を指定します ($n_{i\sigma} = c^{\dagger}_{i\sigma}c_{i\sigma}$)。

CoulombInter: $n_i n_j$ で表される相互作用を指定します ($n_i = n_{i\uparrow} + n_{i\downarrow}$)。

Hund: $n_{i\uparrow}n_{j\uparrow} + n_{i\downarrow}n_{j\downarrow}$ で表される相互作用を指定します。

PairHop: $c^{\dagger}_{i\uparrow}c_{j\uparrow}c^{\dagger}_{i\downarrow}c_{j\downarrow}$ で表される相互作用を指定します。

Exchange: $S^{+}_i S^{-}_j$ で表される相互作用を指定します。

Ising: $S^{z}_i S^{z}_j$ で表される相互作用を指定します。

PairLift: $c^{\dagger}_{i\uparrow}c_{i\downarrow}c^{\dagger}_{j\uparrow}c_{j\downarrow}$ で表される相互作用を指定します。

(2) Green functions:

Initial : 初期値として入力する一体 Green 関数を指定します。$\langle c^{\dagger}_{i\sigma_1}c_{j\sigma_2}\rangle$ を入力します。

OneBodyG : 出力する一体 Green 関数を指定します。$\langle c^{\dagger}_{i\sigma_1}c_{j\sigma_2}\rangle$ が出力されます。

以下、各入力ファイルのファイルフォーマットについて詳細を記載します。

Trans 指定ファイル

ここではハミルトニアン

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

に対する Transfer 積分 $t_{ij\sigma_1\sigma_2}$ を指定します。以下にファイル例を記載します。

```
========================
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
```

(次のページに続く)

(前のページからの続き)

```
    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
…
```

ファイル形式

以下のように行数に応じ異なる形式をとります。

- 1 行: ヘッダ (何が書かれても問題ありません)。
- 2 行: `[ntransfer] [count]`
- 3-5 行: ヘッダ (何が書かれても問題ありません)。
- 6 行以降: `[i] [s1] [j] [s2] [v.real] [v.imag]`

パラメータ

- `[ntransfer]`

  形式 : string 型 (空白不可)

  説明 : Transfer 総数のキーワード名を指定します (任意)。

- `[count]`

  形式 : int 型 (空白不可)

  説明 : Transfer の総数を指定します。

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

  形式 : int 型 (空白不可)

  説明 : サイト番号を指定する整数。0 以上 `Nsite` 未満で指定します。

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

  形式 : int 型 (空白不可)

  説明 : スピンを指定する整数。0=アップスピン, 1=ダウンスピン のいずれかの値を取ります。

- `[v.real]`

  形式 : float 型 (空白不可)

  説明 : $t_{ij\sigma_1\sigma_2}$ の実部を指定します。

- `[v.imag]`

  形式 : float 型 (空白不可)

説明 : $t_{ij\sigma_1\sigma_2}$ の虚部を指定します。

使用ルール

本ファイルを使用するにあたってのルールは以下の通りです。

- 行数固定で読み込みを行う為、ヘッダの省略はできません。
- Hamiltonian がエルミートという制限から $t_{ij\sigma_1\sigma_2} = t^{\dagger}_{ji\sigma_2\sigma_1}$ の関係を満たす必要があります。上記の関係が成立しない場合には、`strict_hermite` パラメータの値に応じて、エラー終了またはメッセージを表示します。
- 成分が重複して指定された場合にはエラー終了します。
- `[count]` と定義されている Transfer の総数が異なる場合はエラー終了します。
- `[i]`, `[j]`, `[s1]`, `[s2]` を指定する際、範囲外の整数を指定した場合はエラー終了します。

InterAll 指定ファイル

ここでは一般二体相互作用をハミルトニアンに付け加えます。付け加える項は以下で与えられます。

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

以下にファイル例を記載します。

```
======================
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
…
```

ファイル形式

以下のように行数に応じ異なる形式をとります。

- 1 行: ヘッダ (何が書かれても問題ありません)。
- 2 行: `[ninterall] [count]`
- 3-5 行: ヘッダ (何が書かれても問題ありません)。
- 6 行以降: `[i] [s1] [j] [s2] [k] [s3] [l] [s4] [v.real] [v.imag]`

パラメータ

- [ninterall]

  形式 : string 型 (空白不可)

  説明 : 二体相互作用の総数のキーワード名を指定します (任意)。

- [count]

  形式 : int 型 (空白不可)

  説明 : 二体相互作用の総数を指定します。

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

  形式 : int 型 (空白不可)

  説明 : サイト番号を指定する整数。0 以上 Nsite 未満で指定します。

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

  形式 : int 型 (空白不可)

  説明 : スピンを指定する整数。0=アップスピン, 1=ダウンスピン のいずれかの値を取ります。

- [v.real]

  形式 : float 型 (空白不可)

  説明 : $I_{ijkl\sigma_1\sigma_2\sigma_3\sigma_4}$ の実部を指定します。

- [v.imag]

  形式 : float 型 (空白不可)

  説明 : $I_{ijkl\sigma_1\sigma_2\sigma_3\sigma_4}$ の虚部を指定します。

使用ルール

本ファイルを使用するにあたってのルールは以下の通りです。

- 行数固定で読み込みを行う為、ヘッダの省略はできません。
- Hamiltonian がエルミートという制限から $I_{ijkl\sigma_1\sigma_2\sigma_3\sigma_4} = I^{\dagger}_{lkji\sigma_4\sigma_3\sigma_2\sigma_1}$ の関係を満たす必要があります。上記の関係が成立しない場合には、strict_hermite パラメータの値に応じて、エラー終了またはメッセージを表示します。また、エルミート共役の形式は $I_{ijkl\sigma_1\sigma_2\sigma_3\sigma_4} c^{\dagger}_{i\sigma_1} c_{j\sigma_2} c^{\dagger}_{k\sigma_3} c_{l\sigma_4}$ に対して、$I_{lkji\sigma_4\sigma_3\sigma_2\sigma_1}\ c^{\dagger}_{l\sigma_4} c_{k\sigma_3} c^{\dagger}_{j\sigma_2} c_{i\sigma_1}$ を満たすように入力してください。
- 成分が重複して指定された場合にはエラー終了します。
- [count] と定義されている InterAll の総数が異なる場合はエラー終了します。
- [i], [j], [k], [l], [s1], [s2], [s3], [s4] を指定する際、範囲外の整数を指定した場合はエラー終了します。

CoulombIntra 指定ファイル

オンサイトクーロン相互作用をハミルトニアンに付け加えます。付け加える項は以下で与えられます。

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

以下にファイル例を記載します。

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

ファイル形式

以下のように行数に応じ異なる形式をとります。

- 1 行: ヘッダ (何が書かれても問題ありません)。
- 2 行: `[ncoulombintra] [count]`
- 3-5 行: ヘッダ (何が書かれても問題ありません)。
- 6 行以降: `[i] [val]`

パラメータ

- `[ncoulombintra]`

  形式 : string 型 (空白不可)

  説明 : オンサイトクーロン相互作用の総数のキーワード名を指定します (任意)。

- `[count]`

  形式 : int 型 (空白不可)

  説明 : オンサイトクーロン相互作用の総数を指定します。

- `[i]`

  形式 : int 型 (空白不可)

  説明 : サイト番号を指定する整数。0 以上 `Nsite` 未満で指定します。

- [val]

  形式 : float 型 (空白不可)

  説明 : U_i を指定します。

使用ルール

本ファイルを使用するにあたってのルールは以下の通りです。

- 行数固定で読み込みを行う為、ヘッダの省略はできません。
- 成分が重複して指定された場合にはエラー終了します。
- [count] と定義されているオンサイトクーロン相互作用の総数が異なる場合はエラー終了します。
- [i] を指定する際、範囲外の整数を指定した場合はエラー終了します。

CoulombInter 指定ファイル

オフサイトクーロン相互作用をハミルトニアンに付け加えます。付け加える項は以下で与えられます。

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

以下にファイル例を記載します。

```
======================
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
```

ファイル形式

以下のように行数に応じ異なる形式をとります。

- 1 行: ヘッダ (何が書かれても問題ありません)。
- 2 行: `[ncoulombinter] [count]`
- 3-5 行: ヘッダ (何が書かれても問題ありません)。
- 6 行以降: `[i] [j] [val]`

パラメータ

- `[ncoulombinter]`

  形式 : string 型 (空白不可)

  説明 : オフサイトクーロン相互作用の総数のキーワード名を指定します (任意)。

- `[count]`

  形式 : int 型 (空白不可)

  説明 : オフサイトクーロン相互作用の総数を指定します。

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

  形式 : int 型 (空白不可)

  説明 : サイト番号を指定する整数。0 以上 `Nsite` 未満で指定します。

- [val]

  形式 : float 型 (空白不可)

  説明 : V_{ij} を指定します。

使用ルール

本ファイルを使用するにあたってのルールは以下の通りです。

- 行数固定で読み込みを行う為、ヘッダの省略はできません。
- 成分が重複して指定された場合にはエラー終了します。
- [count] と定義されているオフサイトクーロン相互作用の総数が異なる場合はエラー終了します。
- [i], [j] を指定する際、範囲外の整数を指定した場合はエラー終了します。

Hund 指定ファイル

Hund カップリングをハミルトニアンに付け加えます。付け加える項は以下で与えられます。

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

以下にファイル例を記載します。

```
======================
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
```

ファイル形式

以下のように行数に応じ異なる形式をとります。

- 1 行: ヘッダ (何が書かれても問題ありません)。
- 2 行: `[nhund] [count]`
- 3-5 行: ヘッダ (何が書かれても問題ありません)。
- 6 行以降: `[i] [j] [val]`

パラメータ

- `[nhund]`

  形式 : string 型 (空白不可)

  説明 : Hund カップリングの総数のキーワード名を指定します (任意)。

- `[count]`

  形式 : int 型 (空白不可)

  説明 : Hund カップリングの総数を指定します。

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

  形式 : int 型 (空白不可)

  説明 : サイト番号を指定する整数。0 以上 `Nsite` 未満で指定します。

- [val]

  形式 : float 型 (空白不可)

  説明 : J_{ij}^{Hund} を指定します。

使用ルール

本ファイルを使用するにあたってのルールは以下の通りです。

- 行数固定で読み込みを行う為、ヘッダの省略はできません。
- 成分が重複して指定された場合にはエラー終了します。
- [count] と定義されている Hund カップリングの総数が異なる場合はエラー終了します。
- [i], [j] を指定する際、範囲外の整数を指定した場合はエラー終了します。

PairHop 指定ファイル

PairHop カップリングをハミルトニアンに付け加えます。付け加える項は以下で与えられます。

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

以下にファイル例を記載します。

```
======================
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
```

ファイル形式

以下のように行数に応じ異なる形式をとります。

- 1 行: ヘッダ (何が書かれても問題ありません)。
- 2 行: `[npairhop] [count]`
- 3-5 行: ヘッダ (何が書かれても問題ありません)。
- 6 行以降: `[i] [j] [val]`

パラメータ

- `[npairhop]`

  形式 : string 型 (空白不可)

  説明 : PairHop カップリングの総数のキーワード名を指定します (任意)。

- `[count]`

  形式 : int 型 (空白不可)

  説明 : PairHop カップリングの総数を指定します。

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

  形式 : int 型 (空白不可)

  説明 : サイト番号を指定する整数。0 以上 `Nsite` 未満で指定します。

- [val]

  形式 : float 型 (空白不可)

  説明 : J_{ij}^{Pair} を指定します。

使用ルール

本ファイルを使用するにあたってのルールは以下の通りです。

- 行数固定で読み込みを行う為、ヘッダの省略はできません。
- 成分が重複して指定された場合にはエラー終了します。
- [count] と定義されている PairHop カップリングの総数が異なる場合はエラー終了します。
- [i], [j] を指定する際、範囲外の整数を指定した場合はエラー終了します。

Exchange 指定ファイル

Exchange カップリングをハミルトニアンに付け加えます。

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

以下にファイル例を記載します。

```
======================
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
```

ファイル形式

以下のように行数に応じ異なる形式をとります。

- 1 行: ヘッダ (何が書かれても問題ありません)。
- 2 行: `[nexchange] [count]`
- 3-5 行: ヘッダ (何が書かれても問題ありません)。
- 6 行以降: `[i] [j] [val]`

パラメータ

- `[nexchange]`

  形式 : string 型 (空白不可)

  説明 : Exchange カップリングの総数のキーワード名を指定します (任意)。

- `[count]`

  形式 : int 型 (空白不可)

  説明 : Exchange カップリングの総数を指定します。

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

  形式 : int 型 (空白不可)

  説明 : サイト番号を指定する整数。0 以上 `Nsite` 未満で指定します。

- [val]

  形式 : float 型 (空白不可)

  説明 : J_{ij}^{Ex} を指定します。

使用ルール

本ファイルを使用するにあたってのルールは以下の通りです。

- 行数固定で読み込みを行う為、ヘッダの省略はできません。
- 成分が重複して指定された場合にはエラー終了します。
- [count] と定義されている Exchange カップリングの総数が異なる場合はエラー終了します。
- [i], [j] を指定する際、範囲外の整数を指定した場合はエラー終了します。

Ising 指定ファイル

Ising 相互作用をハミルトニアンに付け加えます ($S = 1/2$ の系でのみ使用可能)。 電子系の場合には

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

が付け加えられ、スピン系の場合には

$$\mathcal{H} += \sum_{i,j} J^z_{ij} S^z_i S^z_j \tag{3.11}$$

が付け加えられます。 以下にファイル例を記載します。

```
======================
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
```

ファイル形式

以下のように行数に応じ異なる形式をとります。

- 1 行: ヘッダ (何が書かれても問題ありません)。
- 2 行: `[nising] [count]`
- 3-5 行: ヘッダ (何が書かれても問題ありません)。
- 6 行以降: `[i] [j] [val]`

パラメータ

- `[nising]`

  形式 : string 型 (空白不可)

  説明 : Ising 相互作用の総数のキーワード名を指定します (任意)。

- `[count]`

  形式 : int 型 (空白不可)

  説明 : Ising 相互作用の総数を指定します。

- [i], [j]

  形式 : int 型 (空白不可)

  説明 : サイト番号を指定する整数。0 以上 Nsite 未満で指定します。

- [val]

  形式 : double 型 (空白不可)

  説明 : J_{ij}^{z} を指定します。

使用ルール

本ファイルを使用するにあたってのルールは以下の通りです。

- 行数固定で読み込みを行う為、ヘッダの省略はできません。
- 成分が重複して指定された場合にはエラー終了します。
- [count] と定義されている Ising 相互作用の総数が異なる場合はエラー終了します。
- [i], [j] を指定する際、範囲外の整数を指定した場合はエラー終了します。

PairLift 指定ファイル

PairLift カップリングをハミルトニアンに付け加えます。 付け加える項は以下で与えられます。

$$\mathcal{H} = \sum_{i,j} J_{ij}^{\mathrm{PairLift}} (c_{i\uparrow}^{\dagger} c_{i\downarrow} c_{j\uparrow}^{\dagger} c_{j\downarrow} + c_{i\downarrow}^{\dagger} c_{i\uparrow} c_{j\downarrow}^{\dagger} c_{j\uparrow}) \tag{3.12}$$

以下にファイル例を記載します。

```
======================
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
```

ファイル形式

以下のように行数に応じ異なる形式をとります。

- 1 行: ヘッダ (何が書かれても問題ありません)。
- 2 行: `[npairlift] [count]`
- 3-5 行: ヘッダ (何が書かれても問題ありません)。
- 6 行以降: `[i] [j] [val]`

パラメータ

- `[npairlift]`

  形式 : string 型 (空白不可)

  説明 : PairLift カップリングの総数のキーワード名を指定します (任意)。

- `[count]`

  形式 : int 型 (空白不可)

  説明 : PairLift カップリングの総数を指定します。

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

  形式 : int 型 (空白不可)

  説明 : サイト番号を指定する整数。0 以上 `Nsite` 未満で指定します。

- [val]

  形式 : float 型 (空白不可)

  説明 : $J_{ij}^{\mathrm{PairLift}}$ を指定します。

使用ルール

本ファイルを使用するにあたってのルールは以下の通りです。

- 行数固定で読み込みを行う為、ヘッダの省略はできません。
- 成分が重複して指定された場合にはエラー終了します。
- [count] と定義されている PairLift カップリングの総数が異なる場合はエラー終了します。
- [i], [j] を指定する際、範囲外の整数を指定した場合はエラー終了します。

Initial 指定ファイル

グリーン関数 $G_{ij\sigma_1\sigma_2} \equiv \langle c^\dagger_{i\sigma_1} c_{j\sigma_2} \rangle$ の初期値を与えます。ファイル形式は出力ファイルの green ファイルと同じです。なお、値を指定しないグリーン関数の要素には 0 が入ります。以下にファイル例を記載します。

```
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
…
```

ファイル形式

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

パラメータ

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

  形式 : int 型

  説明 : サイト番号を指定する整数。`[i]` が i サイト、`[j]` が j サイトを表します。

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

  形式 : int 型

  説明 : スピンを指定する整数。`[s1]` が σ_1、`[s2]` が σ_2 に対応します。0=アップスピン, 1=ダウンスピン を表します。

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

  形式 : float 型

  説明 : $\langle c^\dagger_{i\sigma_1} c_{j\sigma_2} \rangle$ の値を表します。`[v.real]` が実部、`[v.imag]` が虚部を表します。

OneBodyG 指定ファイル

計算する一体グリーン関数 $\langle c^{\dagger}_{i\sigma_1} c_{j\sigma_2} \rangle$ を指定します。以下にファイル例を記載します。

```
==============================
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
```

ファイル形式

以下のように行数に応じ異なる形式をとります。

- 1 行: ヘッダ (何が書かれても問題ありません)。
- 2 行: `[ncisajs] [count]`
- 3-5 行: ヘッダ (何が書かれても問題ありません)。
- 6 行以降: `[i] [s1] [j] [s2]`

パラメータ

- `[ncisajs]`

  形式 : string 型 (空白不可)

  説明 : 一体グリーン関数成分総数のキーワード名を指定します (任意)。

- `[count]`

  形式 : int 型 (空白不可)

  説明 : 一体グリーン関数成分の総数を指定します。

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

  形式 : int 型 (空白不可)

  説明 : サイト番号を指定する整数。0 以上 `Nsite` 未満で指定します。

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

  形式 : int 型 (空白不可)

  説明 : スピンを指定する整数。0=アップスピン, 1=ダウンスピンの値を取ります。

使用ルール

本ファイルを使用するにあたってのルールは以下の通りです。

- 行数固定で読み込みを行う為、ヘッダの省略はできません。
- 成分が重複して指定された場合にはエラー終了します。
- `[count]` と定義されている一体グリーン関数成分の総数が異なる場合はエラー終了します。
- `[i]`, `[j]`, `[s1]`, `[s2]` を指定する際、範囲外の整数を指定した場合はエラー終了します。

3.2.3 UHFr の出力ファイル

UHFr では固有エネルギー, 固有ベクトル, 一体グリーン関数の出力が行われます。以下、各出力ファイルに関するファイル形式を記載します。

energy

UHF 法で求めたエネルギー、粒子数、スピンに関する計算結果を出力します。ファイル名は環境設定ファイルの中の `file.output` セクションでキーワード `energy` を用いて指定することができます。以下にファイル例を記載します。

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

ファイル形式

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

パラメータ

- `[energy_total]`

  形式 : float 型

  説明 : UHF 法で求めた固有ベクトルを用い計算した全エネルギー。

- `[energy_band]`

  形式 : float 型

  説明 : UHF 法で求めたハミルトニアン行列の固有値のみ考慮した場合のエネルギー。

- `[energy_interall]`

  形式 : float 型

  説明 : 相互作用分のエネルギー。

- [ncond]

  形式 : float 型

  説明 : 全粒子数の期待値。 $\sum_i \langle n_i \rangle$

- [sz]

  形式 : float 型

  説明 : 全スピンの z 成分 S_z の期待値。 $\sum_i \langle (n_{i\uparrow} - n_{i\downarrow}) \rangle / 2$

eigen

収束したハミルトニアンの固有値、固有ベクトルを npz 形式で出力します。ファイルは環境設定ファイルの file.output セクション eigen で指定された文字列 (以下、eigen_str) を用いて、{key}_eigen_str.npz という名前で出力されます。ここで、 {key} は

- mode.param セクションで Sz を指定しない場合: sz-free
- mode.param セクションで Sz を指定した場合: spin-up , spin-down

となります (Sz を指定した場合には 2 つのファイルが書き出されます)。以下、データを読み込む例となります。

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

eigenvalue には固有値が低い順に格納されます。 N を全サイト数とした場合、

- mode.param セクションで Sz を指定しない場合: 2 N 個
- mode.param セクションで Sz を指定した場合: N 個

の固有値が出力されます。

eigenvector には対応する固有ベクトルが格納されます。第 1 列目の index は n_site をサイトの index、n_spin をスピンの index(up の場合に 0, down の場合に 1) として、

- mode.param セクションで Sz を指定しない場合: n_site + n_spin * N
- mode.param セクションで Sz を指定した場合: n_site

に対応し、第 2 列目の index は固有値の index に対応します。

green

一体グリーン関数 $\langle c^{\dagger}_{i\sigma_1} c_{j\sigma_2} \rangle$ の計算結果を出力します。出力する要素のインデックスは OneBodyG で指定します。ファイル名は環境設定ファイルの中の file.output セクションでキーワード green を用いて指定することができます。以下に出力例を記載します。

```
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
…
```

ファイル形式

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

パラメータ

- [i], [j]

  形式 : int 型

  説明 : サイト番号を指定する整数。[i] が i サイト、[j] が j サイトを表します。

- [s1], [s2]

  形式 : int 型

  説明 : スピンを指定する整数。[s1] が σ_1、[s2] が σ_2 に対応します。0=アップスピン, 1=ダウンスピン を表します。

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

  形式 : float 型

  説明 : $\langle c^{\dagger}_{i\sigma_1} c_{j\sigma_2} \rangle$ の値を表します。[v.real] が実部、[v.imag] が虚部を表します。

第4章　波数空間UHF(UHFk)

4.1 チュートリアル

H-wave を波数空間モード (UHFk) で実行するには、入力ファイルとして

1. 環境設定入力ファイル
2. 相互作用定義ファイル

を用意した後、プログラムを実行します。2. は RESPACK 等の外部プログラムの出力を利用する他、StdFace ライブラリを使って生成することもできます。

以下では、 `docs/tutorial/Hubbard/UHFk` ディレクトリにあるサンプルを例にチュートリアルを実施します。相互作用定義ファイルは StdFace ライブラリを用いて生成することもできます。詳細は *StdFace* **を用いた相互作用ファイルの作成** の章をご覧ください。

4.1.1 環境設定入力ファイルの作成

環境設定入力ファイルには、基本パラメータの指定と入出力を制御する情報を記述します。`docs/tutorial/Hubbard/UHFk` ディレクトリ内に `input.toml` というファイルがありますが、これが入力パラメータファイルになります。以下、ファイルの内容を記載します。

```
[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]
```

(次のページに続く)

(前のページからの続き)

```
[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"
```

このファイルは TOML 形式で記述され、内容ごとにセクションに分類されています。

[log] セクション

ログ出力に関する設定を行います。print_level で標準出力のレベル、 print_step でログ出力を行う繰り返し間隔を指定します。

[mode] セクション

実行モードに関する設定および基本パラメータの指定を行います。mode で実空間版 (UHF) または波数空間版 (UHFk) を選択します。[mode.param] サブセクションには計算実行時のパラメータを指定します。

[file] セクション

[file.input] サブセクションでは、入力ファイルを格納するディレクトリ path_to_input および初期配位データファイルのファイル名 initial を指定します。指定がない場合は乱数を用いて初期化されます。[file.input.interaction] サブセクションには、幾何情報および相互作用定義を格納するファイルのファイル名を相互作用のタイプごとに列挙します。

[file.output] サブセクションには、エネルギーなどの物理量を出力するファイル名 energy 、ハミルトニアンの固有値・固有ベクトルを出力するファイル名 eigen 、一体グリーン関数を書き出す出力ファイル名 green を指定します。これらのキーワードがない場合にはその項目は出力されません。

詳細については **ファイルフォーマット** の章をご覧ください。

4.1.2 相互作用定義ファイルの作成

Hamiltonian を構築するための格子の幾何情報および相互作用係数を格納したデータファイルを作成します。項目とファイル名の対応付けは、入力パラメータファイルの [file.input.interaction] セクションで行います。

Geometry

格子の幾何情報を記述します。ファイル例を以下に示します。

```
  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
```

基本ベクトル (1〜3 行目)、軌道の数 (4 行目)、各軌道の Wannier center(5 行目以降) を記載します。

Transfer, CoulombIntra, CoulombInter, Hund, etc

Transfer に指定するファイルは、電子系の Transfer に相当する Hamiltonian の係数を格納します。また、二体相互作用の係数は相互作用のタイプごとに係数を格納するファイルを指定します。

相互作用のタイプは、実空間版 UHF の入力ファイル形式と対応して、CoulombItra, CoulombInter, Hund, Ising, Exchange, PairLift, PairHop が定義されています。

これらのファイルは Wannier90(-like) 形式で記述されます。以下に例を示します。

```
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
```

コメント行 (1 行目)、軌道の数 (2 行目)、並進ベクトルをすべて収める直方体内のセルの総数 nrpts (3 行目)、縮重度 (nrpts 個を 1 行あたり 15 個ずつ)、係数行列の要素を記載します。

行列要素の各行は、並進ベクトル r_x, r_y, r_z 、軌道のインデックス α, β 、係数の値の実部・虚部です。

4.1.3 計算の実行

全ての入力ファイルが準備できた後、プログラムを実行して計算を行います。入力パラメータファイル (ここでは input.toml) を引数とし、ターミナルから H-wave を実行します。

```
$ hwave input.toml
```

計算が開始されると以下のようなログが出力されます。

```
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
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.
```

(次のページに続く)

(前のページからの続き)

```
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
  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
--------------------------------------------------------------------------------
```

入力ファイル読み込みに関するログが出力されたあと、波数空間 UHF 計算の計算過程に関する情報が出力されます。出力ファイルは `input.toml` の `[file.output]` セクションの指定に従い、`output` ディレクトリに `energy.dat` , `eigen.npz`, `green.npz` ファイルが出力されます。

出力ファイルの詳細については **ファイルフォーマット** の章をご覧ください。

4.1.4 状態密度の計算 (hwave_dos)

ポストツール hwave_dos を用いることで、状態密度を計算することができます。ブリルアンゾーン積分を精度良く計算するために libtetrabz を利用しています。 pip を利用してインストールしてください。

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

hwave_dos は、 hwave で利用した入力パラメータファイルを引数として受け取ります

```
$ hwave_dos input.toml
```

hwave_dos は、hwave と同様に [file.output] セクションの指定に従い、output ディレクトリに dos.dat ファイルを出力します。ファイル名は --output オプションで変更することができます。

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

状態密度を計算するエネルギーの範囲は --ene-window オプションで指定します。省略した場合は、hwave で得られたエネルギーの最小値と最大値を $E_{\min}, E_{\max}$ として、$[E_{\min} - 0.2, E_{\max} + 0.2]$ で計算されます。エネルギーの点数は --ene-num オプションで指定します（デフォルトは 101）

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

--plot オプションを指定すると、状態密度をプロットします。 matplotlib が必要です。

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

4.2 ファイルフォーマット

4.2.1 環境設定ファイル

このファイルでは、TOML 形式で H-wave に関する環境を設定します。本ファイルは以下の 3 つのセクションから構成されます。

1. mode セクション: 計算モードに関する設定を指定するセクション。
2. log セクション: 標準出力に関する設定を指定するセクション。
3. file セクション: 入出力ファイルのパスなどを設定するセクションで、 input , output のサブセクションから構成される。

以下、ファイル例を記載します。

```
[log]
  print_level = 1
  print_step = 10
[mode]
  mode = "UHFk"
```

(次のページに続く)

(前のページからの続き)

```
[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"
```

ファイル形式

TOML 形式

パラメータ

mode セクション

- mode

  形式 : string 型

  説明 : 計算モードを指定します。実波数空間 UHF を利用する場合には、 UHFk (波数空間版) を記載します。

- flag_fock

  形式 : bool 型 (デフォルトは true)

  説明 : true の場合には Fock 項を考慮し、 false の場合には Hartree 項のみ取り扱います。

- enable_spin_orbital

  形式 : bool 型 (デフォルトは false)

  説明 : スピン軌道相互作用を有効にします。Transfer 項の軌道のインデックスはスピン自由度を含む形に解釈されます。インデックスの対応は、軌道 α とスピン s に対して $\alpha + N_{\text{orb}} \cdot s$ となります。

mode.param セクション

mode.param セクションでは計算用のパラメータを指定します。

- CellShape

  形式 : int array 型

  説明 : box の形状 Lx, Ly, Lz を指定します。

- SubShape

  形式 : int array 型 (デフォルトは [Lx, Ly, Lz])

  説明 : 副格子の形状 Bx, By, Bz を指定します。

- T

  形式 : float 型 (デフォルトは 0)

  説明 : 温度を指定します。0 以上の値を指定してください。

- Ncond

  形式 : int 型

  説明 : 伝導電子の数を指定します。1 以上の値を指定してください。

- filling

  形式 : float 型

  説明 : 伝導電子の状態数に対する占有率を指定します。0 以上 1 以下の値を指定してください。 Ncond と同時に指定されている場合はエラーで終了します。

- Ncond_round_mode

  形式 : str 型 (デフォルトは "strict")

  説明 : filling から計算される伝導電子数を整数値に丸める方法を指定します。以下のいずれかの値をとります。

  - as-is : 丸めを行いません。(戻り値は float 型です)
  - round-up : 小数点以下を切り上げます。
  - round-down : 小数点以下を切り捨てます。
  - round-off : 小数点以下を四捨五入します。
  - round : 小数点以下を round 関数で丸めます。0.5 は 0 に丸められるので注意。

  - `strict` : 整数でない場合はエラーで終了します。
  - `exact` : 整数でない場合は warning を表示し、 `round` で丸めた整数値を返します。

- `IterationMax`

  形式 : int 型 (デフォルトは 20000)

  説明 : 反復回数の上限を指定します。0 以上の値を指定してください。

- `EPS`

  形式 : int 型 (デフォルトは 6)

  説明 : 収束条件を指定します。一つ前のステップのグリーン関数との差のノルムが $10^{-\mathrm{EPS}}$ 以下になった場合に収束したと判定します。残差は $R = \sum_{i,j}^{N} \sqrt{\left|G_{ij}^{\mathrm{new}} - G_{ij}^{\mathrm{old}}\right|^2}/2N^2$ で定義されます。0 以上の値を指定してください。

- `Mix`

  形式 : float 型 (デフォルトは 0.5)

  説明 : Green 関数の更新時に、古い値と新しく得られた値を混ぜる (simple-mixing) 割合 α を指定します。0 以上から 1 以下の実数で指定してください。1 にすると古い値は使われません。simple-mixing については **アルゴリズムの章** をご覧ください。

- `RndSeed`

  形式 : int 型 (デフォルトは 1234)

  説明 : 乱数のシード (種) を指定します。

- `ene_cutoff`

  形式 : float 型 (デフォルトは 100.0)

  説明 : Fermi 分布関数を計算する際に overflow を避けるためのカットオフを指定します。

- `strict_hermite`

  形式 : bool 型 (デフォルトは false)

  説明 : 相互作用定義ファイルの読み込み時に Hermiticity を厳密にチェックします。true の場合、`hermite_tolerance` 以上のズレが見つかったときはエラーで終了します。false の場合は warning を表示して実行を継続します。

- `hermite_tolerance`

  形式 : float 型 (デフォルトは 10^{-8})

  説明 : Hermiticity の許容値 $|t_{ij} - t_{ji}^*| < \varepsilon$ を指定します。

- `trustme_interaction_range`

  形式 : bool 型 (デフォルトは false)

  説明 : ホッピングや相互作用の距離が `CellShape` の半分以内におさまっているかのチェックを緩和します。true の場合、warning を表示して実行を継続し、 false の場合、エラー終了します。

log セクション

- print_level

  形式 : int 型 (デフォルトは 1)

  説明 : 標準出力のレベルを指定します。1 にすると詳細な情報が出力されます。

- print_step

  形式 : int 型 (デフォルトは 1)

  説明 : 反復計算の途中に計算ログを標準出力に書き出す間隔を指定します。1 以上の値を指定してください。

- print_check

  形式 : str 型 (デフォルトは None)

  説明 : 反復計算の途中に計算ログをファイルに書き出す場合、出力先ファイル名を指定します。無指定のときは出力しません。

file セクション

input と output のサブセクションからなります。前者は入力ファイルに関する情報 (格納場所やファイル名の指定など)、後者は出力ファイルに関する情報 (格納場所など) について指定します。以下、順に説明します。

file.input セクション

- path_to_input

  形式 : str 型 (デフォルトは "")

  説明 : 入力ファイルの格納されているディレクトリを指定します。

- initial

  形式 : str 型

  説明 : 初期状態の一体グリーン関数データを格納したファイルの名前を指定します。ファイルは NumPy binary 形式です。file.output の green の出力ファイル形式に対応しています。

- initial_mode

  形式 : str 型

  説明 : 初期状態の一体グリーン関数データが指定されていない場合、初期値をどのように取るかを指定します。以下のいずれかの値を取ります。

  zero: 初期値を 0 に設定します。(デフォルト)

  one または unity: 初期値を $G_{\alpha\sigma,\beta\sigma'}(\vec{r}) = \delta_{\vec{r},0}\delta_{\alpha\beta}\delta_{\sigma\sigma'}$ で与えます。

  random: 初期値を乱数で与えます。

file.input.interaction セクション

波数空間版 UHF で、幾何情報や相互作用のタイプと定義ファイルとの対応付けを行います。

- path_to_input

  形式 : str 型

  説明 : 入力ファイルを格納するディレクトリを指定します。 file.input セクションの path_to_input とは独立に指定できます。

- Geometry

  形式 : str 型

  説明 : 幾何情報のファイル名を指定します。

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

  形式 : str 型

  説明 : 各相互作用の係数ファイル名を指定します。

file.output セクション

- path_to_output

  形式 : str 型 (デフォルトは "output")

  説明 : 出力ファイルを格納するディレクトリを指定します。

- energy

  形式 : str 型

  説明 : エネルギーを出力するファイル名を指定します。このキーワードがない場合には情報は出力されません。

- eigen

  形式 : str 型

  説明 : ハミルトニアンの固有値を出力するファイル名を指定します。このキーワードがない場合には情報は出力されません。

- green

  形式 : str 型 (デフォルトは "green")

  説明 : 一体グリーン関数を出力するファイル名を指定します。空文字列 "" を指定した場合には情報は出力されません。

- rpa

  形式 : str 型

説明 : UHF 法で近似した相互作用項を含む一体項を出力するファイル名を指定します。出力ファイルは RPA の初期値として `trans_mod` パラメータに指定して利用できます。このキーワードがない場合には情報は出力されません。

4.2.2 UHFk 用入力ファイル

波数空間 UHF で使用する入力ファイルに関して説明します。入力ファイルの種別は以下の 2 つに分類され、データフォーマットは Wannier90 形式です。

(1) 幾何情報:

Geometry: 格子の幾何情報を設定します。

(2) 相互作用定義:

UHF のハミルトニアンを電子系の表式により指定します。具体的には、キーワードで指定する各相互作用のタイプについて、ハミルトニアンの係数をデータとして与えます。

$\mathcal{H}\Phi$ や mVMC の Expert Mode 入力に相当する以下のキーワードを指定できます。

Transfer: $c^\dagger_{i\sigma_1}c_{j\sigma_2}$ で表される一体項を指定します。

CoulombIntra: $n_{i\uparrow}n_{i\downarrow}$ で表される相互作用を指定します ($n_{i\sigma} = c^\dagger_{i\sigma}c_{i\sigma}$)。

CoulombInter: n_in_j で表される相互作用を指定します ($n_i = n_{i\uparrow} + n_{i\downarrow}$)。

Hund: $n_{i\uparrow}n_{j\uparrow} + n_{i\downarrow}n_{j\downarrow}$ で表される相互作用を指定します。

Ising: $S^z_iS^z_j$ で表される相互作用を指定します。

Exchange: $S^+_iS^-_j$ で表される相互作用を指定します。

PairLift: $c^\dagger_{i\uparrow}c_{i\downarrow}c^\dagger_{j\uparrow}c_{j\downarrow}$ で表される相互作用を指定します。

PairHop: $c^\dagger_{i\uparrow}c_{j\uparrow}c^\dagger_{i\downarrow}c_{j\downarrow}$ で表される相互作用を指定します。

以下のセクションでデータフォーマットについて記述します。

幾何情報指定ファイル

格子の幾何情報を記述します。ファイル例を以下に示します。

```
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
```

(次のページに続く)

(前のページからの続き)

```
 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
```

ファイル形式

- 1-3 行: `[ax_i] [ay_i] [az_i]`
- 4 行: `[Norbit]`
- 5 行以降: `[vx_i] [vy_i] [vz_i]`

パラメータ

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

  形式 : float 型

  説明 : 基本格子ベクトル $\vec{a}_1, \vec{a}_2, \vec{a}_3$ を指定します。

- `[Norbit]`

  形式 : int 型

  説明 : ユニットセル内の軌道の数 N_{orbit} を指定します。

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

  形式 : float 型

  説明 : 各軌道の Wannier center $\vec{v}_i$ を fractional coordinate 表記で指定します。

使用ルール

本ファイルを使用するにあたってのルールは以下の通りです。

- 軌道のインデックスは暗黙的に Wannier center の並び順に 1〜N_{orbit} で与えられます。

相互作用指定ファイル

次の形で表わされるハミルトニアンの一体項および二体相互作用項について、その係数 $T_{\alpha\beta}(r_{ij})$ および $J_{\alpha\beta}(r_{ij}), V_{\alpha\beta}(r_{ij}), U_{\alpha}$ を共通の Wannier90(-like) 形式で記述します。なお、波数空間版 UHF では一般化二体相互作用 InterAll 形式には対応していません。

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

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

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

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

Ising: $\sum_{ij\alpha\beta} J^{\mathrm{Ising}}_{\alpha\beta}(r_{ij}) S^{z}_{i\alpha} S^{z}_{j\beta}$ $(S^{z}_{i\alpha} = \frac{1}{2}(n_{i\alpha\uparrow} - n_{i\alpha\downarrow}))$

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

Exchange: $\sum_{ij\alpha\beta} J^{\mathrm{Ex}}_{\alpha\beta}(r_{ij})\, c^{\dagger}_{i\alpha\uparrow} c_{j\beta\uparrow} c^{\dagger}_{j\beta\downarrow} c_{i\alpha\downarrow}$

PairLift: $\sum_{ij\alpha\beta} J^{\mathrm{PairLift}}_{\alpha\beta}(r_{ij})\, c^{\dagger}_{i\alpha\uparrow} c_{i\alpha\downarrow} c^{\dagger}_{j\beta\uparrow} c_{j\beta\downarrow}$

以下にファイル例を示します。

```
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
...(略)
 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
...(略)
```

ファイル形式

- 1 行: ヘッダ (何が書かれても問題ありません)。
- 2 行: `[Norbit]`
- 3 行: `[Npts]`
- 4 - $\lceil N_{\text{pts}}/15 \rceil + 3$ 行: `[n_1] [n_2] ...`
- $\lceil N_{\text{pts}}/15 \rceil + 4$ 行以降: `[r_x] [r_x] [r_x] [alpha] [beta] [J.real] [J.imag]`

パラメータ

- `[Norbit]`

  形式 : int 型

  説明 : ユニットセル内の軌道の数 N_{orbit} を指定します。

- `[Npts]`

  形式 : int 型

  説明 : 並進ベクトル全体が入る直方体に含まれるセルの数を指定します。

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

  形式 : int 型

  説明 : 各セルの縮重度を指定します (通常は 1)。一行あたり 15 点を列挙します。

- `[r_x]`, `[r_y]`, `[r_z]`

  形式 : int 型

  説明 : 並進ベクトルを指定します。

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

  形式 : int 型

  説明 : 軌道のインデックスを指定します。`[alpha]` が元のセル内の軌道、`[beta]` が $\vec{r}$ 離れたセル内の軌道を指します。

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

  形式 : float 型

  説明 : 係数 $J_{\alpha\beta}(\vec{r})$ の実部と虚部を指定します。

使用ルール

本ファイルを使用するにあたってのルールは以下の通りです。

- 行数固定で読み込みを行うため、ヘッダの省略はできません。
- 係数行列のうち、省略された要素は 0 と仮定します。
- 並進ベクトルは全て `CellShape` 内に収まるとします。`r_x`, `r_y`, `r_z` の範囲が `CellShape` の x,y,z 軸のサイズを超える場合はエラーで終了します。
- `mode.enable_spin_orbital` が `true` の場合、Transfer 項の軌道のインデックスはスピン自由度を含む一般化軌道インデックスと読み替え、1〜 $2N_{\mathrm{orbital}}$ の値をとります。1〜 N_{orbital} が spin up、$N_{\mathrm{orbital}}+1$ 〜 $2N_{\mathrm{orbital}}$ が spin down に対応します。 `mode.enable_spin_orbital` が `false` の場合は、インデックスの範囲が 1〜 N_{orbital} の行のみ考慮します。

4.2.3 UHFk の出力ファイル

UHFk の出力ファイルに関するファイル形式を記載します。

energy

波数空間版 UHF 法で求めたエネルギー、粒子数、スピンに関する計算結果を出力します。ファイル名は環境設定ファイルの中の `file.output` セクションでキーワード `energy` を用いて指定することができます。以下にファイル例を記載します。

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

ファイル形式

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

パラメータ

- [energy_total]

  形式 : float 型

  説明 : UHF 法で求めた固有ベクトルを用い計算した全エネルギー。

- [energy_band]

  形式 : float 型

  説明 : UHF 法で求めたハミルトニアン行列の固有値のみ考慮した場合のエネルギー。

- [energy_type]

  形式 : float 型

  説明 : 相互作用分のエネルギー。相互作用のタイプごとに出力される。

- [ncond]

  形式 : float 型

  説明 : 全粒子数の期待値。 $\sum_i \langle n_i \rangle$

- [sz]

  形式 : float 型

  説明 : 全スピンの z 成分 S_z の期待値。 $\sum_i \langle (n_{i\uparrow} - n_{i\downarrow}) \rangle / 2$

eigen

収束したハミルトニアンの固有値、固有ベクトルを npz 形式で出力します。ファイル名は、環境設定ファイルの file.output セクション eigen で指定された文字列 (以下、 eigen_str) を用いて、 eigen_str.npz という名前で出力されます。

以下、データを読み込む例となります。

```
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 には波数ごとの固有値 $\lambda_l(\vec{k})$ が格納されます。副格子を指定している場合は、副格子を単位とした値になります。データ形式は numpy ndarray で、データの並びは eigenvalue[k][l] です。k は波数ベクトル $\vec{k}$ を一次元化したインデックス、l はセル内の固有値のインデックスです。Sz 固定の場合は、固有値のインデックスは軌道部分 l' とスピン s (up-spin は 0, down-spin は 1) に対して l' + Norb * s となります。 Norb はセル内の軌道数です。

eigenvector には対応する固有ベクトルが格納されます。データ形式は numpy ndarray で、データの並びは eigenvector[k][j][l] です。k, l は対応する波数および固有値のインデックス、j はセル内の軌道・スピンのインデックスです。

ファイルには波数の情報も出力されます。wavevector_unit には逆格子ベクトル $\vec{b}_i$ を用いて $2\pi\vec{b}_i/N_i$ で表される単位波数ベクトルが格納されます。wavevector_index には波数のインデックスと 1 次元化したインデックスとの対応が格納されます。インデックス k に対応する波数ベクトルは以下で求められます。

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

green

一体グリーン関数 $\langle c^\dagger_{i\sigma_1} c_{j\sigma_2} \rangle$ の計算結果を npz 形式で出力します。ファイル名は、環境設定ファイルの中の file.output セクションでキーワード green を用いて指定された文字列 (以下、green_str) を用いて、green_str.npz という名前で出力されます。

データはキー green にバインドされます。データ配列は ndarray(r, s, a, t, b) で、インデックスは以下のとおりです。

- r: 並進ベクトル $[r_x\ r_y\ r_z]$ を 1 次元化したインデックスで、$\mathrm{r} = r_z + N_z \cdot (r_y + N_y r_x)$
- a, b: 軌道のインデックス α, β
- s, t: スピンのインデックス σ_1, σ_2

出力ファイルは、file.input セクションの initial で指定するグリーン関数の初期データとして使用できます。

副格子を指定している場合は、上記に加えて、副格子を単位としたグリーン関数の値がキー green_sublattice にバインドされます。並進ベクトルおよび軌道のインデックスは副格子に読み替えます。

以下、データを読み込む例となります。

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

第5章　乱雑位相近似 (RPA)

5.1 チュートリアル

H-wave を乱雑位相近似モード (RPA) で実行するには、入力ファイルとして

1. 環境設定入力ファイル
2. 相互作用定義ファイル

を用意した後、プログラムを実行します。以下では、 docs/tutorial/Hubbard/RPA ディレクトリにあるサンプルを例にチュートリアルを実施します。相互作用定義ファイルは StdFace ライブラリを用いて生成することもできます。詳細は *StdFace* **を用いた相互作用ファイルの作成** の章をご覧ください。

5.1.1 環境設定入力ファイルの作成

環境設定入力ファイルには、基本パラメータの指定と入出力を制御する情報を記述します。docs/tutorial/Hubbard/RPA ディレクトリ内に input.toml というファイルがありますが、これが入力パラメータファイルになります。以下、ファイルの内容を記載します。

```
[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"
```

(次のページに続く)

(前のページからの続き)

```
  # 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"
```

このファイルは TOML 形式で記述され、内容ごとにセクションに分類されています。

[log] セクション

ログ出力に関する設定を行います。print_level で標準出力のレベルを指定します。

[mode] セクション

実行モードに関する設定および基本パラメータの指定を行います。mode で乱雑位相近似 (RPA) を選択します。[mode.param] サブセクションには計算実行時のパラメータを指定します。

[file] セクション

[file.input] サブセクションでは、入力ファイルを格納するディレクトリ path_to_input を指定します。以前に計算した既約感受率 $\chi_0(\vec{q})$ のデータを chi0q_init に指定したファイルから読み込んで感受率の計算を行うことができます。[file.input.interaction] サブセクションには、幾何情報および相互作用定義を格納するファイルのファイル名を相互作用のタイプごとに列挙します。

[file.output] サブセクションには、既約感受率 $\chi_0(\vec{q})$ を出力するファイル名 chi0q 、感受率 $\chi(\vec{q})$ を出力するファイル名 chiq を指定します。これらのキーワードがない場合にはその項目は出力されません。

詳細については **ファイルフォーマット** の章をご覧ください。

5.1.2 相互作用定義ファイルの作成

Hamiltonian を構築するための格子の幾何情報および相互作用係数を格納したデータファイルを作成します。項目とファイル名の対応付けは、入力パラメータファイルの [file.input.interaction] セクションで行います。

Geometry

格子の幾何情報を記述します。ファイル例を以下に示します。

```
  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
```

基本ベクトル (1〜3 行目)、軌道の数 (4 行目)、各軌道の Wannier center(5 行目以降) を記載します。

Transfer, CoulombIntra, CoulombInter, Hund, etc

Transfer に指定するファイルは、電子系の Transfer に相当する Hamiltonian の係数を格納します。また、二体相互作用の係数は相互作用のタイプごとに係数を格納するファイルを指定します。

相互作用のタイプは、実空間版および波数空間版 UHF の入力ファイル形式と対応して、CoulombItra, CoulombInter, Hund, Ising, Exchange, PairLift, PairHop が定義されています。

これらのファイルは Wannier90(-like) 形式で記述されます。以下に例を示します。

```
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
```

コメント行 (1 行目)、軌道の数 (2 行目)、 nrpts (3 行目)、縮重度 (nrpts 個を 1 行あたり 15 個ずつ)、係数行列の要素を記載します。nrpts は、相互作用定義の並進ベクトルの x,y,z 軸に沿った上限と下限から作られる直方体について、それに含まれるセルの総数です。

行列要素の各行は、並進ベクトル r_x, r_y, r_z 、軌道のインデックス α, β 、係数の値の実部・虚部です。

5.1.3 計算の実行

全ての入力ファイルが準備できた後、プログラムを実行して計算を行います。入力パラメータファイル (ここでは input.toml) を引数とし、ターミナルから H-wave を実行します。

```
$ hwave input.toml
```

計算が開始されると以下のようなログが出力されます。

```
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
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.
```

(次のページに続く)

(前のページからの続き)

```
--------------------------------------------------------------------------------
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
--------------------------------------------------------------------------------
```

入力ファイル読み込みに関するログが出力されたあと、乱雑位相近似計算の計算過程に関する情報が出力されます。出力ファイルは `input.toml` の `[file.output]` セクションの指定に従い、`output` ディレクトリに `chi0q.npz` , `chiq.npz` ファイルが出力されます。

出力ファイルの詳細については **ファイルフォーマット** の章をご覧ください。

計算のポスト処理として、計算結果を可視化するためのツールが `sample/RPA/view.py` に用意されています。このスクリプトファイルを現在のディレクトリにコピーし、ターミナルから以下を実行します。このツールを実行するには `matplotlib` パッケージが必要です。

```
$ python3 view.py
```

`output/chi0q.npz` と `output/chiq.npz` を読み込み、松原振動数 $i\omega_m = 0$ での電荷感受率 $\chi_c(\vec{q})$ およびスピン感受率 $\chi_s(\vec{q})$ の値を各 $\vec{q}$ ごとに標準出力に書き出します。あわせて、これらの値を 3D プロットした以下の図を PNG 形式で出力します。

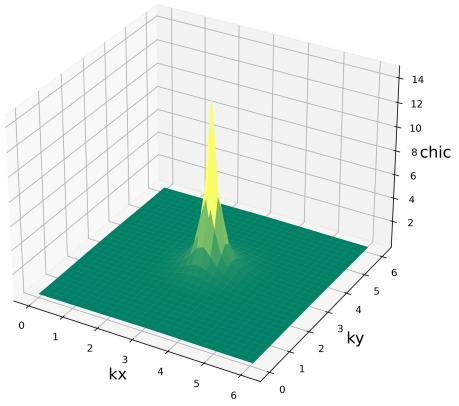

図 5.1: $\chi_c(\vec{q})$

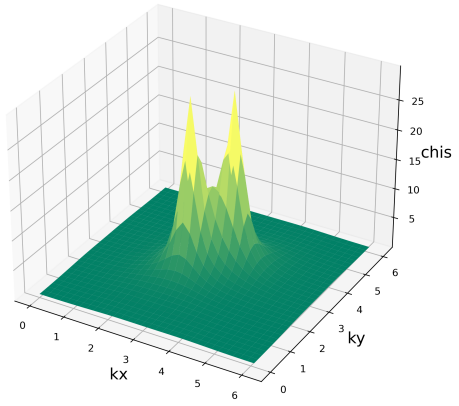

図 5.2: $\chi_s(\vec{q})$

5.2 ファイルフォーマット

5.2.1 環境設定ファイル

このファイルでは、TOML 形式で H-wave に関する環境を設定します。本ファイルは以下の 3 つのセクションから構成されます。

1. mode セクション: 計算モードに関する設定を指定するセクション。
2. log セクション: 標準出力に関する設定を指定するセクション。
3. file セクション: 入出力ファイルのパスなどを設定するセクションで、 input , output のサブセクションから構成される。

以下、ファイル例を記載します。

```
[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"

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

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

(次のページに続く)

(前のページからの続き)

```
chiq = "chiq"
chi0q = "chi0q"
```

ファイル形式

TOML 形式

パラメータ

mode セクション

- mode

  形式 : string 型

  説明 : 計算モードを指定します。乱雑位相近似を利用する場合には "RPA" を記載します。

- enable_spin_orbital

  形式 : bool 型 (デフォルトは false)

  説明 : スピン軌道相互作用を有効にします。Transfer 項の軌道のインデックスはスピン自由度を含む形に解釈されます。インデックスの対応は、軌道 α とスピン s に対して $\alpha + N_{\text{orb}} \cdot s$ となります。

- calc_scheme

  形式 : string 型 (デフォルトは "auto")

  説明 : 軌道・スピンの取り扱い方を指定します。以下のいずれかの値をとります。

  - general: 軌道とスピンを統一した一般化軌道として扱い、感受率行列は最も一般的な形式をとります。行列のサイズは $N_{\text{orb}}^4 N_{\text{spin}}^4 N_k N_\omega$ となります。
  - reduced: 軌道とスピンを統一した一般化軌道として扱い、感受率行列は $\alpha = \alpha', \beta = \beta'$ のみ計算します。行列のサイズは $N_{\text{orb}}^2 N_{\text{spin}}^2 N_k N_\omega$ となります。二体相互作用は CoulombIntra, CoulombInter, Ising, Hund のみを考慮します。
  - squashed: 軌道とスピンを分離し、軌道については $\alpha = \alpha', \beta = \beta'$ のみ扱います。感受率行列のサイズは $N_{\text{orb}}^2 N_{\text{spin}}^4 N_k N_\omega$ となります。詳細は **アルゴリズム** の章を参照してください。
  - auto: 相互作用項の指定に応じて自動判別します。 chi0q のみを計算する場合は指定できません。

mode.param セクション

mode.param セクションでは計算用のパラメータを指定します。

- CellShape

  形式 : int array 型

  説明 : box の形状 Lx, Ly, Lz を指定します。

- SubShape

  形式 : int array 型 (デフォルトは [Lx, Ly, Lz])

  説明 : 副格子の形状 Bx, By, Bz を指定します。

- T

  形式 : float 型 (デフォルトは 0)

  説明 : 温度を指定します。0 以上の値を指定してください。

- mu

  形式 : float 型 または None (デフォルトは None)

  説明 : 化学ポテンシャル μ の値を指定します。指定しない場合は、電子数の期待値が Ncond になるように μ の値を計算します。 mu と Ncond または filling が同時に指定されている場合はエラーで終了します。

- Ncond

  形式 : int 型

  説明 : 伝導電子の数を指定します。1 以上の値を指定してください。

- filling

  形式 : float 型

  説明 : 伝導電子の状態数に対する占有率を指定します。 Ncond と同時に指定されている場合はエラーで終了します。

- Ncond_round_mode

  形式 : str 型 (デフォルトは "strict")

  説明 : 温度 T = 0 の場合、 filling から計算される伝導電子数を整数値に丸める方法を指定します。以下のいずれかの値をとります。

  - as-is : 丸めを行いません。(戻り値は float 型です)
  - round-up : 小数点以下を切り上げます。
  - round-down : 小数点以下を切り捨てます。
  - round-off : 小数点以下を四捨五入します。
  - round : 小数点以下を round 関数で丸めます。0.5 は 0 に丸められるので注意。

- `strict` : 整数でない場合はエラーで終了します。
- `exact` : 整数でない場合は warning を表示し、 `round` で丸めた整数値を返します。

- `Nmat`

  形式 : int 型 (デフォルトは 1024)

  説明 : 松原振動数のカットオフを指定します。1 以上の偶数の値を指定してください。松原振動数の定義は以下の通りです。インデックス n は 0〜`Nmat-1` の値を取ります。

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

- `coeff_tail`

  形式 : float 型 (デフォルトは 0.0)

  説明 : フーリエ変換の尾部の補正を行う際の、補正の大きさを指定します。対角化された一体 Green 関数に対して $\texttt{coeff\_tail}/(i\omega_n)$ を引き虚時間表示にフーリエ変換した後、$-\beta/2 \cdot$ `coeff_tail` を付加します。

- `matsubara_frequency`

  形式 : int 型, list 型, または str 型 (デフォルトは `"all"`)

  説明 : 感受率行列 $\chi(\vec{q})$ を計算する松原振動数のインデックスを指定します。指定方法は以下のいずれかです。

  - 整数値 : 指定した松原振動数における値を計算します。
  - `[ min, max (, step) ]` : `min` から `max` まで `step` おきに計算します。 `step` を省略した場合は `step=1` として扱われます。
  - `all` : 全ての松原振動数での値を計算します。
  - `center` : `Nmat/2` における値を計算します。
  - none : 計算しません。

  感受率行列 $\chi(\vec{q})$ および既約感受率行列 $\chi_0(\vec{q})$ をファイルに出力する場合、このパラメータで指定した松原振動数における値が出力されます。

- `coeff_extern`

  形式 : float 型 (デフォルトは 0.0)

  説明 : 外場の係数 h を指定します。外場は $\pm h H_{\alpha\beta}(r_{ij})$ の形式で導入され、行列要素 $H_{\alpha\beta}(r_{ij})$ の定義は相互作用入力ファイルで与えます。符号は $+(-)$ が spin up(down) です。

- `RndSeed`

  形式 : int 型 (デフォルトは 1234)

  説明 : 乱数のシード (種) を指定します。

- ene_cutoff

  形式 : float 型 (デフォルトは 100.0)

  説明 : Fermi 分布関数を計算する際に overflow を避けるため、e の冪指数に対する上限を指定します。

log セクション

- print_level

  形式 : int 型 (デフォルトは 1)

  説明 : 標準出力のレベルを指定します。1 にすると詳細な情報が出力されます。

file セクション

input と output のサブセクションからなります。入力ファイルおよび出力ファイルについて、ファイルの種別、格納するディレクトリやファイル名などの情報を指定します。以下、順に説明します。

file.input セクション

- path_to_input

  形式 : str 型 (デフォルトは "" (空文字列))

  説明 : 入力ファイルの格納されているディレクトリを指定します。

- chi0q_init

  形式 : str 型

  説明 : 計算済みの既約感受率 $\chi_0(\vec{q})$ を利用して感受率を計算する場合に利用します。既約感受率のデータを格納したファイルの名前を指定します。ファイルは Numpy zip 形式です。file.output の chi0q の出力ファイル形式に対応しています。

- trans_mod

  形式 : str 型

  説明 : UHF 法で近似した相互作用項を含む一体項を RPA の初期値として利用します。UHFk の file.output.rpa に指定した出力ファイルのファイル名を指定します。

- green_init

  形式 : str 型

  説明 : RPA の初期値として与える Green 関数を格納したファイルの名前を指定します。UHFk の green の出力ファイル形式に対応しています。 trans_mod が指定されている場合、 green_init は使われません。

file.input.interaction セクション

乱雑位相近似で、幾何情報や相互作用のタイプと定義ファイルとの対応付けを行います。

- path_to_input

  形式 : str 型

  説明 : 入力ファイルを格納するディレクトリを指定します。 file.input セクションの path_to_input とは独立に指定できます。

- Geometry

  形式 : str 型

  説明 : 幾何情報のファイル名を指定します。

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

  形式 : str 型

  説明 : 各相互作用の係数ファイル名を指定します。二体相互作用項 (CoulombIntra, CoulombInter, Hund, Ising, Exchange, PairLift, PairHop) の指定がない場合は、chi0q のみを計算して終了します。

file.output セクション

- path_to_output

  形式 : str 型 (デフォルトは "output")

  説明 : 出力ファイルを格納するディレクトリを指定します。

- chi0q

  形式 : str 型

  説明 : 既約感受率行列 $\chi_0(\vec{q})$ を出力するファイル名を指定します。このキーワードがない場合には情報は出力されません。

- chiq

  形式 : str 型

  説明 : 感受率行列 $\chi(\vec{q})$ を出力するファイル名を指定します。このキーワードがない場合には情報は出力されません。

5.2.2 RPA 用入力ファイル

乱雑位相近似で使用する入力ファイルに関して説明します。入力ファイルの種別は以下の 2 つに分類され、データフォーマットは Wannier90 形式です。

(1) 幾何情報:

Geometry: 格子の幾何情報を設定します。

(2) 相互作用定義:

RPA のハミルトニアンを電子系の表式により指定します。具体的には、キーワードで指定する各相互作用のタイプについて、ハミルトニアンの係数をデータとして与えます。

$\mathcal{H}\Phi$ や mVMC の Expert Mode 入力に相当する以下のキーワードを指定できます。

Transfer: $c^\dagger_{i\sigma_1}c_{j\sigma_2}$ で表される一体項を指定します。

Extern: $\sigma^z_{\sigma_1\sigma_2}c^\dagger_i c_j$ で表される一体項に対する外場を指定します。

CoulombIntra: $n_{i\uparrow}n_{i\downarrow}$ で表される相互作用を指定します ($n_{i\sigma} = c^\dagger_{i\sigma}c_{i\sigma}$)。

CoulombInter: $n_i n_j$ で表される相互作用を指定します ($n_i = n_{i\uparrow} + n_{i\downarrow}$)。

Hund: $n_{i\uparrow}n_{j\uparrow} + n_{i\downarrow}n_{j\downarrow}$ で表される相互作用を指定します。

Ising: $S^z_i S^z_j$ で表される相互作用を指定します。

Exchange: $S^+_i S^-_j$ で表される相互作用を指定します。

PairLift: $c^\dagger_{i\uparrow}c_{i\downarrow}c^\dagger_{j\uparrow}c_{j\downarrow}$ で表される相互作用を指定します。

PairHop: $c^\dagger_{i\uparrow}c_{j\uparrow}c^\dagger_{i\downarrow}c_{j\downarrow}$ で表される相互作用を指定します。

以下のセクションでデータフォーマットについて記述します。

幾何情報指定ファイル

格子の幾何情報を記述します。ファイル例を以下に示します。

```
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
```

(次のページに続く)

(前のページからの続き)

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

ファイル形式

- 1-3 行: `[ax_i] [ay_i] [az_i]`
- 4 行: `[Norbit]`
- 5 行以降: `[vx_i] [vy_i] [vz_i]`

パラメータ

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

  形式 : float 型

  説明 : 基本格子ベクトル $\vec{a}_1, \vec{a}_2, \vec{a}_3$ を指定します。

- `[Norbit]`

  形式 : int 型

  説明 : ユニットセル内の軌道の数 N_{orbit} を指定します。

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

  形式 : float 型

  説明 : 各軌道の Wannier center $\vec{v}_i$ を fractional coordinate 表記で指定します。

使用ルール

本ファイルを使用するにあたってのルールは以下の通りです。

- 軌道のインデックスは暗黙的に Wannier center の並び順に 1〜N_{orbit} で与えられます。

相互作用指定ファイル

次の形で表わされるハミルトニアンの一体項および二体相互作用項について、その係数 $T_{\alpha\beta}(r_{ij})$ および $J_{\alpha\beta}(r_{ij})$, $V_{\alpha\beta}(r_{ij})$, U_α を共通の Wannier90(-like) 形式で記述します。なお、乱雑位相近似では一般化二体相互作用 InterAll 形式には対応していません。

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

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

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

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

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

PairHop: $\sum_{ij\alpha\beta} J^{\mathrm{PH}}_{\alpha\beta}(r_{ij})\, c^\dagger_{i\alpha\uparrow} c_{j\beta\uparrow} c^\dagger_{i\alpha\downarrow} c_{j\beta\downarrow} + h.c.$

Exchange: $\sum_{ij\alpha\beta} J^{\mathrm{Ex}}_{\alpha\beta}(r_{ij})\, c^\dagger_{i\alpha\uparrow} c_{j\beta\uparrow} c^\dagger_{j\beta\downarrow} c_{i\alpha\downarrow}$

PairLift: $\sum_{ij\alpha\beta} J^{\mathrm{PairLift}}_{\alpha\beta}(r_{ij})\, c^\dagger_{i\alpha\uparrow} c_{i\alpha\downarrow} c^\dagger_{j\beta\uparrow} c_{j\beta\downarrow}$

以下にファイル例を示します。

```
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
...(略)
 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
...(略)
```

ファイル形式

- 1 行: ヘッダ (何が書かれても問題ありません)。
- 2 行: `[Norbit]`
- 3 行: `[Npts]`
- 4 - $\lceil N_{\mathrm{pts}}/15 \rceil + 3$ 行: `[n_1] [n_2] ...`
- $\lceil N_{\mathrm{pts}}/15 \rceil + 4$ 行以降: `[r_x] [r_x] [r_x] [alpha] [beta] [J.real] [J.imag]`

パラメータ

- `[Norbit]`

  形式 : int 型

  説明 : ユニットセル内の軌道の数 N_{orbit} を指定します。

- `[Npts]`

  形式 : int 型

  説明 : 並進ベクトル全体が入る直方体に含まれるセルの数を指定します。

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

  形式 : int 型

  説明 : 各セルの縮重度を指定します (通常は 1)。一行あたり 15 点を列挙します。

- `[r_x]`, `[r_y]`, `[r_z]`

  形式 : int 型

  説明 : 並進ベクトルを指定します。

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

  形式 : int 型

  説明 : 軌道のインデックスを指定します。`[alpha]` が元のセル内の軌道、`[beta]` が $\vec{r}$ 離れたセル内の軌道を指します。

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

  形式 : float 型

  説明 : 係数 $J_{\alpha\beta}(\vec{r})$ の実部と虚部を指定します。

使用ルール

本ファイルを使用するにあたってのルールは以下の通りです。

- 行数固定で読み込みを行うため、ヘッダの省略はできません。
- 係数行列のうち、省略された要素は 0 と仮定します。
- 並進ベクトルは全て `CellShape` 内に収まるとします。`r_x`, `r_y`, `r_z` の範囲が `CellShape` の x,y,z 軸のサイズを超える場合はエラーで終了します。
- `mode.enable_spin_orbital` が `true` の場合、Transfer 項の軌道のインデックスはスピン自由度を含む一般化軌道インデックスと読み替え、1〜 $2N_{\text{orbital}}$ の値をとります。1〜 N_{orbital} が spin up、$N_{\text{orbital}}+1$ 〜 $2N_{\text{orbital}}$ が spin down に対応します。 `mode.enable_spin_orbital` が `false` の場合は、インデックスの範囲が 1〜 N_{orbital} の行のみ考慮します。

5.2.3 RPA の出力ファイル

RPA の出力ファイルに関するファイル形式を記載します。

chiq, chi0q

感受率行列および既約感受率行列の計算結果を Numpy zip 形式で出力します。ファイル名は、環境設定ファイルの中の `file.output` セクションでキーワード `chiq` および `chi0q` を用いて指定された文字列 (以下 `chiq_str` と記述) を用いて、 `chiq_str.npz` という名前で出力されます。

ファイルの内容は、以下のキーにバインドされる複数の配列データからなります。

- `chiq` または `chi0q`:

  感受率行列または既約感受率行列のデータ。データ形式は以下の節で説明します。

- `freq_index`:

  出力する松原振動数の値または範囲はパラメータ `matsubara_frequency` で指定されます。出力データの配列インデックスと、実際の松原振動数のラベルの対応付けを `freq_index` に格納します。

- `wavevector_unit` および `wavevector_index`:

  波数ベクトルの情報を格納します。詳細は *UHFk* **の出力ファイル** を参照してください。

副格子を指定している場合は、出力されるデータは副格子を単位とした感受率の値です。波数ベクトルおよび軌道のインデックスは副格子に読み替えます。

`chi0q` の出力ファイルは、計算済み既約感受率データとして `file.input` セクションの `chi0q_init` に指定して使用できます。

chi0q のデータ形式

`chi0q` のデータ形式は、スピン軌道相互作用および外場の有無と、 `mode.calc_scheme` パラメータの値によって以下の形をとります。

- スピン非依存 (spin-free):

  `enable_spin_orbital` パラメータが `false` の場合、もしくは `true` であっても $T_{\tilde{\alpha}\tilde{\beta}}(k)$ がスピンについて対角かつ対称な場合で、外場がないとき、既約感受率行列のスピン非依存部分を出力します。

  - `calc_scheme = general` の場合、データ配列は `ndarray(l,q,a,ap,b,bp)` で、インデックスは以下のとおりです

    * `l`: 松原振動数のラベル。インデックスとラベルの対応は前述の `freq_index` で与えられます。
    * `q` : 波数ベクトルのインデックス $[q_x\ q_y\ q_z]$ を 1 次元化したインデックスで、 $q = q_z + N_z \cdot (q_y + N_y \cdot q_x)$ となります。
    * `a`, `ap`, `b`, `bp` はスピンを含まない軌道インデックス α, α', β, β' に対応します。

  - `calc_scheme = reduced` または `squashed` の場合、データ配列は `ndarray(l,q,a,b)` です。インデックスの意味は上記と同じです。

- スピン対角 (spin-diagonal):

  `enable_spin_orbital` パラメータが `false` で外場がある場合、もしくは `enable_spin_orbital` パラメータが `true` で $T_{\tilde{\alpha}\tilde{\beta}}(k)$ がスピンについて対角な場合に、既約感受率行列の spin up/down 成分を出力します。

  - `calc_scheme = general` の場合、データ配列は `ndarray(s,l,q,a,ap,b,bp)` となります。 `s = 0 (1)` はそれぞれ spin up (down) 成分を表し、それ以外のインデックスは上記と同じです。

  - `calc_scheme = reduced` または `squashed` の場合、データ配列は `ndarray(s,l,q,a,b)` となります。

- スピン依存 (spinful):

  `enable_spin_orbital` パラメータが `true` で Transfer 項が一般的な形の場合、一般化軌道をインデックスとする既約感受率行列を出力します。

  - `calc_scheme = general` の場合、データ配列は `ndarray(l,q,a,ap,b,bp)` となります。 `a`, `ap`, `b`, `bp` はスピンを含む一般化軌道インデックス $\tilde{\alpha}, \tilde{\alpha}', \tilde{\beta}, \tilde{\beta}'$ に対応します。

  - `calc_scheme = reduced` の場合、データ配列は `ndarray(l,q,a,b)` となります。 `a`, `b` はスピンを含む一般化軌道インデックス $\tilde{\alpha}, \tilde{\beta}$ に対応します。

chiq のデータ形式

`chiq` のデータ形式は `calc_scheme` パラメータの値によって以下の形をとります。

- `calc_scheme = general` の場合、データ配列は `ndarray(l,q,a,ap,b,bp)` となります。 `a`, `ap`, `b`, `bp` はスピンを含む一般化軌道インデックス $\tilde{\alpha}, \tilde{\alpha}', \tilde{\beta}, \tilde{\beta}'$ に対応します。

- `calc_scheme = reduced` の場合、データ配列は `ndarray(l,q,a,b)` となります。 `a`, `b` はスピンを含む一般化軌道インデックス $\tilde{\alpha}, \tilde{\beta}$ に対応します。

- `calc_scheme = squashed` の場合、データ配列は `ndarray(l,q,s1,s2,a,s3,s4,b)` となります。 `a`, `b` は軌道インデックス α, β に対応し、 `s1` 〜 `s4` はスピンインデックス $\sigma, \sigma', \sigma_1, \sigma_1'$ にそれぞれ対応します。

データ読み込みの例

以下にデータの読み込み方の例を示します。

```
import numpy as np
data = np.load('output/chiq.npz')

chiq = data['chiq']
freq_index = data['freq_index']
```

第6章　アルゴリズム

6.1 非制限 Hartree-Fock 法

6.1.1 概要

非制限 Hartree-Fock(UHF) 近似では一体の演算子の揺らぎについて一次のみを取り入れ、二体項を一体項へと近似します。一般的な二体相互作用については以下の近似を行うことに相当します。

$$c_i^\dagger c_j^\dagger c_k c_l \sim \langle c_i^\dagger c_l\rangle c_j^\dagger c_k + c_i^\dagger c_l \langle c_j^\dagger c_k\rangle - \langle c_i^\dagger c_k\rangle c_j^\dagger c_l - c_i^\dagger c_k \langle c_j^\dagger c_l\rangle \\ - (\langle c_i^\dagger c_l\rangle\langle c_j^\dagger c_k\rangle - \langle c_i^\dagger c_k\rangle\langle c_j^\dagger c_l\rangle)$$

H-wave では以下の形式で二体相互作用を定義しています。

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

ここで、上式の第 2 項のように一体項が存在することに注意が必要です。さて、一体項で与えられたハミルトニアンは一般的に以下のように書けます。

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

ここで、簡単化のため、 $i \equiv (i,\alpha,\sigma_1), j \equiv (j,\beta,\sigma_2)$ 、 H は H_{ij} を成分に持つ行列、 $\hat{c}$ は c_i を成分にもつ行ベクトルを表します。このとき、 H はエルミート行列なので、 $\hat{\xi}$ を H の固有値を対角成分に持つ行列、U は各固有ベクトルに対応する行列として、$H = U\hat{\xi}U^\dagger$ のように変形できることから、$\hat{d} = U^\dagger \hat{c}$ とすると、

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

のように書き直すことが出来ます。したがって、UHF 近似の一体相互作用からくるエネルギーは

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

として求められます。実際の数値計算では、H は UHF 近似を通して一体グリーン関数 $\langle c_i^\dagger c_j\rangle$ に依存するため、一体グリーン関数を最初に初期値として与え、

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

の関係から一体グリーン関数を更新し、一体グリーン関数が収束するまで計算を繰り返します。ただし、上式において β は逆温度 $1/k_BT$, μ は化学ポテンシャルとしました。なお、粒子数を固定するカノニカル計算では、粒子数を N とした場合に、

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

を満たすように、各ステップで μ を決定・更新しながら計算を進めます。H-wave では、更新に対しては simple-mixing を現在行っています。simple-mixing では n 番目のステップの一体グリーン関数を $\langle c_i^\dagger c_j \rangle^{(n)}$ とした場合に、 n 番目の一体グリーン関数と $n+1$ 番目のステップで求められた一体グリーン関数を混ぜた上で更新を行います：

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

ここで、 α は 0 から 1 までの定数を表します。現在の H-wave では実装していませんが、simple-mixing 以外の更新方法としては、Anderson-mixing などもあります。なお、H-wave の実空間 UHF では、全ての相互作用を InterAll 形式にマップした後に解析を行う仕様になっています。有限温度の自由エネルギーは

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

で与えられます。

6.1.2 波数空間への拡張

一体項で与えられたハミルトニアンについて、Fourier 変換 $c_i = \frac{1}{\sqrt{V}}\sum_k e^{ikr_i} c_k$ を用いて波数表示に書き換えます。

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

相互作用は並進対称性を仮定し、係数 $J_{ij\alpha\beta}$ は空間座標について並進ベクトル $r_{ij} = r_j - r_i$ のみに依存するとします。なお、波数空間版 UHF では一般化された InterAll 形式の相互作用は扱いません。

波数空間表示のハミルトニアンは波数 k について対角的なので、固有値・固有ベクトルの計算は、$N_{\text{site}}N_{\text{orbit}} \times N_{\text{site}}N_{\text{orbit}}$ 行列の対角化から、N_{site} 個の $N_{\text{orbit}} \times N_{\text{orbit}}$ 行列の対角化に計算量を抑えることができます。ここで N_{site} はサイト数、 N_{orbit} はスピン自由度も含めた 1 サイト当たりの軌道縮緘度です。

6.2 乱雑位相近似法

乱雑位相近似 (RPA) では相互作用のない状態を出発点に、電子相関効果による一体の演算子の揺らぎの応答を検出します。UHF 近似ではあらかじめ初期配置を予想しておく必要があるのに対して、RPA 法では 2 次転移により生じる秩序相を推定することが可能です。H-wave では松原振動数を利用した RPA 法を実装しており、解析接続によって実験で観測される動的な物理量との比較も行うことが可能です。

以下、アルゴリズムについて掲載します。H-wave の RPA モードでは以下の Hamiltonian を取り扱います。

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

ここで、以下のフーリエ変換

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

を行うと、Hamiltonian は以下のように書き換えられます。

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

RPA では $\mathcal{H}_0$ に対して、電子間相互作用を介した密度揺らぎの効果を考慮します。具体的には、 $\mathcal{H}_0$ が対角化されるような軌道・スピンの混成基底を用いて、相互作用の項を以下のように近似します。

$$\begin{aligned}&W^{\beta\beta',\alpha\alpha'}_{\mathbf{q}} c^\dagger_{\mathbf{k}+\mathbf{q},\alpha}c_{\mathbf{k},\alpha'}c^\dagger_{\mathbf{k}'-\mathbf{q},\beta'}c_{\mathbf{k}',\beta} \\ &\sim W^{\beta\beta',\alpha\alpha'}_{\mathbf{q}}\sum_{\gamma,\gamma'}(u^*_{\alpha\gamma,\mathbf{k}+\mathbf{q}}d^\dagger_{\mathbf{k}+\mathbf{q},\gamma}u_{\alpha'\gamma,\mathbf{k}}d_{\mathbf{k},\gamma})(u^*_{\beta'\gamma',\mathbf{k}'-\mathbf{q}}d^\dagger_{\mathbf{k}'-\mathbf{q},\gamma'}u_{\beta\gamma',\mathbf{k}'}d_{\mathbf{k}',\gamma'}).\end{aligned}$$

ここで、

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

であり、 $d_{\mathbf{k},\gamma}$ は $\mathcal{H}_0$ を対角化する消滅演算子を表します (γ は固有値の index)。このとき、一体の既約グリーン関数は以下のように与えられます。

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

既約感受率は対角化された成分で閉じる必要があるため、以下のように与えられます。

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

この既約感受率を用いることで、RPA で得られる感受率が以下のように得られます。

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

ここで、 $\alpha\alpha'$ などをまとめて一つの index にすると行列形式で表すことができ、最終的には以下のような形式で感受率を得ることができます。

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

上記の実装では、軌道とスピンを統一した一般化軌道として取り扱いました。計算の実行に必要な配列のうち、 感受率 ($X^{(0)\alpha\alpha',\beta\beta'}(\mathbf{q}, i\omega_n), X^{\alpha\alpha',\beta\beta'}(\mathbf{q}, i\omega_n)$) が一番大きなサイズの多次元配列となり、そのサイズは $N^4_{\text{orb}}N^4_{\text{spin}}N_kN_\omega$ で与えられ、サイズが大きくなるとメモリコスト、計算量が増大します。以下で説明するように、軌道とスピンを分離することで感受率の多次元配列のサイズを減らすことができます。H-wave の RPA モードで取り扱う二体相互作用では、軌道とスピンを分離することで、

$$W^{\beta\sigma_1\sigma_1',\alpha\sigma\sigma'}_{\mathbf{q}} c^\dagger_{\mathbf{k}+\mathbf{q},\alpha\sigma}c_{\mathbf{k},\alpha\sigma'}c^\dagger_{\mathbf{k}'-\mathbf{q},\beta\sigma_1'}c_{\mathbf{k}',\beta\sigma_1} \tag{6.15}$$

と書けます。軌道に対しては同一の軌道での散乱となるため、既約感受率は

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

となり、 $N^2_{\text{orb}}N^4_{\text{spin}}N_kN_\omega$ にサイズを抑えることができます。このとき、RPA で得られる感受率は

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

となります。 $\alpha\sigma\sigma'$ を一つの index とみなせば、行列形式にすることができ、一般化軌道の場合と同様に、

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

と書けることがわかります。以上が一般的な RPA の定式化になります。

上述の近似では既約感受率の計算を

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

として行っています。この場合、対角化した成分の和が必要となり、計算コストが多くかかってしまいます。そのため、先行研究の多くは一体グリーン関数を

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

のように近似し、既約感受率を

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

として計算して高速化する場合が多いです。この既約感受率を用いた計算では、対角化成分が混在してしまう状況で近似精度が悪くなりますが、バンド交差による γ への技術的な対応を行う必要がないというメリットもあります。先行研究との比較をするためにも、H-Wave ではこの手法を採用しています (グリーン関数と既約感受率を正しく取り扱うモードについても実装する予定です)。なお、より高次な相関効果を考慮する手法として vertex 補正の考慮などがあります。詳細については、例えばこちらの文献[1] を参考にしてください。

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

第7章　謝辞

H-wave は東京大学物性研究所ソフトウェア高度化プロジェクト (2022 年度) の支援の下に開発されました。

付録A　Appendix

A.1 StdFaceを用いた相互作用ファイルの作成

A.1.1 StdFaceライブラリのコンパイル

H-waveで使用する相互作用定義ファイルは、StdFaceライブラリを使用することで簡単に作成することができます。以下ではStdFaceライブラリを使用した入力ファイルの作成を行う方法を記載します。

H-waveに対応したStdFaceライブラリは以下から取得できます。

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

ダウンロード終了後、以下のコマンドでコンパイルを行います。

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

コンパイルに成功すると、 src ディレクトリに実行ファイル hwave_dry.out ができます。

A.1.2 StdFaceライブラリの使用

hwave_dry.out の入力ファイルはサンプルディレクトリ内の stan.in ファイルにそれぞれ置いてあります。ファイルの内容は以下のとおりです。

```
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 は対象となる模型を指定するキーワードです。現状では電子数を固定した Hubbard 模型 Hubbard のみに対応しています。
- lattice は結晶構造を指定するキーワードです。 ここでは正方格子 square を選択しています。 W, L は格子のサイズです。
- t はホッピング、 V は隣接サイトクーロン相互作用のパラメータです。
- calcmode は出力形式を指定します。"uhfk" または "rpa" を指定した場合は Wannier90(-like) 形式の入力ファイルを生成します。"uhfr" を指定した場合は H-wave の実空間 UHF プログラム UHFr 向けの入力ファイルを出力します。デフォルトは calcmode = "uhfk" です。
- exportall = 0 は Wannier90(-like) 形式の出力をコンパクトにするオプションです。

入力ファイルの詳細については、*UHFr* **用入力ファイル** , *UHFk* **用入力ファイル** , *RPA* **用入力ファイル** のセクションを参照してください。

上記のファイルを入力ファイルとして、 hwave_dry.out を以下のコマンドで実行します。

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

実行終了後、実行ディレクトリに幾何情報ファイル geom.dat 、相互作用定義ファイル transfer.dat, coulombinter.dat が生成されます。StdFace ライブラリで指定できる格子や相互作用の種類については、H Φもしくは mVMC のマニュアルにあるスタンダードモードの入力ファイル形式を参照してください。

A.2 エラーメッセージ一覧

- mode is not defined in [mode].

  説明 : パラメータファイルの [mode] セクションに mode パラメータが指定されていない。

  モード : main

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

  説明 : get_param() の引数が不正

  モード : UHFr (read_input)

- duplicate items found in *file*

  説明 : *file* に重複するエントリがある

  モード : UHFr (read_input)

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

  説明 : 入力ファイルの行数とファイル内の行数指定が異なっている

  モード : UHFr (read_input)

- `Unknown keyword` *keyword*

  説明： `[file.input.interaction]` に不明なキーワードがある

  モード： UHFk (read_input_k)

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

  説明： `initial` と `initial_uhf` は同時に指定できない

  モード： UHFk (read_input_k)

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

  説明： *file* が存在しない

  モード： UHFk (read_input_k)

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

  説明： `get_param()` の引数が不正

  モード： UHFk (read_input_k)

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

  説明： `Geometry` に指定されているファイル *file* が存在しない

  モード： UHFk (wan90)

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

  説明： パラメータ `2Sz` と `Ncond` の偶奇が一致していない

  モード： solver base

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

  説明： *type* の値が範囲外

  モード： UHFk

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

  説明： 相互作用項の並進ベクトルが CellShape 内に収まっていない

  モード： UHFk

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

  説明： Transfer 項が Hermite でない

  モード： UHFk

- `Parameter range check failed for param_mod.`

  説明： [mode.param] のパラメータの値が範囲外

  モード： solver base

- `Parameter check failed for param_mod.`

  説明： [mode.param] のパラメータが不正

  モード： solver base

- `Hermite check failed for` *type*

  説明： *type* が Hermite でない

  モード： UHFr

- `Parameter check failed for info_mode.`

  説明： [mode] のパラメータが不正

  モード： solver base

- `value not integer`

  説明： パラメータの値が整数ではない

  モード： RPA

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

  説明： [mode.param] に `CellShape` が指定されていない

  モード： RPA

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

  説明： `Ncond` に 0 以上の値が指定されていない

  モード： RPA

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

  説明： `Nmat` に 0 以上の値が指定されていない

  モード： RPA

- `RPA._find_mu:  not converged. abort`

  説明： `mu` の計算が収束しなかった

  モード： RPA

- `SubShape is not compatible with CellShape.`

  説明： 副格子の指定が不正

  モード： RPA

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

  説明： `T` に 0 以上の値が指定されていない

  モード： RPA

- `both mu and Ncond or filling are specified`

  説明： `mu` と ``Ncond または `filling` が同時に指定されている

  モード： RPA

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

  説明： `CellShape` の指定が不正

  モード： RPA

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

  説明： 副格子の指定が不正

  モード： RPA

- `invalid CellShape.`

  説明： CellShape の指定が不正

  モード： RPA

- `invalid SubShape.`

  説明： SubShape の指定が不正

  モード： RPA

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

  説明： `mu` または `Ncond`, `filling` のいずれも指定されていない

  モード： RPA

- `read_chi0q failed:` *info*

  説明： `chi0q` の読み込みに問題があった

  モード： RPA

- `round_to_int: unknown mode` *mode*

  説明： 丸めモードの指定が不正

  モード： RPA

- `unexpected data size` *error*

  説明： データサイズが不正

  モード： RPA

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

  説明： `[mode]` に mode パラメータが指定されていない

  モード： RPA

- orbital index check failed for *type*

  説明 : 軌道のインデックスが不正

  モード : UHFk

- initial green function in coord space requires geometry.dat

  説明 : 実空間でのグリーン関数の読み込みは geometry.dat を同時に指定する必要がある

  モード : UHFk

- CellShape is missing. abort

  説明 : CellShape パラメータが指定されていない

  モード : UHFk

- Ncond or Nelec is missing. abort

  説明 : Ncond または Nelec パラメータが指定されていない

  モード : UHFk

- SubShape is not compatible with CellShape. abort

  説明 : 副格子の指定が不正

  モード : UHFk

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

  説明 : 相互作用定義ファイルの軌道インデックスが不正

  モード : UHFk

- _save_greenone: onebodyg_uhf and geometry_uhf are required

  説明 : onebodyg_uhf と geometry_uhf が指定されていない

  モード : UHFk

- find mu: not converged. abort

  説明 : mu の計算が収束しなかった

  モード : UHFk

- range check failed for Initial

  説明 : Initial の指定が不正

  モード : UHFr

- OneBodyG is required to output green function.

  説明 : グリーン関数の出力のための OneBodyG の指定がない

  モード : UHFr

- `hermite check failed for Initial`

  説明： `Initial` が Hermite でない

  モード： UHFr

- `Range check failed for Transfer`

  説明： `Transfer` 定義ファイルのインデックスが範囲外

  モード： UHFr

- `Range check failed for` *type*

  説明： *type* 定義ファイルのインデックスが範囲外

  モード： UHFr

- `parameter range check failed.`

  説明： パラメータの値が不正

  モード： UHFr

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

  説明： `mode` の指定が不正

  モード： UHFr

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

  説明： パラメータ *key* の値が不正

  モード： solver base [warning]

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

  説明： パラメータ *key* の値が不正

  モード： solver base [warning]

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

  説明： パラメータ *key* が設定されていない

  モード： solver base [warning]

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

  説明： `[mode]` セクションの `mode` パラメータの値が不正

  モード： solver base [warning]

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

  説明： `[mode]` セクションに `mode` パラメータが指定されていない

  モード： solver base [warning]

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

  説明： `TRUST-ME` モードが有効。パラメータのチェックを行わない

  モード： solver base [warning]

- `value not integer`

  説明： 設定値が整数でない

  モード： RPA [warning]

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

  説明： `mode` パラメータの値が不正

  モード： RPA [warning]

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

  説明： f_{ij} の計算で `2Sz` は偶数でなければならない

  モード： UHFr [warning]

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

  説明： f_{ij} の計算で `Ne` は偶数でなければならない

  モード： UHFr [warning]

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

  説明： f_{ij} の計算で `Sz` が 0 以外の偶数の場合は未サポート

  モード： UHFr [warning]

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

  説明： キーワード *key* が不正

  モード： UHFr [warning]

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

  説明： UHFr の計算が収束しなかった

  モード： UHFr [warning]