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

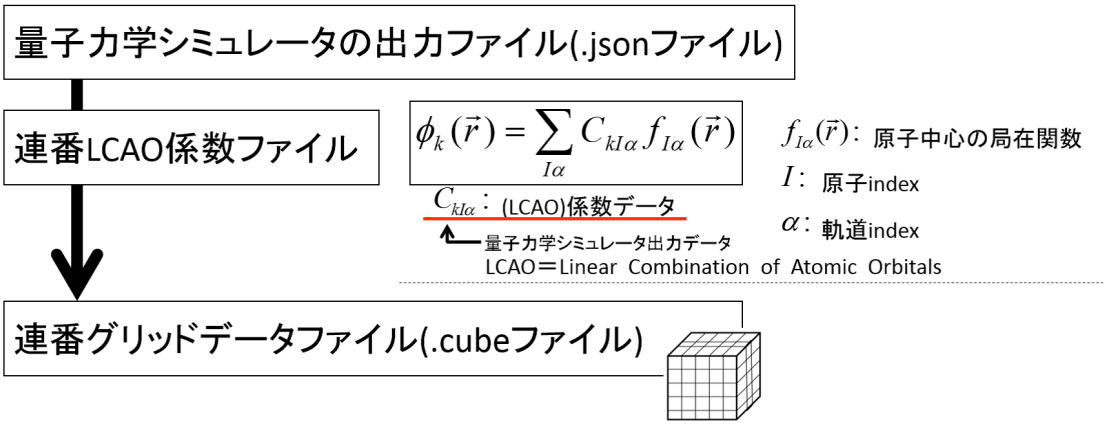

図 1　Wave Tool のワークフロー

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。

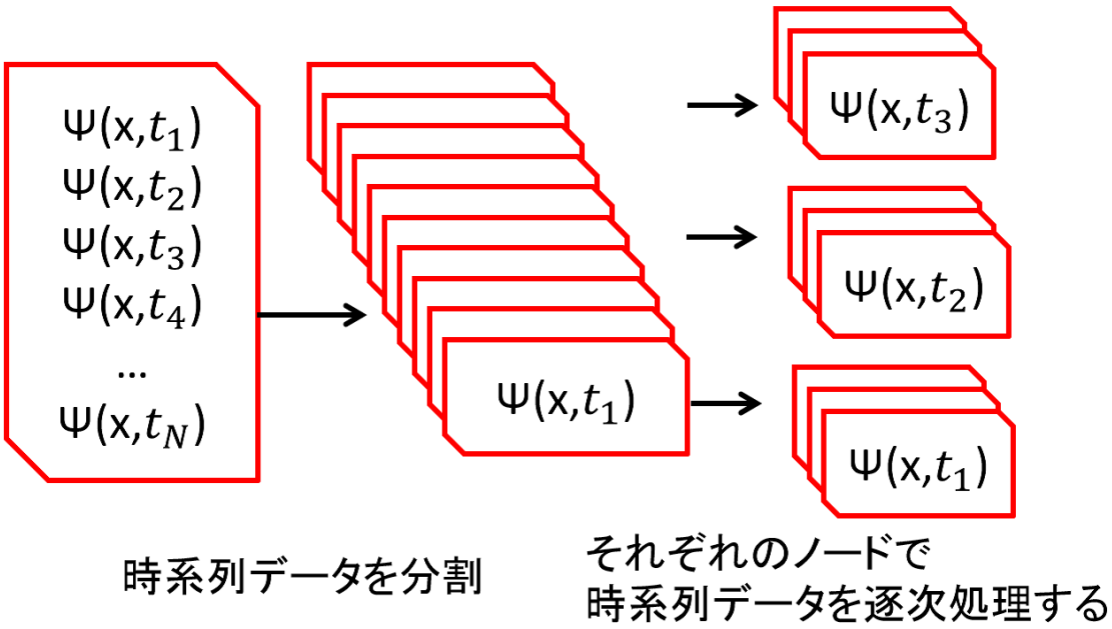

図 2　Wave Tool の並列イメージ

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 の構成

WaveTool_20170119.zip の展開

```
$ unzip WaveTool_20170119.zip
```

展開すると、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。

1.4 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)

1.4.1 **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
```

1.5 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

1.5.1 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

```
$ 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
```

1.6 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

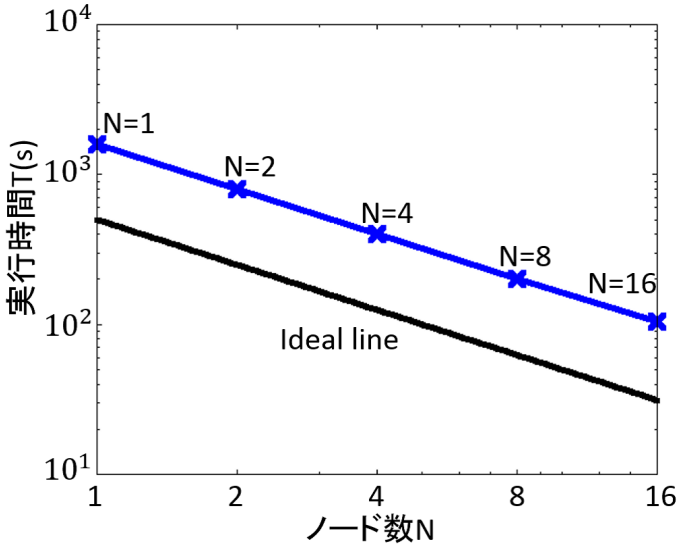

図 3　laurel での WaveTool の並列化を行った部分の並列効率。

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

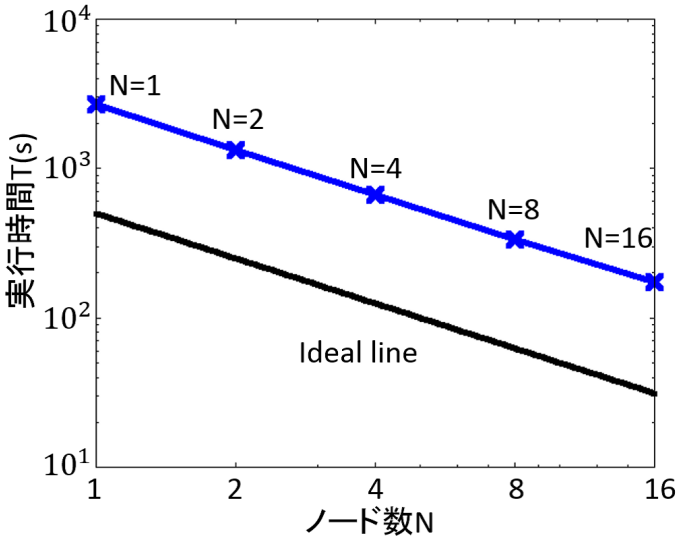

図 4　Oakleaf FX10 での WaveTool の並列化を行った部分の並列効率。

1.7 残りのツールについて

1.7.1 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
```

1.7.2 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/
```

1.7.3 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
```

1.7.4 CUBE → png へのバッチ処理によるグリッド可視化

VisBAR WB のマニュアルを参照

1.7.5 連番 png → gif 動画の生成

VisBAR IT のマニュアルを参照

付録 A　更新履歴

- 初稿 2016/09/08
- 引数に basis infomation を追加 2016/09/08