

# PHASE/0 と ASE を連携して利用する

## 1. 概要

PHASE/0 は, Atomic Simulation Environment (ASE)

(Ask Hjorth Larsen, Jens Jørgen Mortensen, Jakob Blomqvist et al. “The atomic simulation environment—a Python library for working with atoms” J. Phys.: Condens. Matter 29 (2017) 273002, <https://wiki.fysik.dtu.dk/ase/>)と連携して利用することが可能です。本稿ではそのための手続きを説明します。なお、対応するのは ASE バージョン 3.16 以降です。

## 2. ASE のインストール方法

ASE は, Python 2.7 以上もしくは 3.4 以上の環境で動作します。

そこで, まずは Python 2.7 以上もしくは 3.4 以上の環境を準備します。また, NumPy (<https://docs.scipy.org/doc/numpy/>)は必須です。さらに, 必須ではありませんがいくつかの機能は SciPy (<https://www.scipy.org/>), Matplotlib (<https://matplotlib.org/>)に依存します。これらは, 事前にインストールしておくことが望ましいです。インストールは, pip (<https://pip.pypa.io/en/stable/>)を利用すると簡単に行えます。pip は, Python 2.7.9 以上もしくは 3.4 以上には標準インストールされています。もし Python 2.7.9 未満のバージョンを利用している場合, pip のインストールから始めることが推奨されます。pip は, Python 2.7 以上を利用しているのであれば `get_pip.py` スクリプト (<https://bootstrap.pypa.io/get-pip.py>)を適当なディレクトリーから実行すればインストールできます。

pip がインストールされている環境では, NumPy, SciPy, Matplotlib は以下の要領でインストールすることができます。

```
$ pip install --upgrade --user numpy scipy matplotlib
```

上記の要領で NumPy, SciPy, Matplotlib のインストールができれば, 以下のコマンドによって ASE をインストールします。

```
pip install --upgrade --user ase
```

上述のコマンドでインストールの場合,

`$HOME/.local/lib/pythonX.Y/site-packages/ase` 以下に ASE がインストールされます。必須ではありませんが, `$HOME/.local/bin` を環境変数 `PATH` に加えておくと

便利に利用できます。

PHASE/0 を ASE と連携して利用するには、`phase0_ase/ase` 以下にあるファイルを対応する位置にコピーします。もしくは、`ase` フォルダを丸ごと上書きコピーしてください。ただし、`formats.py` は上書きコピーになるので、バックアップを取っておくことを推奨します。

- `ase/io/formats.py`
- `ase/io/phase0.py`
- `ase/calculators/phase0.py`

### 3. 利用方法

#### 3.1 準備

ASE から PHASE/0 を利用する場合、環境変数を設定する必要があります（座標データをコンバートするだけならば不要）。以下の環境変数を設定します。

ASE_PHASE0_COMMAND	PHASE/0 を実行するコマンドを指定します。たとえば <code>export ASE_PHASE0_COMMAND='mpirunp -n 2 phase'</code> など
PHASE_PP_PATH	PHASE/0 で読み込める擬ポテンシャルデータが置かれているディレクトリを指定します。たとえば <code>export PHASE_PP_PATH=\$HOME/pp</code> など。

#### 3.2 ase.calculators.phase0.phase0 の利用方法

PHASE/0 を ASE のエンジンとして利用する場合、`ase.calculators.phase0.phase0` クラスを利用します。たとえば、以下の要領でインスタンス化します。

```
from ase.calculators.phase0 import phase0
phase = phase0(ecut=25, stress=True)
```

コンストラクターに渡す引数によって PHASE/0 の計算条件をカスタマイズすることができます。引数は、キーワード = 値 という形式で指定します。すべてのキーワードにデフォルト値が設定されているので、`phase0()` としてインスタンス化しても計算を実行することはできます。

`phase0` クラスのコンストラクターに渡せるキーワードとその説明は、以下の通り。

キーワード	説明
ecut	カットオフエネルギー。単位は Rydberg. デフォルト値は 25 Rydberg.
symmetry	対称性を考慮するかどうかを真偽値(True もしくは False)で指定する。デフォルト値は True.
inversion	反転対称性がある場合にそれを積極的に活用するかどうかを指定する。デフォルト値は False.
stress	ストレステンソルを計算するかどうかを真偽値で指定する。デフォルト値は False.
stress_correction	stress=True の場合に、ストレステンソル計算の補正を行うかどうかを指定する真偽値。デフォルト値は True.
kpts	$\mathbf{k}$ 点サンプリングを指定する。カンマ区切りで整数を 3 つ指定するか、実数を 1 つ指定する。カンマ区切りで整数を 3 つ指定する場合、 $a, b, c$ 軸の分割数と解釈される。実数を 1 つ指定する場合、 $\mathbf{k}$ 点の逆空間における“密度”と解釈される。デフォルト値は 3.5.
gamma	$\mathbf{k}$ 点サンプリングとして $\Gamma$ 点のみを使用するかどうかを指定する真偽値。True の場合、kpts の設定は無視される。デフォルト値は False.
mobile	原子の mobile 属性を指定する。デフォルト値は False.
spin	スピンを考慮した計算を行うかどうかを指定する真偽値。デフォルト値は False.
zeta	スピンを考慮する計算の場合に、元素ごとの初期スピン分極を指定する Python dictionary. zeta={elem1:zeta1,elem2:zeta2,..}のように指定する。この例では、elem1 という元素の初期スピン分極は zeta1 に、elem2 という元素の初期スピン分極は zeta2 になる。指定のない元素はスピン分極 0 が採用される。
preproc	PHASE/0 を preparation モードで実行するか

	どうかを指定する真偽値。デフォルト値は False.
dE	SCF 計算の収束判定条件を指定する実数。デフォルト値は 1.0e-9.
pp	<p>利用する擬ポテンシャルファイルを指定する Python dictionary.</p> <p>pp={elem1:pp1,elem2:pp2...}のように指定する。この例では、elem1 という元素に pp1 という擬ポテンシャルファイルが、elem2 という元素に pp2 という擬ポテンシャルファイルが割り当てられる。指定がない場合、PHASE_PP_PATH で指定されるディレクトリの下から検索し、得られた擬ポテンシャルファイルが採用される。</p>

### 3.3 ase.io モジュールの利用方法

ase.io モジュールは、外部ファイルの原子配置座標データを ASE の原子配置データオブジェクトに割り付けるための関数が定義されたモジュールです。read および write という関数を利用することができます。次のように利用することができます。

```
import ase.io.read
import ase.io.write

atoms = ase.io.read(filename,index=None,format=None, **kwargs)
ase.io.write(filename,atoms,format=None,**kwargs)
```

**\*\*kwargs** は、原子配置データファイルの種類に応じたオプションです。利用できるオプションは以下の通り。

#### read 関数

オプション	説明
filename	座標データファイル名を指定する。 nfnfp.data ファイルというファイル名の場合 PHASE/0 の入力ファイル、nfdynm.data というファイル名の場合 PHASE/0 の出力ファイルとみなされる。このようなファイル名以

	外でも、後述の <code>format</code> オプションによってファイルの種類を指定することができる。
<code>format</code>	PHASE 入力ファイルの場合 <code>phase0</code> , 出力ファイル ( <code>nfdynm.data</code> ファイル) の場合 <code>phase0-out</code> を指定します。入力ファイルの場合 <code>nfinp.data</code> , 出力ファイルの場合 <code>nfdynm.data</code> というファイル名のファイルを利用する場合 <code>format</code> 指定は不要です。
<code>index</code>	<p>フレームデータの場合に、どのフレームを読み込むかを整数値で指定する。負の値の場合は最後のフレームを読み込む。もしくは、Python の <code>slice</code> を指定することによって複数のフレームを取り込むことも可能であり、この場合の返り値は <code>Atoms</code> のリストとなる。Python の <code>slice</code> は、以下のように作成することができる。</p> <p><code>slice(start,end,step)</code></p> <p><code>start</code> に始インデックスを 0 始まりで指定する。<code>end</code> に終インデックスを指定する。<code>step</code> に、スキップするフレーム数を指定する。<code>end</code> は、たとえば-1 を指定すると最終フレームを指定することができる。詳しくは Python の <code>slice</code> のドキュメントを参照されたい。</p>
<code>wrap</code>	周期境界条件を考慮して単位胞に収まらない原子を収めるかどうかを指定する真偽値。デフォルト値は <code>False</code> 。
<code>fname_from_fnamesdata</code>	ファイル名を <code>file_names.data</code> ファイルから読み込むかどうかを指定する真偽値。 <code>True</code> の場合、 <code>filename</code> の指定は無視され、 <code>file_names.data</code> ファイルから読み込まれるファイル名が採用される。デフォルト値は <code>False</code> 。

#### write 関数

オプション	説明
<code>filename</code>	座標データファイル名を指定する。PHASE/0

	に関しては、書き出しに対応しているのは入力パラメーターファイルのみである。
format	PHASE/0 入力の場合 phase0 を指定します。ファイル名が nfinp.data の場合個の指定は不要です。なお、write 関数は nfdynm.data ファイルの出力には対応していません。
atoms	原子配置を表す Atoms オブジェクトのインスタンス
cartesian	カルテシアン座標で書き出すかどうかを指定する真偽値。デフォルト値は False.
angstrom	Å 単位で書き出すかどうかを指定する真偽値。デフォルト値は False (Bohr 単位で書き出される).
mobile	“mobile”属性値を設定する真偽値。すべての原子について同じ値を設定することしかできない。デフォルト値は False.

### 3.4 ASE 利用例

ASE を利用した Python スクリプトの例を紹介します。

例 1. CIF から結晶構造を読み込み、構造最適化を施す

```

1  from ase.calculators.phase0 import phase0
2  from ase.optimize import QuasiNewton
3  from ase import io
4
5  atoms = io.read('coordinate.cif')
6  #何か処理を行う。。。
7  #...
8  #...
9
10 #phase0 オブジェクトを作成
11 phase = phase0(ecut=36, kpts=3.0, spin=True, zeta={'Fe':0.2})
12 #phase0 オブジェクトを atoms オブジェクトに設定
13 atoms.set_calculator(phase)
14 #QuasiNewton ソルバーのオブジェクトを作成, 実行

```

```

15     qn = QuasiNewton(atoms, trajectory='ase.traj')
16     qn.run(fmax=0.01)
17     #結果解析
18     result = io.read('ase.traj')
19     result.get_potential_energy()
20     ...
21     ...

```

まず1行目から3行目で必要なモジュールやクラスを読み込んでいます。1行目で読み込んでいるのは `phase0` クラスであり、PHASE/0 を利用して計算する場合に必要となります。2行目では `QuasiNewton` クラスを読み込んでいます。このクラスは、ASE に組み込まれている準ニュートンソルバーを使った構造最適化を行うことができるクラスです。3行目で読み込んでいるのは `read` 関数であり、座標データファイルから ASE の原子配置を表すオブジェクトを作成することができる関数です。

5行目からが実際の処理です。まず5行目で `coordinate.cif` というファイルから原子配置データを読み込んでいます。このケースでは拡張子が `'cif'` であり CIF であることが明らかなため、ファイルの種類を指定する必要はありませんが `format` オプションによって明示的に指定することもできます。`atoms` オブジェクトに必要な応じて処理を施し、11行目では `phase0` オブジェクトを作成しています。`phase0` オブジェクトは引数で計算条件をいろいろ設定することが可能です (PHASE/0 のすべての機能が利用できるわけではありません)。13行目では、`atoms` オブジェクトに `phase0 calculator` のオブジェクトを設定しています。15行目では `QuasiNewton` ソルバーオブジェクトを `atoms` を引数に取りながら作成しています。このように記述することによって、`atoms` に記録されている原子配置データを PHASE/0 を使って最適化することができます。16行目で `QuasiNewton` クラスの `run` メソッドを使って最適化を走らせています。最後に、18,19行目で結果ファイルを読み込み、必要に応じて解析作業を行っています。

例 2. `nfdynm.data` から結晶構造を読み込み、PHASE/0 入力を作成する

```

1     from ase.calculators.phase0 import phase0
2     from ase.io import read
3
4     atoms = read('nfdynm.data')
5     #何か処理を行う。。。
6     #...

```

```

7      #...
8
9      #nfinp.data および file_names.data ファイルを出力
10     phase = phase0()
11     phase.export_input(atoms)

```

1 行目と 2 行目で必要なモジュールやクラスを読み込んでいます。1 行目で phase0 の入力  
を出力するためのクラス, 2 行目で座標データから ASE の原子配置オブジェクトを作成す  
る関数を読み込んでいます。

4 行目の処理で, 原子配置データを nfdynm.data ファイルから読み込んでいます。  
nfdynm.data ファイルは, ファイル名によって PHASE/0 の出力フォーマットであることが  
判定できるので, 座標データの種類に関する指定は特にしていません。読み込んだあと必要  
に応じて処理を施した後 10 行目で phase0 オブジェクトを作成し, 11 行目で PHASE/0 で  
そのまま利用できる形式の入力パラメーターファイルを出力しています。

### 3.5 phasew.py の利用方法

本パッケージには ASE を利用した phasew.py というスクリプトが付属しています。このス  
クリプトは PHASE/0 のラッパーとなっており, たとえば CIF から手軽に構造最適化や状  
態方程式の計算が行いたい, という場合に有用です。phase0\_ase/bin の下にあります。た  
えば, 以下のようなコマンドによって構造最適化を行うことができます。

```

export      ASE_PHASE0_COMMAND='mpirun      -np      4
$HOME/phase0_2018.01/bin/phase ne=2 nk=2 '
export PHASE_PP_PATH=$HOME/pp
$HOME/phase0_ase/bin/phasew.py -f foo.cif -e 16 -k 4,8,1

```

phasew.py スクリプトは, -h オプションをつけて実行すると以下のようなヘルプメッセー  
ジを表示させることができます。

```

$ phasew.py -h
Usage: phasew.py [options]

PHASE/0 wrapper script. environment variable ASE_PHASE0_COMMAND and
PHASE_PP_PATH must be set prior to the execution of this script.
ASE_PHASE0_COMMAND specifies the command used to boot PHASE/0. example :

```



'mpirun -np 8 phase ne=2 nk=4'. PHASE\_PP\_PATH specifies the directory under which the PHASE/0 pseudopotential files reside. example : \$HOME/pp

Options:

-h, --help show this help message and exit

-f COORD\_FILE, --file=COORD\_FILE  
the file name for the coordinate file. Required.

-i INDEX, --index=INDEX  
the index of the coordinate if the specified file supports frame. defaults to -1, i.e., the last frame

-t COORD\_TYPE, --type=COORD\_TYPE  
type of the coordinate file. defaults to None, in which case the ase resolver resolves the file type

-e ECUT, --ecut=ECUT cutoff energy for the WFs in Rydberg units. the default value is 25

-k KPTS, --kpts=KPTS specify the kpoint sampling. specify either one float value or three comma-seperatred integers.  
the former specifies the density of the kpoints in  $\text{\AA}^{-1}$  units. the latter corresponds to the mesh for the a,b and c axis. the default value is 3.5

-x FMAX, --fmax=FMAX convergence criterion (max. force) in eV/\AA units. the default value is 0.01

-s, --spin enable this option in order to perform spin-polarized calculations.

--nosymm enable this option in order to disable symmetry.

--inversion enable this option in order to set the sw\_inversion parameter to 'on'.

-z ZETA, --zeta=ZETA specify the initial per-element spin polarization. use the following format :  
--zeta=elem1:zeta1,elem2:zeta2 ...

-p PP, --pp=PP specify pseudopotential files in the following format : --pp=elem1:pp1,elem2:pp2,...

-c, --clean

-m MODE, --mode=MODE specify the operation mode. one of : optimize, eos, preproc, or input\_only. optimize : structural

```
optimization by the ase quasi newton solver (default)

eos      : equation of state calculation preproc :

run PHASE/0 in preparation mode input_only : create

input and exit

-d DELTA, --delta=DELTA

the eps value for the calculate_eos method. does not

make sense when --mode is not eos

-n NPOINTS, --npoints=NPOINTS

the number of points for the calculate_eos method.

does not make sense when --mode is not eos
```

利用可能なオプションは、下記の通り。

オプション	説明
-h, --help	ヘルプメッセージを表示し、終了する。
-f COORD_FILE, --file=COORD_FILE	計算対象としたい座標データファイルを指定する。必須オプション。
-i INDEX, --index=INDEX	フレームデータの場合、対象としたいフレーム番号を 0 始まりの整数で指定する。-1 を指定すると、最後のフレームが選択される。デフォルト値は-1
-t COORD_TYPE, --type=COORD_TYPE	座標データの種別を指定する。座標データの種別がファイル名から判定できる場合 None を指定すれば ASE が判定する。指定がある場合、その指定が優先される。デフォルト値は None.
-e ECUT, --ecut=ECUT	カットオフエネルギーを Rydberg 単位で指定する。デフォルト値は 25.
-k KPTS, --kpts=KPTS	<b>k</b> 点サンプリングを指定する。カンマ区切りで整数を 3 つ指定するか、実数を 1 つ指定する。カンマ区切りで整数を 3 つ指定する場合、 <i>a, b, c</i> 軸の分割数と解釈される。実数を 1 つ指定する場合、 <b>k</b> 点の逆空間における“密度”と解釈される。この場合の単位は 1/Å.デフォルト値は 3.5.
-x FMAX, --fmax=FMAX	構造最適化の収束判定条件を指定する。収束判定条件は原子間力の最大値であり、単位は

	eV/Å. デフォルト値は 0.01
-s, --spin	スピンを考慮した計算を行う場合このオプションを有効にする。
--nosymm	対称性を無効にしたい場合このオプションを有効にする。
--inversion	系に反転対称性があり，それを活用して計算負荷を減らしたい場合にこのオプションを有効にする。
-z ZETA, --zeta=ZETA	スピンを考慮する計算の場合に，元素ごとの初期スピン分極を指定する。--zeta=elem1:zeta1,elem2:zeta2...のように指定する。この例では，elem1 という元素の初期スピン分極は zeta1 に，elem2 という元素の初期スピン分極は zeta2 になる。指定のない元素はスピン分極 0 が採用される。-s もしくは--spin をつけている場合に意味のある設定。
-p PP, --pp=PP	利用する擬ポテンシャルファイルを指定する。--pp=elem1:pp1,elem2:pp2...のように指定する。この例では，elem1 という元素に pp1 という擬ポテンシャルファイルが，elem2 という元素に pp2 という擬ポテンシャルファイルが割り当てられる。指定がない場合，PHASE_PP_PATH で指定されるディレクトリーの下から検索し，得られた擬ポテンシャルファイルが採用される。
-m MODE, --mode=MODE	このスクリプトの動作モードを指定する。 optimize, eos, preproc, input_only のいずれか。 optimize：ASE 標準の準ニュートン法によって構造最適化を施す。 eos：状態方程式計算を行う。 preproc：PHASE/0 の”preparation モード”を走らせる input_only：PHASE/0 の入力ファイルを作成し，終了する。
-d DELTA, --delta=DELTA	--mode=eos の場合に，体積の上限と下限を決

	めるファクターを指定する。座標データファイルの体積が $V_0$ だったとすると、 $V_0 \cdot (1 - \text{delta})$ から $V_0 \cdot (1 + \text{delta})$ の体積を考慮するように動作する。デフォルト値は 0.1
<code>-n NPOINTS, --npoints=NPOINTS</code>	<code>--mode=eos</code> の場合に、体積を何点考慮するかを指定する。デフォルト値は 10.

計算結果は、拡張子 traj ファイルに格納されます（ファイル名の接頭辞は指定した座標データファイルの接頭辞となります）。traj ファイルは座標やエネルギー、原子間力を格納することができる ASE の内部フォーマットです。その参照の仕方などについては ASE のウェブサイトなどを参照してください。手軽に解析をする方法としては、ASE の GUI を利用する方法があります。`$HOME/.local/bin` にパスが通っていると仮定すると、以下の要領で ASE の gui を起動することができます。

```
$ ase gui foo.traj
```

この GUI から、原子配置の履歴や原子間力・エネルギーのグラフ表示などを行うことが可能です。

### 3.6 initial\_replica\_builder.py の利用方法

initial\_replica\_builder.py は、NEB 計算で利用できる初期レプリカ列を作成することができるスクリプトです。線形補完によるレプリカ列作成は PHASE/0 の標準の機能で実現できますが、initial\_replica\_builder.py を利用すると“IDPP 法” (Smidstrup S, Pedersen A, Stokbro K and Jónsson H “Improved initial guess for minimum energy path calculations” J. Chem. Phys. 140 214106) による初期レプリカ列の作成を行うことができます。このスクリプトも、`phase0_ase/bin` の下にあります。

initial\_replica\_builder.py を `-h` オプションをつけて実行すると以下のようなヘルプメッセージを表示させることができます。

```
$ initial_replica_builder.py -h
Usage: initial_replica_builder.py [options]

create initial replicas for the PHASE/0 NEB feature

Options:
```

```

-h, --help          show this help message and exit

-i INITIAL_FILE, --initial=INITIAL_FILE

                        initial coordinates. Required

-f FINAL_FILE, --final=FINAL_FILE

                        final coordinates. Required

--initial_format=INITIAL_FORMAT

                        format for the initial coordinate file. Defaults to
                        phase0-out

--final_format=FINAL_FORMAT

                        format for the final coordinate file. Defaults to
                        phase0-out

-n NIMAGE, --nimage=NIMAGE

                        number of intermediate images. defaults to 6

-l, --linear         specify this option in order to create replicas by
                        simple linear interpolation. the default is to create
                        replicas by the IDPP method.

```

利用可能なオプションは、下記の通り。

オプション	説明
-h, --help	ヘルプメッセージを表示し、終了する。
-i INITIAL_FILE, --initial=INITIAL_FILE	始状態の座標データファイルを指定する。必須オプション。
--initial_format=INITIAL_FORMAT	始状態の座標データファイルの種類を指定する。この指定がない場合、PHASE/0 の出力形式 (F_DYNM 形式)であると判定される。
--final_format=FINAL_FORMAT	終状態の座標データファイルの種類を指定する。この指定がない場合、PHASE/0 の出力形式 (F_DYNM 形式)であると判定される。
-n NIMAGE, --nimage=NIMAGE	中間イメージの数を指定する。デフォルト値は 6.
-l, --linear	IDPP 法によってレプリカ列を求めるのがデフォルトの振る舞いであるが、このオプションをつけると線形補間によってレプリカ列を作成することができる。

レプリカ列は、`imagex.data` というファイルと `initial_replicas.cif` という CIF に記録されます。`imagex.data` ( $x$  は 0 始まりの整数) は PHASE/0 が NEB 計算時に外部ファイルとして読み込み可能な座標データファイルです。 $x=0$  が始状態,  $x=nr-1$  ( $nr$  はレプリカ数) が終状態に対応する座標データファイルです。中間イメージは,  $x=1$  から  $x=nr-2$  のファイルに記録されます。これらを PHASE/0 から参照するためには, 以下のように入力ファイルを作成する必要があります。

```
multiple_replica{
...
  structure{
    number_of_replicas = 6
    replicas{
      #tag replica_number  howtogive_coordinates
      1          file
      2          file
      3          file
      4          file
      5          file
      6          file
    }
    endpoint_images = directin
  }
}
...
...
```

`multiple_replica` ブロックの下に `structure` ブロックでレプリカ列の指定を行います。`number_of_replicas` に中間イメージの数を指定し, その下の `replicas` ブロックにおいて中間イメージの作成方法を指定します。`file` とするとファイルから読み込むという設定となります。また, `endpointimages` に `file` を指定すると始状態および終状態のデータをファイルから読み込むことができます。ただし, 後述のように始状態と終状態のファイルからの読み込みには不具合があるので, 現時点では入力パラメーターファイルで直接指定する方法を採用する必要があります。

また, `filenames.data` に以下のような記述を行います。

```
&fname  
F_POT(1) = ...  
...  
/  
&nebfiles  
F_IMAGE(1) = './image1.data'  
F_IMAGE(2) = './image2.data'  
...  
/
```

&nebfiles セクションを作成し、座標データファイルを指定します。F\_IMAGE(0)および F\_IMAGE(-1)は始状態および終状態の座標データファイル用のファイルポインターであり、F\_IMAGE(1)から F\_IMAGE(N)が中間イメージの座標データファイル用のファイルポインターです。N は中間イメージの数です。

ただし、現バージョン(2018.01 版)の PHASE/0 には、始状態および終状態をファイルから指定すると異常終了してしまう不具合があります。そこで、始状態および終状態の指定に関しては上述の方法ではなく入力パラメーターファイルに直接記述する方法を利用するようにしてください。