物理計算工学研究グループ資料

WaveTool マニュアル

目次

図目次

1　波動関数グリッドデータ作成ツール (WaveTool)

1.1　WaveTool とは

Wave Tool とは、量子力学シミュレータの出力データ (JSON 形式) からグリッドデータファイル (GaussianCube 形式) を作成するツールである

1.2　WaveTool の流れ

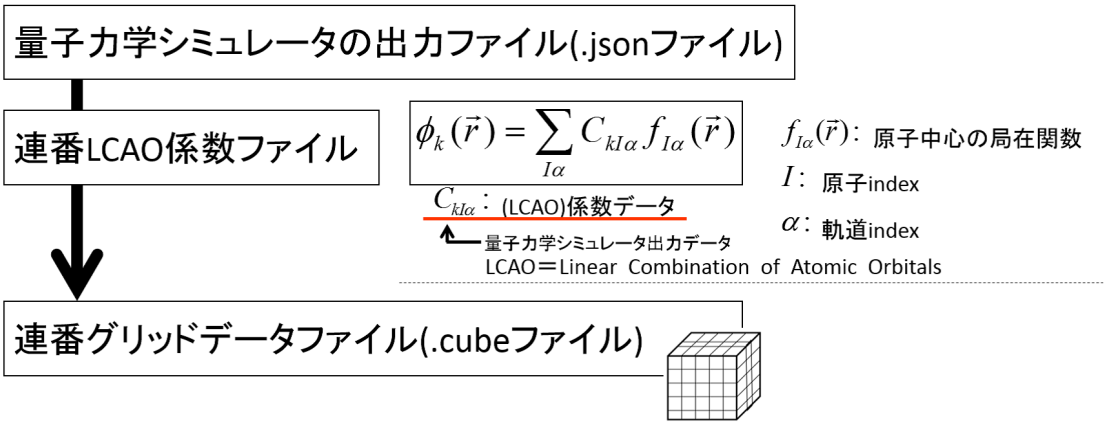

図 1　Wave Tool のワークフロー

具体的に行っている作業を簡単に説明すると、まず量子力学シミュレータの出力ファイル (JSON 形式) から、時系列データを取り出している。次に取り出した時系列データを LCAO 係数ファイルへ変換している。最後に LCAO 係数ファイルから elses-generate-cubefile(elses に付属している LCAO ファイルから CUBE ファイルを作成するツール) を使用して CUBE ファイルを作成している。

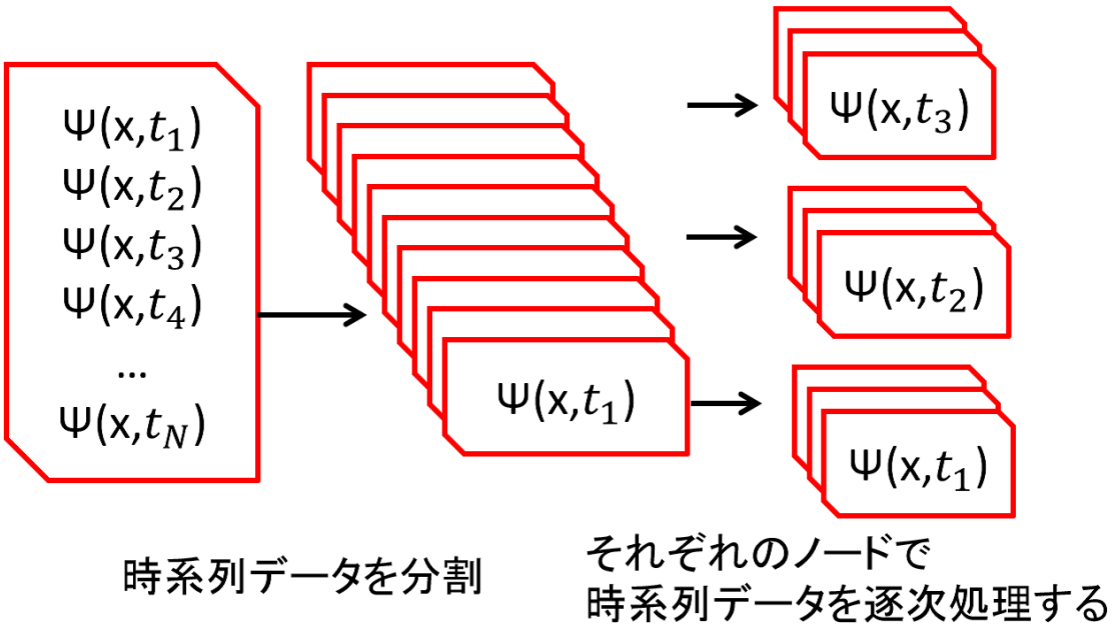

図 2　Wave Tool の並列イメージ

主に、並列化が行われている部分は LCAO 係数ファイルを作成した後の LCAO 係数ファイルから CUBE ファイルを作成する部分である。具体的には、LCAO ファイルを作成した後の作業は、1 つ 1 つが完全に独立しているため行う作業を並列数分で分割してそれぞれで逐次処理を行っている。
ちなみに、LCAO 係数ファイルから CUBE ファイルを作成する部分は Python が fortran のコードを実行して計算を行っている形になっていて fortran のコードが OpenMP 並列化されている。

1.3 WaveTool に必要な環境・入力ファイル

1.3.1 WaveTool に必要な環境

必須

- linux
- python2.7

mpi 並列で使用する場合 (必須ではない)

- mpi4py(python の library)
  インストール方法については付録 B を参照。

1.3.2 WaveTool に入力ファイル

WaveTool では以下の 4 つの入力ファイルが必須です。

- elses-generate-cubefile
  elses の付属ソフト
- JSON ファイル
  計算時に出力される JSON ファイル。以下のようなファイル全てが必要です

```
out.json
out_000001-000100.json
out_000001-000100.json.dat
```

- XYZ ファイル
  計算時の構造ファイル。(計算中全ての構造が必要になります)
  以下のコマンドでも作成できます。

```
$ python extract.py --type xyz out.json
```

- basis_infomation ファイル
  計算に使用されている基底情報が書かれたファイル。
  以下のように、output タグに basis_info を追加すると出力されます。

```
<output>
  <basis_info filename="output_basis_information.txt" />
```

```
</output>
```

又は、ouput_wavefunction.txt でも代用可能です。

1.4 WaveTool の構成

WaveTool には以下の 9 個のファイルがある。

- Main_AutoWave.py
  ⇒ WaveTool のメインプログラムである。Windows 上で実行した場合 OSError を起こすため、Linux 上で実行すること。

- LCAO_converter.py
  ⇒ ELSES の elses-generate-cubefile を利用して CUBE ファイルを作成する。

- charge_makes_cube.py
  ⇒ real の CUBE ファイルと imaginary の CUBE ファイルから charge の CUBE ファイルを作成する。

- input_mesh_grid_lib.py
  ⇒ input_mesh_grid を作成する。

- make_input_mesh_grid_20151013.py
  ⇒ input_mesh_grid を自動生成するツール。

- Main_AutoWave_not_LCAO_20151022.py
  ⇒ LCAO 係数ファイルを作成せずに cube ファイルを作成するツール。1 度 cube ファイルを作成した後、mesh を変更したくなった場合に使用する。

- charge_makes_cube_dir.py
  ⇒ real,imag の cube ファイルがあるときに char の cube ファイルを作成するツール。

- readme.txt ⇒ Wave Tool の使用方法が簡単に書いてある。

- WaveToolManual.pdf(本マニュアル)
  ⇒ Wave Tool の仕様などが書いてある。

1.5 WaveTool の実行

実行コマンドは以下の通りである。
注）json ファイルを分割している場合 (-s オプションを使用している場合) は親玉の json ファイル (数字が書かれていないもの) を入力する

```
usage: Main_AutoWave.py [-h] [-s STRIDE] [-load-min LOAD-MIN]
                        [-load-max LOAD-MAX] [-cutoff-au CUTOFF]
                        [-input_mesh_grid PATH] [-alpha ALPHA] [-mesh MESH]
                        [-target target] [-core CORE] [-periodic periodic]
                        [--parallel] [--position] [--LCAO] [--big-endian]
                        elses-generate-cubefile JSON XYZ BASIS
Main_AutoWave.py: error: too few arguments
```

エラー無く、実行できると、「***.json.cube」というフォルダが出来上がり、中に、real、imag、char というフォルダが出来ており、中に Cube ファイルが作成されている。

オプションの説明

- -h
  help

- -s STRIDE
  何個飛ばしで読み込むか (wavepacket 内部のステップを割り切れるかどうかで判定している) (default:1)

- -load-min LOAD-MIN
  読み込むステップの最小値を決めている (default:0)

- -load-max LOAD-MAX
  読み込むステップの最大値を決めている (default:10000000)

- -cutoff-au CUTOFF
  elses-generate-cubefile で使用する cutoff の値 (default:8.0)

- -input_mesh_grid PATH
  別階層にある input_mesh_grid を現在のディレクトリに移動する (すでにある場合は何もしない)(自動で input_mesh_grid を作製するオプションと併用した場合はこのオプションは動作しない)

- -alpha ALPHA
  最初のステップの mean の値と最後のステップの値の合計の MSD を使用して $mean \pm alpha\sqrt{MSD}$ の範囲 (正方形) の input_mesh_grid を作製 (既にある input_mesh_grid を上書してしまうので注意)
  (--position と -alpha を同時に使用すると最大で position で作製される範囲になるように alpha で作製される)

- -mesh MESH
  オプションで input_mesh_grid を作製する際のメッシュ数を変更する (一辺のメッシュ数は大体 長さ [a.u.]*MESH で作製される) default : 1.0

- -target target

作りたい wavepacket 内部のステップを指定する (-load-min,max は無視される)(複数ステップを入力する場合は「115,223,587」のように「,」区切りでスペースがないように入力)

- -core CORE
  --parallel を使用したときの並列数を決めている (default では OMP_NUM_THREADS の値になる。OMP_NUM_THREADS がない場合は最大の並列数になる)

- --parallel
  読み込みと char の cube ファイル作製を python で並列化させる (メモリが十分にあるときだけ使用すること)

- --position
  xyz ファイルを読み込みその原子が全て含まれるように自動で input_mesh_grid を作製 (既にある input_mesh_grid を上書してしまうので注意)

- --big-endian
  oakleaf,K での bynari 出力を読み込む時に使用する

- --LCAO
  LCAO 係数データファイル (output_wavefunction) のみを出力するモード (mpi 並列で実行しても意味ありません)

- -periodic periodic
  周期境界条件を適用するときに使用する。入力した xyz ファイルに cell の情報が書かれていない場合は動きません。入力方法は -periodic zxy のように拡張したい方向を入力する (順不同)。周期セルに合わせるように input_mesh_grid.txt を作成する。ただし、他の自動作成のオプションを使用した場合はそちらが優先される。

  注) 現在 (2017/03/17)、このオプションを使用しても余り意味がありません (周期境界条件ぴったりの input_mesh_grid.txt を作成するのみ)。周期境界条件を変更したい場合は、「output_basis_infomation.txt」の中を直接編集してください。
  3 から 5 行目の右端の数字が周期境界条件を適用するかどうか現しているので、この値を適用する場合は「1」、適用しない場合は「0」にしてください。

```
# file_format= v0.04.05
       209       518
     40.95296659       0.00000000       0.00000000    0
      0.00000000      31.59932121       0.00000000    0
      0.00000000       0.00000000      29.04442559    0
```

  xyz ファイルに cell の情報を追加していない時に周期境界条件を適用する場合は、「input_mesh_grid.txt」を自動で作成せずに以下のように手動で作成してください。

```
80 80 80 # Number of mesh points
0.0d0 0.0d0 0.0d0 internal # origin of the mesh grid region
1.0d0 1.0d0 1.0d0 internal # sizes of the mesh grid region
```

1.5.1 具体例

初めの 100 ステップ分の Cube ファイルを作成する場合

```
$ python Main_AutoWave.py -load-min 0 -load-max 100 \
elses-generate-cubefile out.json position.xyz output_basis_information.txt
```

5 ステップ刻みで Cube ファイルを作成する場合

```
$ python Main_AutoWave.py -s 5 elses-generate-cubefile \
out.json position.xyz output_basis_information.txt
```

0,10,100 ステップ目の Cube ファイルを作成する場合

```
$ python Main_AutoWave.py -target 0,10,100 elses-generate-cubefile \
out.json position.xyz output_basis_information.txt
```

1.6 mpi4py で並列実行する場合

実行方法は、mpi4py をインストールしている状態で以下のように「python Main_AutoWave.py」の前に「mpirun -n 4」などと普通の mpi プログラムのように並列数指定して実行する。

実行コマンド

```
$ mpirun -n 4 python Main_AutoWave.py
usage: Main_AutoWave.py [-h] [-s STRIDE] [-load-min LOAD-MIN]
                        [-load-max LOAD-MAX] [-cutoff-au CUTOFF]
                        [-input_mesh_grid PATH] [-alpha ALPHA] [-mesh MESH]
                        [-target target] [-core CORE] [--parallel]
                        [--position] [--big-endian]
                        elses-generate-cubefile JSON XYZ BASIS
Main_AutoWave.py: error: too few arguments
```

オプションについては普通に実行する場合と同様

1.6.1 具体例

初めの 100 ステップ分の Cube ファイルを作成する場合

```
$ mpirun -n 4 python Main_AutoWave.py -load-min 0 -load-max 100 \
elses-generate-cubefile out.json position.xyz output_basis_information.txt
```

5 ステップ刻みで Cube ファイルを作成する場合

```
$ mpirun -n 4 python Main_AutoWave.py -s 5 elses-generate-cubefile \
out.json position.xyz output_basis_information.txt
```

1.7　並列性能の結果

mpi4py を使用した WaveTool の並列化を行った部分の並列性能についてベンチマークを取った。使用した環境は、京都大学のスパコン (laurel) と東京大学のスパコン (oakleaf FX10)。結果として、ノード数を 2,4,8,…倍と上げていくと計算時間が短縮されるのを確認できた。

表 1　laurel での WaveTool の並列化を行った部分の実行時間データ

ノード数 N	実行時間 T(s)
1	1579.838029
2	800.036737
4	399.55529
8	200.265127
16	104.539619

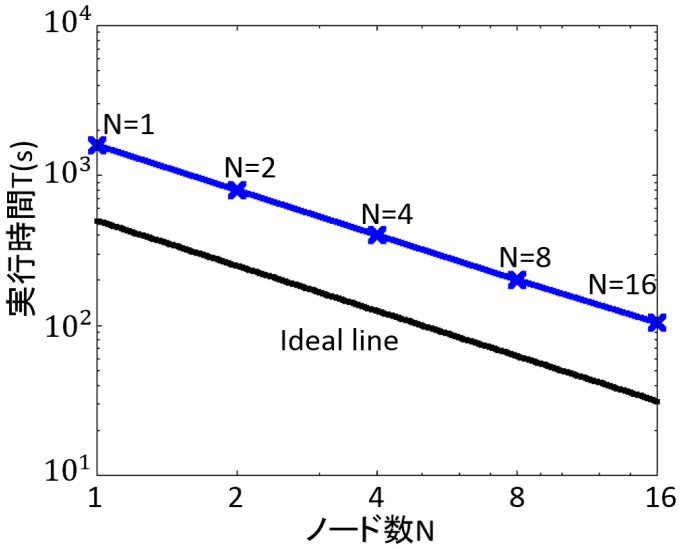

図 3　laurel での WaveTool の並列化を行った部分の並列効率。

表 2　Oakleaf FX10 での WaveTool の並列化を行った部分の実行時間データ

ノード数 N	実行時間 T(s)
1	2670.982229
2	1334.658283
4	668.098373
8	333.59436
16	174.566309

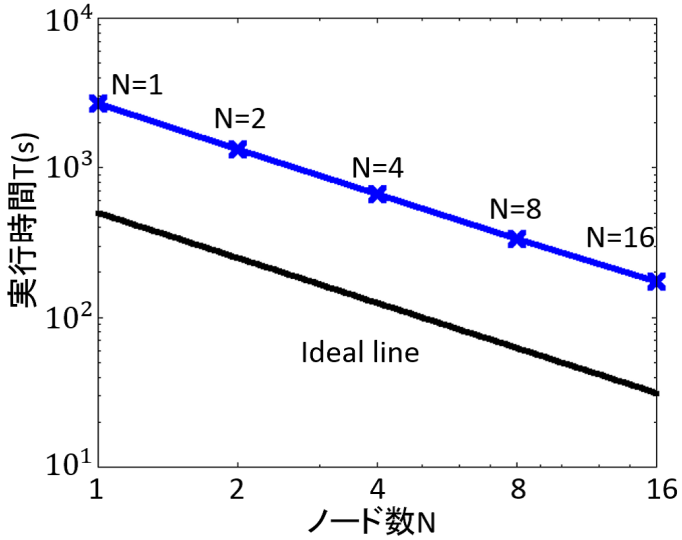

図 4　Oakleaf FX10 での WaveTool の並列化を行った部分の並列効率。

1.8　残りのツールについて

1.8.1　make_input_mesh_grid_20151013.py

input_mesh_grid を自動生成するツール。

実行コマンド

```
>python make_input_mesh_grid_20151013.py
usage: make_input_mesh_grid_20151013.py [-h] [-group GROUP]
                                        [-group-min GROUP-MIN]
                                        [-group-max GROUP-MAX]
                                        [-cutoff-au CUTOFF] [-mesh MESH]
                                        XYZ
make_input_mesh_grid_20151013.py: error: too few arguments
```

オプションの説明

- -group GROUP
  group id を読み込み、原子に group id を割り振る。(default は、一原子一グループで xyz ファイルに書かれている順番でグループが割り当てられる。)
- -group-min GROUP-MIN
  作成する範囲に入れるグループの最低値を決めている。default：1
- -group-max GROUP-MAX
  作成する範囲に入れるグループの最大値を決めている。default：10000000
- -cutoff-au CUTOFF
  一番端の原子からどのくらいの距離を離すか決めている。単位は [a.u.]。default：8.0
- -mesh MESH
  メッシュ数を変更する (一辺のメッシュ数は大体 長さ [a.u.]*MESH で作製される) default：1.0

具体例

45 から 55 番目のグループの場所に input_mesh_grid.txt を合わせたい場合

```
$ python make_input_mesh_grid_20151013.py -group group_id.txt \
-group-min 45 -group-max 55 position.xyz
```

1.8.2 Main_AutoWave_not_LCAO_20151022.py

LCAO 係数ファイルを作成せずに cube ファイルを作成するツール。1 度 cube ファイルを作成した後、mesh を変更したくなった場合に使用する。

```
>python Main_AutoWave_not_LCAO_20151022.py
usage: Main_AutoWave_not_LCAO_20151022.py [-h] [-s STRIDE] [-cutoff-au CUTOFF]
                                          [-input_mesh_grid PATH]
                                          [-position PATH] [-mesh MESH]
                                          [--parallel]
                                          elses-generate-cubefile basis_dir
Main_AutoWave_not_LCAO_20151022.py: error: too few arguments
```

basis_dir には、「*.json.cube/」を入力する。

オプションの説明

- -position PATH
  PATH に xyz ファイルを入力するとその xyz ファイルの原子がすべて入るように自動で input_mesh_grig.txt を作成する。

- 残りのオプションは Main_AutoWave.py と同じ。

具体例

mesh の位置を作り直す場合

```
$ python make_input_mesh_grid_20151013.py -group group_id.txt \
-group-min 45 -group-max 55 position.xyz
$ python Main_AutoWave_not_LCAO_20151022.py \
elses-generate-cubefile out.json.cube/
```

mesh の数を増やす場合

```
$ python Main_AutoWave_not_LCAO_20151022.py -position position.xyz \
-mesh 2.0 elses-generate-cubefile out.json.cube/
```

1.8.3 charge_makes_cube_dir.py

real,imag の cube ファイルがあるときに char の cube ファイルを作成するツール。

<dir> に*.json.cube を指定すると*.json.cube の中に char というフォルダを作成して*.json.cube の中にある

real,imag のフォルダの中にあるファイルから char を作成する。

```
>python charge_makes_cube_dir.py
Usage: python charge_makes_cube.py <dir> [parallel]
```

第 3 引数に「parallel」を入力すると並列化されます。

具体例

```
$ python charge_makes_cube.py out.json.cube parallel
```

2　CUBE → png へのバッチ処理によるグリッド可視化

この章では、VisBAR Wave Batch を使用したグリッドデータの可視化について簡単に説明する。
詳しい仕様などは、VisBAR Wave Batch のマニュアルを参照してください。

2.1　VisBAR Wave Batch の入手方法

VisBAR Wave Batch は以下のサイトで公開されている。
`https://github.com/visbar/visbar_wb`

2.2　カメラ情報の作成

以下のコマンドで可視化したい初期ステップと最終ステップのグリッドデータを作成する。

```
$ python Main_AutoWave.py -target 0,999 --position --parallel \
elses-generate-cubefile out.json position.xyz output_basis_information.txt
```

作成したグリッドデータを GUI 操作可能な環境 (ディスプレイが付いている環境) に持っていく。(charge のグリッドデータを持っていくのが望ましい。real や imaginary は、時間によって振動しているため)
以下のコマンドでグリッドデータを可視化し、適当な等値面、カメラアングルにしたら c を押して、カメラ情報を保存する。

```
$ python VisBAR_wave_batch.py ./Input -o ./Output
```

2.3　batch 処理

上記手順で作成したカメラ情報を移し、以下のコマンドで batch 処理を行う。
(カメラ情報を移す際、ファイル名を「visbar wb setting output.txt」から「visbar wb setting.txt」に変更すること)

```
$ python Main_AutoWave.py -s 10 --parallel \
elses-generate-cubefile out.json position.xyz output_basis_information.txt
$ python VisBAR_wave_batch.py -b -parallel out.json.cube/real/
$ python VisBAR_wave_batch.py -b -parallel out.json.cube/imag/
$ python VisBAR_wave_batch.py -b -parallel out.json.cube/char/
```

batch 処理が終わると以下のフォルダ内に連番 png ファイルが作成される。

```
out.json.cube/real/
out.json.cube/imag/
out.json.cube/char/
```

作成した連番 png ファイルをつなぎ合わせるとアニメーションが作成できる。

付録 A　argparse のインストール (Python v.2.6 以前を利用する場合)

WaveTool は python2.7 から標準で搭載された argparse を使用している。
python2.6 以前のバージョンで WaveTool 使用する場合は argparse のインストールする必要がある。
以下は argparse のインストール方法である。

`https://pypi.python.org/pypi/argparse` から「argparse-1.4.0.tar.gz」をダウンロードする。
「argparse-1.4.0.tar.gz」のあるフォルダに移動後以下のコマンドを実行する。

```
$ tar zxvf argparse-1.4.0.tar.gz
$ cd argparse-1.4.0
$ python setup.py build
$ python setup.py install --user
```

付録 B　mpi4py のインストール

WaveTool は mpi4py を使用することで分散型の並列処理を行うことができる。(1 つ 1 つ処理時間が短くなるわけではない)
以下は mpi4py のインストール方法である。

`https://bitbucket.org/mpi4py/mpi4py/downloads` から mpi4py-1.3.1.tar.gz をダウンロードする。

```
$ tar zxvf mpi4py-1.3.1.tar.gz
$ cd mpi4py-1.3.1
$ python setup.py build
$ python setup.py install --user
```

付録 C　更新履歴

- 初稿 2016/09/08
- 引数に basis infomation を追加 2016/09/08
- 「CUBE → png へのバッチ処理によるグリッド可視化」を section に変更し、可視化方法の例を掲載。
  付録に argparse と mpi4py のインストール方法を追加。
  「WaveTool に必要な環境・入力ファイル」を追加
  2017/03/10
- オプション「-periodic」の説明を追加 2017/03/17