ablCS Documentation

リリース *2.0.1*

ablCS's developer team

2022 年 11 月 04 日

目 次

第1章　abICS とは？

1.1 概要

abICS は、第一原理計算を再現する機械学習モデルを訓練し、不規則系での統計熱力学サンプリングを高速に実行するためのソフトウェアフレームワークです。金属や酸化物合金などの多成分固体系に特に重点を置いています。現在は aenet で実装されているニューラルネットワークポテンシャルを機械学習モデルとして利用することができます。機械学習の基となる第一原理計算用入力ファイルの自動生成にも対応しており、Quantum Espresso, VASP, OpenMX を利用することができます。

1.2 開発者

abICS は以下のメンバーで開発しています.

- **ver. 2.0**
  - 笠松 秀輔 (山形大学 学術研究院 (理学部主担当))
  - 本山 裕一 (東京大学 物性研究所)
  - 青山 龍美 (東京大学 物性研究所)
  - 吉見 一慶 (東京大学 物性研究所)
  - 杉野 修 (東京大学 物性研究所)
- **ver. 1.0**
  - 笠松 秀輔 (山形大学 学術研究院 (理学部主担当))
  - 本山 裕一 (東京大学 物性研究所)
  - 吉見 一慶 (東京大学 物性研究所)
  - 山本 良幸 (東京大学 物性研究所)
  - 杉野 修 (東京大学 物性研究所)
  - 尾崎 泰助 (東京大学 物性研究所)

1.3 バージョン履歴

- ver.2.0.1 : 2022/11/04.
- ver.2.0 : 2022/06/24.
- ver.1.0 : 2020/05/01.
- ver.1.0-beta : 2020/03/31.
- ver.0.1 : 2019/12/09.

1.4 ライセンス

本ソフトウェアのプログラムパッケージおよびソースコード一式は GNU General Public License version 3（GPL v3）に準じて配布されています。

1.5 コピーライト

本ソフトウェアは 2019, 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。

第2章　インストール方法

2.1 必要なライブラリ・環境

abICS をインストール・実行するには、 バージョン 3.7 以上の Python が必要です。また、以下の Python パッケージが必要です。

- numpy
- scipy
- toml
- mpi4py
- pymatgen (>=2019.12.3)
- qe-tools

これらのライブラリは自動でインストールされますが、 mpi4py と pymatgen はあらかじめ関連ソフトウェアが必要です。

- mpi4py をインストールするには、なんらかの MPI 環境をあらかじめインストールしておく必要があります。
- pymatgen をインストールするには、 Cython をインストールしておく必要があります。:

```
$ pip3 install cython
```

2.2 PyPI からインストールする

abICS は PyPI に登録されているため、 `pip` コマンドで簡単にインストールできます。:

```
$ pip3 install abics
```

書き込み権限がないなどで、ユーザローカルのディレクトリにインストールする場合には `--user` オプションを追加してください。この場合、 `~/.local/` 以下に実行可能スクリプトやライブラリがインストールされます。また、インストールディレクトリを指定したい場合には、 `--prefix=DIRECTORY` (`DIRECTORY` はインストールしたいディレクトリ) オプションを指定してください:

```
$ pip3 install --user abics
```

2.3 ソースからインストールする

多くの場合には PyPI からインストールすれば良いですが、機能追加する場合などはソースからインストールしてください。

2.3.1 ダウンロード

abICS のソースコードは GitHub page からダウンロードできます。

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

2.3.2 ディレクトリ構成

abICS のディレクトリ構成は以下のようになっています. python モジュールは `abics` ディレクトリ以下に一式格納されています.

```
.
|-- COPYING
|-- README.md
|-- abics/
|   |-- __init__.py
|   |-- applications/
|   |-- exception.py
|   |-- mc.py
|   |-- mc_mpi.py
|   |-- replica_params.py
|   |-- scripts/
|   |-- util.py
|-- docs/
|   |-- sphinx/
|-- examples/
|-- pyproject.toml
|-- test/
|-- tests/
```

2.3.3 インストール

- `pip3 install` の引数に abICS のルートディレクトリを渡すことでインストール可能です

```
$ pip3 install ./abICS
```

2.4 アンインストール

- `pip3 uninstall abics` でアンインストールできます.

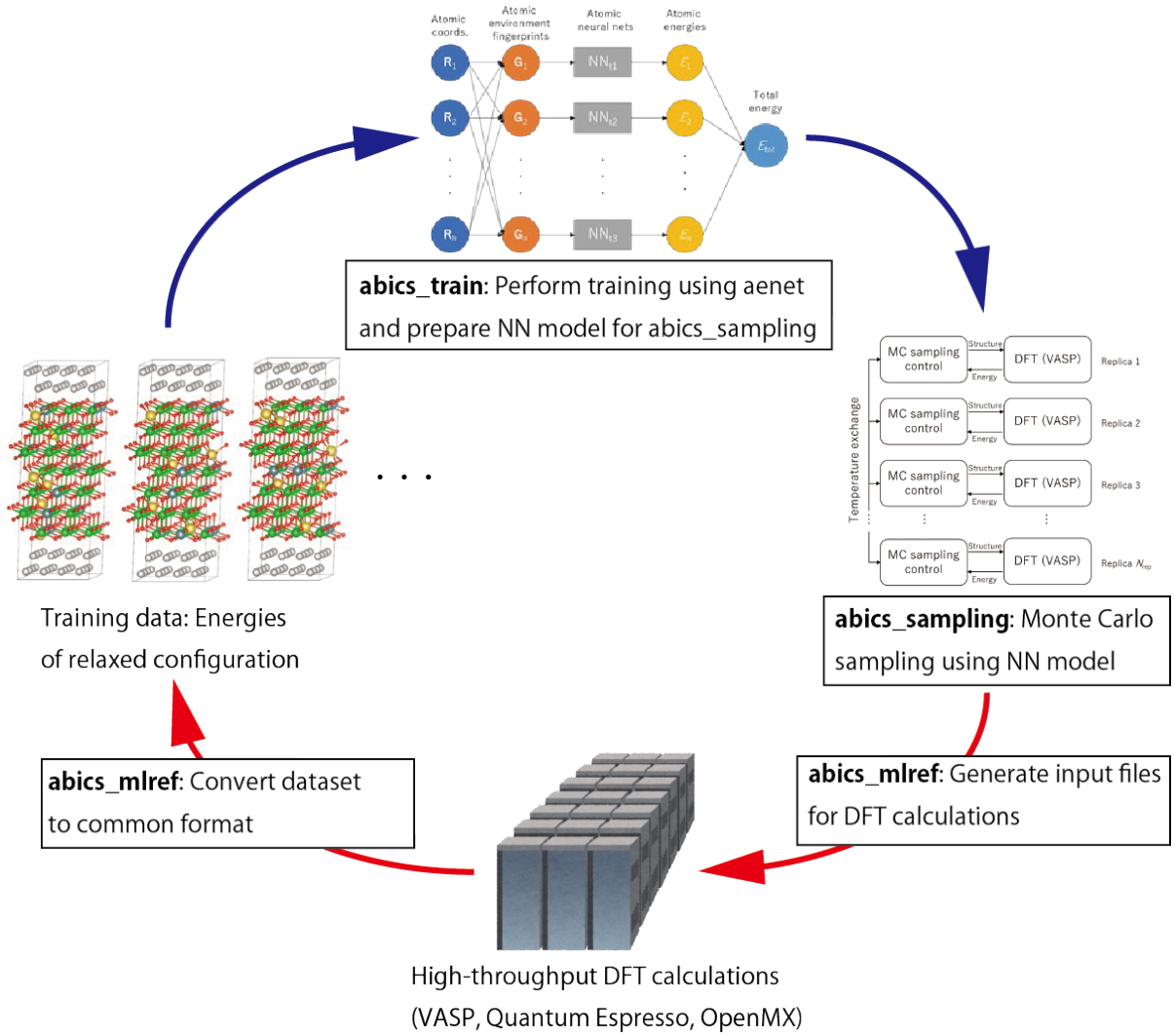
第3章　基本的な使用方法

3.1 能動学習について

abICS は元々、第一原理計算とレプリカ交換モンテカルロ法を直接組み合わせて統計熱力学計算を行うことを念頭に開発されましたが、計算できるモデル規模やステップ数が、第一原理計算の大きな計算コストのために限られてしまいます。これに対して、Ver. 2 では、構造最適化後のエネルギーを高速に予測する機械学習モデルを構成するための能動学習手法を実装し、飛躍的にサンプリング速度を向上させました [論文プレプリント] 。

abICS に実装されている能動学習手法の大まかな流れは以下の通りです。

1. ランダムに生成した多数の原子配置に対して第一原理計算を行い、訓練データ（原子配置とエネルギーの対応関係）を用意する
2. 用意した訓練データを使って、原子配置からエネルギーを予測する機械学習モデルを構築する
3. 機械学習モデルを使って、レプリカ交換モンテカルロ法による原子配置の統計熱力学サンプリングを行う
4. モンテカルロ計算で出現したイオン配置をサンプリングし、それぞれに対して第一原理計算を行うことで、機械学習モデルの精度の評価を行う
5. 不十分であった場合は、4. で計算した結果を訓練データに追加し、2. から繰り返す。



abICS を用いた能動学習のイメージ図

3.2 abICS 制御用入力ファイルの準備

まず、abICS の動作全般を制御する入力ファイルを作成します。abICS の入力ファイルは, 以下の 5 つのセクションから構成されます.

1. [sampling] セクション

   レプリカ数や温度の幅, モンテカルロステップ数など, レプリカ交換モンテカルロ部分のパラメータを指定します. また、利用するソルバーの種類 (VASP, QE, ...)、ソルバーへのパス、不変な入力ファイルのあるディレクトリなど（第一原理計算）ソルバーのパラメータを指定します.

2. [mlref] セクション

   ニューラルネットワークモデルの精度評価と訓練データの拡張などを行うため、サンプリングの結果から原子配置のみを取り出す際のオプションを設定します. `abics_mlref` のみで使用されます. また、訓練データを作成するために利用するソルバーの種類 (VASP, QE, ...)、ソルバーへのパス、不変な入力ファイルのあるディレクトリなど（第一原理計算）ソルバーのパラメータを指定します.

3. [train] セクション

   訓練データから配置エネルギー予測モデルを学習する学習器の設定を行います. abics_train のみで使用されます.

4. [observer] セクション

   取得する物理量の種類などを指定します.

5. [config] セクション

   合金の配位などを指定します.

これらの詳細については **入力ファイルフォーマット** をご覧ください。以下に入力ファイルの例を記載します。

```
[sampling]
nreplicas = 8
nprocs_per_replica = 1
kTstart = 600.0
kTend = 2000.0
nsteps = 6400 # Number of steps for sampling
RXtrial_frequency = 4
sample_frequency = 16
print_frequency = 1
reload = false

[sampling.solver]
type = 'aenet'
path= 'predict.x-2.0.4-ifort_serial'
base_input_dir = './baseinput'
perturb = 0.0
run_scheme = 'subprocess' #'mpi_spawn_ready'
ignore_species = ["O"]

[mlref]
nreplicas = 8
ndata = 5

[mlref.solver]
type = 'qe'
base_input_dir = './baseinput_ref'
perturb = 0.05
ignore_species = []

[train]
type = 'aenet'
base_input_dir = './aenet_train_input'
exe_command = ['generate.x-2.0.4-ifort_serial', 'srun train.x-2.0.4-ifort_intelmpi']
```

(次のページに続く)

(前のページからの続き)

```
ignore_species = ["O"]
vac_map = []
restart = false

[config]
unitcell = [[8.1135997772, 0.0000000000, 0.0000000000],
            [0.0000000000, 8.1135997772, 0.0000000000],
            [0.0000000000, 0.0000000000, 8.1135997772]]
supercell = [1,1,1]

[[config.base_structure]]
type = "O"
coords = [
     [0.237399980, 0.237399980, 0.237399980],
     [0.762599945, 0.762599945, 0.762599945],
     [0.512599945, 0.012600004, 0.737399936],
     [0.487399966, 0.987399936, 0.262599975],
     [0.012600004, 0.737399936, 0.512599945],
     [0.987399936, 0.262599975, 0.487399966],
     [0.737399936, 0.512599945, 0.012600004],
     [0.262599975, 0.487399966, 0.987399936],
     [0.987399936, 0.487399966, 0.262599975],
     [0.012600004, 0.512599945, 0.737399936],
     [0.487399966, 0.262599975, 0.987399936],
     [0.512599945, 0.737399936, 0.012600004],
     [0.262599975, 0.987399936, 0.487399966],
     [0.737399936, 0.012600004, 0.512599945],
     [0.237399980, 0.737399936, 0.737399936],
     [0.762599945, 0.262599975, 0.262599975],
     [0.512599945, 0.512599945, 0.237399980],
     [0.487399966, 0.487399966, 0.762599945],
     [0.012600004, 0.237399980, 0.012600004],
     [0.987399936, 0.762599945, 0.987399936],
     [0.987399936, 0.987399936, 0.762599945],
     [0.012600004, 0.012600004, 0.237399980],
     [0.487399966, 0.762599945, 0.487399966],
     [0.512599945, 0.237399980, 0.512599945],
     [0.737399936, 0.237399980, 0.737399936],
     [0.262599975, 0.762599945, 0.262599975],
     [0.237399980, 0.512599945, 0.512599945],
     [0.762599945, 0.487399966, 0.487399966],
     [0.762599945, 0.987399936, 0.987399936],
     [0.237399980, 0.012600004, 0.012600004],
     [0.737399936, 0.737399936, 0.237399980],
     [0.262599975, 0.262599975, 0.762599945],
```

(次のページに続く)

(前のページからの続き)

```
    ]

[[config.defect_structure]]
coords = [
    [0.000000000, 0.000000000, 0.000000000],
    [0.749999940, 0.249999985, 0.499999970],
    [0.249999985, 0.749999940, 0.499999970],
    [0.249999985, 0.499999970, 0.749999940],
    [0.749999940, 0.499999970, 0.249999985],
    [0.499999970, 0.749999940, 0.249999985],
    [0.499999970, 0.249999985, 0.749999940],
    [0.000000000, 0.499999970, 0.499999970],
    [0.749999940, 0.749999940, 0.000000000],
    [0.249999985, 0.249999985, 0.000000000],
    [0.249999985, 0.000000000, 0.249999985],
    [0.749999940, 0.000000000, 0.749999940],
    [0.499999970, 0.000000000, 0.499999970],
    [0.000000000, 0.749999940, 0.749999940],
    [0.000000000, 0.249999985, 0.249999985],
    [0.499999970, 0.499999970, 0.000000000],
    [0.374999970, 0.374999970, 0.374999970],
    [0.624999940, 0.624999940, 0.624999940],
    [0.374999970, 0.874999940, 0.874999940],
    [0.624999940, 0.124999993, 0.124999993],
    [0.874999940, 0.874999940, 0.374999970],
    [0.124999993, 0.124999993, 0.624999940],
    [0.874999940, 0.374999970, 0.874999940],
    [0.124999993, 0.624999940, 0.124999993],
    ]
[[config.defect_structure.groups]]
name = 'Al'
# species = ['Al']    # default
# coords = [[[0,0,0]]]  # default
num = 16 #432 #16000
[[config.defect_structure.groups]]
name = 'Mg'
# species = ['Mg']    # default
# coords = [[[0,0,0]]]  # default
num = 8 #216 #8000


[observer]
ignored_species = ['O']
```

3.3 第一原理ソルバー用参照ファイルの準備

訓練データの生成に用いる第一原理ソルバーの入力形式に従った入力ファイルを用意します。参照ファイルのパスは abICS の入力ファイルにある `[mlref.solver]` セクションの `base_input_dir` で指定します。座標情報については記載する必要はありません。以下、Quantum ESPRESSO の参照ファイルの例について記載します。

```
&CONTROL
  calculation = 'relax'
  tstress = .false.
  tprnfor = .false.
  pseudo_dir = './pseudo'
  disk_io = 'low'
  wf_collect = .false.
/
&SYSTEM
  ecutwfc      =  60.0
  occupations  = "smearing"
  smearing     = "gauss"
  degauss      = 0.01
/
&electrons
  mixing_beta = 0.7
  conv_thr = 1.0d-8
  electron_maxstep = 100
/
&ions
/
ATOMIC_SPECIES
Al 26.981 Al.pbe-nl-kjpaw_psl.1.0.0.UPF
Mg 24.305 Mg.pbe-spnl-kjpaw_psl.1.0.0.UPF
O  16.000 O.pbe-n-kjpaw_psl.1.0.0.UPF
ATOMIC_POSITIONS crystal

K_POINTS gamma
```

3.3.1 第一原理ソルバー利用時の注意点

原子座標以外の設定については基本的にソルバーごとに指定する必要があります。ただし、構造最適化をする原子の指定については abICS 側で制御することが可能です。構造最適化機能を有効にする場合には、ソルバーの参照ファイルで構造最適化オプションを有効にした上で、構造最適化のステップ数なども指定することで最適化が行われます。また、abICS ではソルバー毎に、参照ファイル名、実装時に仮定している参照ファイルのルールなどがあります。以下、それらについて説明します。

VASP

- URL : https://www.vasp.at
- 参照ファイル
  - INCAR, POTCAR, KPOINTS ファイルを用意してください。
    - POTCAR ファイルは元素をアルファベット順に並べてください。
    - POSCAR ファイルは不要ですが、依存パッケージである `pymatgen` のバージョンによっては必要になります。その場合、なにか適当なファイルを用意してください。

Quantum Espresso

- URL : https://www.quantum-espresso.org
- バージョンは 6.2 以上を利用してください。
  - いわゆる旧形式 XML バージョンは利用できません。
- 参照ファイル
  - 参照ファイル名は `scf.in` にしてください。
  - `calculation` は `scf` と `relax` のみ対応しています。
  - Γ 点のみで計算する場合には、 `kpoints` を `Gamma` に指定すると高速化します。

OpenMX

- URL : http://www.openmx-square.org
- バージョンは 3.9 を利用してください。
- 参照ファイル
  - 参照ファイル名は `base.dat` にしてください。

3.4 機械学習モデル訓練および評価用参照ファイルの準備

使用する機械学習モデルソルバー（現在は aenet のみに対応）の入力形式に従った入力ファイルを用意します。参照ファイルのパスは abICS の入力ファイルにある `[solver]` セクションの `base_input_dir` で指定します。座標情報については、abICS の入力ファイルを参照するため、記載する必要はありません。

3.4.1 機械学習モデルソルバー利用時の注意点

aenet

- URL : http://ann.atomistic.net
- バージョン 2.0.4 で動作確認済。
- 参照ファイル (参照ファイルの具体例についてはチュートリアル参照)
  - aenet 用の入力ファイルを [train] セクションの base_input_dir で設定したディレクトリ内の generate 、 train 、および predict ディレクトリに設置してください。
  - aenet では、訓練用の原子配置とエネルギーのデータを、原子環境記述子とエネルギーの関係に変換した中間バイナリフォーマットにまとめてから訓練を行います。この変換を行う generate.x 用の 入力ファイルを generate ディレクトリに設置してください。
  - generate.x で生成された訓練データを読み込み、訓練を行う train.x 用の入力ファイルを train ディレクトリに設置します。 ファイル名は train.in としてください。
  - 訓練したポテンシャルモデルを使って入力座標に対してエネルギーを 評価するための predict.x 用の入力ファイル predict.in を、 predict ディレクトリに設置してください。
- abICS 入力ファイル
  - [solver] セクションで type , perturb , run_scheme に関しては、能動学習スキームを用いる場合は以下に設定してください。

```
type = "aenet"
perturb = 0.0
run_scheme = 'subprocess'
```

3.5 学習データの作成

1. abics_mlref を用いて訓練データの大元となる第一原理計算用の入力ファイルを生成します。
2. 1 で生成した入力ファイルに対して第一原理計算を実施します (チュートリアルでは GNU parallel を利用し網羅計算を実行しています)。

3.6 機械学習モデルの作成

1. abics_mlref を再度実行して、学習で用いる abics_train が読み込めるよう第一原理計算の結果を変換します。
2. 次に abics_train を実行して機械学習モデルの作成を行います。 計算が無事終了すると、baseinput ディレクトリに学習済みの機械学習モデル（ニューラルネットワーク）が出力されます。

3.7 モンテカルロサンプリングの実行

`abics_sampling` を用いてモンテカルロサンプリングを行います (MPI 実行時に指定するプロセス数はレプリカ数以上である必要があります)。実行すると、カレントディレクトリ以下にレプリカ番号を名前にもつディレクトリが作られ、各レプリカはその中でソルバーを実行します。

第4章　チュートリアル

このチュートリアルでは多元系イオン結晶 $MgAl_2O_4$ の Mg, Al 原子の反転度計算を例に、abICS の利用方法について説明します。入力ファイルは `examples/active_learning_qe/` にあります。

4.1 ニューラルネットワークの構築

ここでは、aenet を用いてニューラルネットワークモデルの構築を行う方法について記載します。第一原理ソルバーには Quantum ESPRESSO (QE) を使います。例題として MgAl2O4 スピネルの Mg/Al 反転度の温度依存性を計算します。最安定構造では全ての Mg が酸素に四面体配位されていて、Al は八面体配位になっています。このサイト間の反転度の温度依存性をシミュレーションします。なお、本チュートリアルで使用する入力ファイル一式は `examples/active_learning_qe/` にあります。以下では aenet および GNU parallel のインストールについても簡単に説明していますが、システムに既にインストールされている場合はそちらを使ってください。なお、計算の実行環境は物性研究所スーパーコンピュータシステム B の ohtaka を利用しています。

4.1.1 事前準備

aenet のインストール

abICS では、ニューラルネットワークモデルの構築のために aenet を利用します。aenet は http://ann.atomistic.net からダウンロードできます。Documentation の Installation に従ってインストールしてください。なお、abICS では、ニューラルネットワークの学習と評価に aenet の `train.x` と `predict.x` を使います。`train.x` については MPI 並列版が利用可能ですが、 `predict.x` については MPI を使用しない実行ファイル (serial) を使用する必要があります。そのため、makefiles の下にある serial 版もインストールするようにしてください。

GNU parallel のインストール

チュートリアルでは、GNU parallel を用いて Quantum Espresso による第一原理計算を並列実行します。GNU parallel は https://www.gnu.org/software/parallel/ からダウンロードできます (Mac の場合は homebrew により直接インストールすることも可能です)。インストールは基本的には、ダウンロードして解凍したディレクトリ に移動した後、

```
$ ./configure && make && make install
```

でインストールできます。詳細な設定は公式マニュアルを参考にしてください。

4.1.2 学習データ生成用入力ファイルの準備

ニューラルネットワークの作成にあたり、原子配置を入力、エネルギーを出力とした学習データを、第一原理計算により作成する必要があります。学習データの作成には、

- abICS
- 利用する第一原理ソルバー

に関する入力ファイルを準備する必要があります。

abICS 制御ファイル (input.toml)

計算対象とする格子構造の定義と、abICS による能動学習のループ全体の制御、およびレプリカ交換モンテカルロ法に関するパラメータを設定します。st2abics ツールを使うことで、結晶構造ファイルから input.toml のひな形を自動で生成することができます。

```
$ cd [example_dir]
$ st2abics st2abics_MgAl2O4.toml MgAl2O4.vasp > input.toml
```

以下では input.toml の中で第一原理計算によ る学習データに関連するセクションの設定内容をもう少し詳しく解説します。

(i) [mlref] セクション

```
[mlref]
nreplicas = 8
ndata = 5
```

RXMC 計算の結果から、ニューラルネットワークモデルの精度評価と訓練データの拡張のために原子配置を取り出す際のオプションが設定できます。基本的に、nreplicas は 後述する [sampling] セクションと同じ値にしてください。ndata は、RXMC 計算で出力される配置の数（後述する [sampling] セクションの nsteps/sample_frequency の値）から、何個のデータを機械学習用に取り出すかを指定します。従って、RXMC 計算で出力される配置の数以下の値に設定してください。

(ii) [mlref.solver] セクション

```
[mlref.solver] # 参照第一原理ソルバーの設定
type = 'qe'
base_input_dir = ['./baseinput_ref', './baseinput_ref', './baseinput_ref'] #, './
↪baseinput_ref']
perturb = 0.05
ignore_species = []
```

訓練データ（配置エネルギー）の計算に用いるソルバーの設定を行います。この例では Quantum Espresso で配置エネルギーを求めます。base_input_dir は自由に設定して構いません。設定したディレクトリの中に、ソルバーの入力ファイルを設置します（後述）。

上記の例のようにリスト形式で複数設定した場合は、各々のディレクトリ内の入力を使った計算が順番に実行されます。このときに、2 番目以降の計算では前の計算の最終ステップでの構造が初期座標として用いられます。そして、最後の計算のエネルギーが学習に使われます。例えば、1 つ目の入力ファイルでで精度を犠牲にして高速な構造最適化を行い、2 番目以降の入力ファイルで高精度な設定で構造最適化を行うといったことが可能になります。あるいは、格子ベクトルの緩和を行う場合に、設定した平面波カットオフに基づいて計算メッシュをリセットするために同じ入力の計算を複数回実行するといったことも可能です。

perturb は、ランダムに各原子を変位させることで、対称性を崩した構造から構造最適化を開始するための設定です。この場合は、構造緩和を行う原子を全て 0.05 Å、ランダムな方向に変位させた構造から 1 番目の計算が開始されます。

ignore_species は、第一原理ソルバーを訓練データ生成に用いる場合は空リストを指定しますが、一部の元素を無視するようなモデルを使って訓練データを生成する場合は、無視する元素を指定します。

(iii) [config] セクション

```
[config] # 以下、結晶格子の情報と、格子上に配置される原子や空孔の情報が続く
unitcell = [[8.1135997772, 0.0000000000, 0.0000000000],
            [0.0000000000, 8.1135997772, 0.0000000000],
            [0.0000000000, 0.0000000000, 8.1135997772]]
supercell = [1,1,1]

[[config.base_structure]]
type = "O"
coords = [
    [0.237399980, 0.237399980, 0.237399980],
    [0.762599945, 0.762599945, 0.762599945],
    [0.512599945, 0.012600004, 0.737399936],
    [0.487399966, 0.987399936, 0.262599975],
    ...
```

基本的に st2abics ツールで生成されたものをそのまま利用できます。モンテカルロサンプリングを行う原子配置の情報が設定されます。abics_sampling が未実施の場合には、この情報をもとに原子配置がランダムに与えられ、それらの原子配置を持った第一原理計算用の入力ファイルが生成されます。abics_sampling が既に実行されている場合には、 config セクションではなく、モンテカルロサンプリングで与えられた原子配置に対する第一原理計算用の入力ファイルが生成されます。

QE 参照ファイルの準備

baseinput_ref に QE の scf 計算で参照する入力ファイルを配置します。以下、サンプルディレクトリにある scf.in ファイルを記載します。

```
&CONTROL
calculation = 'relax'
tstress = .false.
tprnfor = .false.
pseudo_dir = './pseudo'
disk_io = 'low'
wf_collect = .false.
/
&SYSTEM
  ecutwfc      =  60.0
  occupations  = "smearing"
  smearing     = "gauss"
  degauss      = 0.01
/
&electrons
  mixing_beta = 0.7
  conv_thr = 1.0d-8
  electron_maxstep = 100
/
&ions
/
ATOMIC_SPECIES
Al 26.981 Al.pbe-nl-kjpaw_psl.1.0.0.UPF
Mg 24.305 Mg.pbe-spnl-kjpaw_psl.1.0.0.UPF
O  16.000 O.pbe-n-kjpaw_psl.1.0.0.UPF
ATOMIC_POSITIONS crystal

K_POINTS gamma
```

なお、擬ポテンシャルを格納したディレクトリ pseudo_dir や ATOMIC_SPECIES で使用する擬ポテンシャルについて、自分の環境に従い書き換える必要があります。本サンプルで使用している擬ポテンシャルは以下のリンクからダウンロードできます。（ダウンロードスクリプト download_pp.sh が用意されています。）

- https://pseudopotentials.quantum-espresso.org/upf_files/Al.pbe-nl-kjpaw_psl.1.0.0.UPF
- https://pseudopotentials.quantum-espresso.org/upf_files/Mg.pbe-spnl-kjpaw_psl.1.0.0.UPF
- https://pseudopotentials.quantum-espresso.org/upf_files/O.pbe-n-kjpaw_psl.1.0.0.UPF

このサンプルでは、QE 計算時に構造最適化を行うため calculation = 'relax' を、計算高速化のため、K_POINTS は gammma を選択しています。

4.1.3 ニューラルネットワーク生成用入力ファイルの準備

本チュートリアルでは aenet を用いニューラルネットワークを作成します。ニューラルネットワークの作成には、

- abICS
- aenet

に関する入力ファイルを準備・設定する必要があります。

abICS 制御ファイル (input.toml)

(i) [train] セクション

```
[train] # モデル学習器の設定
type = 'aenet'
base_input_dir = './aenet_train_input'
exe_command = ['generate.x-2.0.4-ifort_serial',
               'srun train.x-2.0.4-ifort_intelmpi']
ignore_species = ["O"]
vac_map = []
restart = false
```

訓練データから配置エネルギー予測モデルを学習する学習器の設定を行います。現在のところ、abICS では aenet のみに対応しています。 base_input_dir は自由に設定して構いません。設定したディレクトリの中に、学習器の設定ファイルを設置します（後述）。 exe_command には aenet の generate.x と train.x へのパスを指定します。train.x については MPI 並列版が利用可能で、その場合は、上の例で示すように、MPI 実行するためのコマンド（ srun 、 mpirun など）を合わせて設定してください。

ignore_species は、第一原理ソルバーを訓練データ生成に用いる場合は空リストを指定しますが、一部の元素を無視するようなモデルを使って訓練データを生成する場合は、無視する元素を指定します。vac_map 、 restart については現状対応していないので、例のように設定してください。

aenet 用の入力ファイル

aenet 用の入力ファイルを [train] セクションの base_input_dir で設定したディレクトリ内の generate 、 train 、および predict ディレクトリに設置します。

generate

aenet では、訓練用の原子配置とエネルギーのデータを、原子環境記述子とエネルギーの関係に変換した中間バイナリフォーマットにまとめてから訓練を行います。この変換を行う generate.x 用の入力ファイルを generate ディレクトリに設置します。

まず、元素種ごとの記述子設定ファイルを用意します。ファイル名は任意ですが、チュートリアルでは Al.fingerprint.stp , Mg.fingerprint.stp という名前にしています。例として Al.fingerprint.stp の内容を示します：

```
DESCR
 N. Artrith and A. Urban, Comput. Mater. Sci. 114 (2016) 135-150.
 N. Artrith, A. Urban, and G. Ceder, Phys. Rev. B 96 (2017) 014112.
END DESCR

ATOM Al # 元素を指定

ENV 2 # ATOM で指定した元素と相互作用する元素種の数と元素名を指定
Al
Mg

RMIN 0.55d0 # 原子間の最隣接距離

BASIS type=Chebyshev # チェビシェフ記述子の設定
radial_Rc = 8.0  radial_N = 16 angular_Rc = 6.5  angular_N = 4
```

記述子設定の詳細については aenet のドキュメントを参照してください。

次に、generate.in.head という名前で以下の内容のファイルを準備します：

```
OUTPUT aenet.train

TYPES
2
Al -0.0  ! eV
Mg -0.0  ! eV

SETUPS
Al   Al.fingerprint.stp
Mg    Mg.fingerprint.stp
```

OUTPUT には必ず aenet.train を指定してください。TYPES 以下には訓練データ中の元素種とその数を指定します。元素種ごとにエネルギーの基準を指定することもできますが、基本的には 0 に設定しておくのが無難です。SETUPS 以下には元素種ごとの記述子設定ファイルを指定します。ファイルの末尾には必ず改行が入っていることを確認してください。abICS は generate.in.head の末尾に座標ファイルのリストを追加して generate.in を生成し、generate.x を実行します。

train

generate で生成された訓練データを読み込み、訓練を行う train.x 用の入力ファイルを train ディレクトリに設置します。ファイル名は train.in としてください：

```
TRAININGSET aenet.train
TESTPERCENT 10
ITERATIONS  500

MAXENERGY 10000

TIMING

!SAVE_ENERGIES

METHOD
bfgs

NETWORKS
! atom   network         hidden
! types  file-name       layers  nodes:activation
  Al     Al.15t-15t.nn    2      15:tanh 15:tanh
  Mg       Mg.15t-15t.nn    2      15:tanh 15:tanh
```

基本的には、 NETWORKS セクション以外は変更の必要はありません。NETWORKS セクションでは、生成する元素種ごとのポテンシャルファイル名と、ニューラルネットワーク構造、および活性化関数を指定します。

predict

訓練したポテンシャルモデルを使って入力座標に対してエネルギーを評価するための predict.x 用の入力ファイル predict.in を、 predict ディレクトリに設置します：

```
TYPES
2
Mg
Al

NETWORKS
Mg  Mg.15t-15t.nn
Al  Al.15t-15t.nn

VERBOSITY low
```

TYPES セクションには元素種の数と元素名を、 NETWORKS セクションには元素種ごとのポテンシャルファイル名（ train.in で設定したもの）を入力してください。

また、 VERBOSITY は必ず low に設定してください。

- [sampling.solver] セクションの path を実行環境における aenet の predict.x のパスに設定する。
- [sampling.solver] と [train] セクションで ignore_species = ["O"] を指定する。

計算実行

入力ファイルの準備・設定が完了後、実際に計算する方法について説明します。サンプルスクリプトには、計算手順を簡略化するためのスクリプト AL.sh が準備されています。run_pw.sh は QE の計算を実行するためのスクリプトで、後述する parallel_run.sh 内部で呼び出されます。AL.sh の中身は以下の通りです。

```
#!/bin/sh
#SBATCH -p i8cpu
#SBATCH -N 4
#SBATCH -n 512
#SBATCH -J spinel
#SBATCH -c 1
#SBATCH --time=0:30:00

# Run reference DFT calc.
echo start AL sample
srun -n 8 abics_mlref input.toml >> abics_mlref.out

echo start parallel_run 1
sh parallel_run.sh

echo start AL final
srun -n 8 abics_mlref input.toml >> abics_mlref.out

#train
echo start training
abics_train input.toml >> abics_train.out

echo Done
```

先頭の #SBATCH で始まる数行は物性研スパコンでのジョブスケジューラに関するコマンドです。ここでは、プロセス数 512 の MPI 並列を指定しています。また、 srun はスパコンの並列環境でプログラムを実行するためのコマンドです（ mpiexec に相当します)。ジョブスケジューラに関する詳細は、実際に利用する計算機システムのマニュアルを参照してください。

```
# Run reference DFT calc.
echo start AL sample
srun -n 8 abics_mlref input.toml >> abics_mlref.out
```

まず、 abics_mlref を用いて訓練データの大元となる第一原理計算用の入力ファイルを生成します。初回実行時は、指定した数の原子配置をランダムに生成し、それぞれの原子配置ごとに個別のディレクトリを

用意した上で、入力ファイルをその中に作成します。あわせて、これらのディレクトリの path を記載したファイル rundirs.txt を生成します。

次に、得られたファイルをもとに第一原理計算を実行します。

```
echo start parallel_run 1
sh parallel_run.sh
```

parallel_run.sh は、GNU parallel を用いて QE の網羅計算を行うためのスクリプトで、rundirs.txt に記載されたディレクトリを対象に QE の網羅計算が行われます。QE の計算結果はそれぞれのディレクトリに格納されます。

QE の網羅計算により教師データが作成されましたので、次に aenet でのニューラルネットワークポテンシャルの作成に移ります。最初に、 abics_mlref を再度実行し、第一原理計算の結果を abics_train が読み込む共通フォーマットに変換したファイルを作成します。

```
echo start AL final
srun -n 8 abics_mlref input.toml >> abics_mlref.out
```

次に、学習データをもとに aenet によりニューラルネットワークポテンシャルの作成を行います。ニューラルネットワークポテンシャルの計算には abics_train を使います。[train] セクションの base_input_dir で指定したディレクトリに格納された入力ファイルを読み込み、計算を行います。計算が無事終了すると、baseinput ディレクトリに学習済みのニューラルネットワークが出力されます。

```
#train
echo start training
abics_train input.toml >> abics_train.out
```

以上の手続きで、能動学習を行うための AL.sh のプロセスが完了します。

4.2 モンテカルロサンプリング

次に、学習したニューラルネットワークポテンシャルを用い、abICS によりモンテカルロサンプリングをします。

4.2.1 入力ファイルの準備

abICS でモンテカルロサンプリングを行うには abICS 制御ファイルでパラメータの設定をする必要があります。

abICS 制御ファイル (input.toml)

レプリカ交換モンテカルロ法に関連するセクション [sampling] で計算パラメータを設定します。

(i) [sampling] セクション

```
[sampling]
nreplicas = 8
nprocs_per_replica = 1
kTstart = 600.0
kTend = 2000.0
nsteps = 6400
RXtrial_frequency = 4
sample_frequency = 16
print_frequency = 1
reload = false
```

レプリカ交換モンテカルロ (RXMC) 法のレプリカの数や温度範囲などに関する設定を行います（**入力ファイルフォーマット**）。今回は、RXMC 計算のエネルギーソルバーとして aenet の predict.x を用います。現状、MPI 版の predict.x はサポートしていないため、 nprocs_per_replica は 1 を指定してください。基本的に、 nreplicas は [mlref] セクションと同じ値にしてください。

(ii) [sampling.solver] セクション

```
[sampling.solver] # RXMC計算に使うソルバーの設定
type = 'aenet'
path= 'predict.x-2.0.4-ifort_serial'
base_input_dir = './baseinput'
perturb = 0.0
run_scheme = 'subprocess'
ignore_species = ["O"]
```

RXMC 計算に使うエネルギーソルバーの設定を行います。今回は、aenet を使ってニューラルネットワークモデルの評価を行います。type , perturb , run_scheme に関しては、能動学習スキームを用いる場合は上の例のまま変更しないでください。path には、実行環境における aenet の predict.x のパスを指定してください。 base_input_dir は自由に設定して構いません。設定したディレクトリの中に predict.x に対応した入力ファイルが自動で設置されます（後述）。

ignore_species では、ニューラルネットワークモデルで「無視」する原子種を指定できます。今回の例題では、O の副格子は常に占有率 1 なので、O の配置はエネルギーに影響を及ぼしません。こういった場合は、ニューラルネットワークモデルの訓練および評価時に存在を無視した方が、計算効率が高くなります。

4.2.2 計算実行

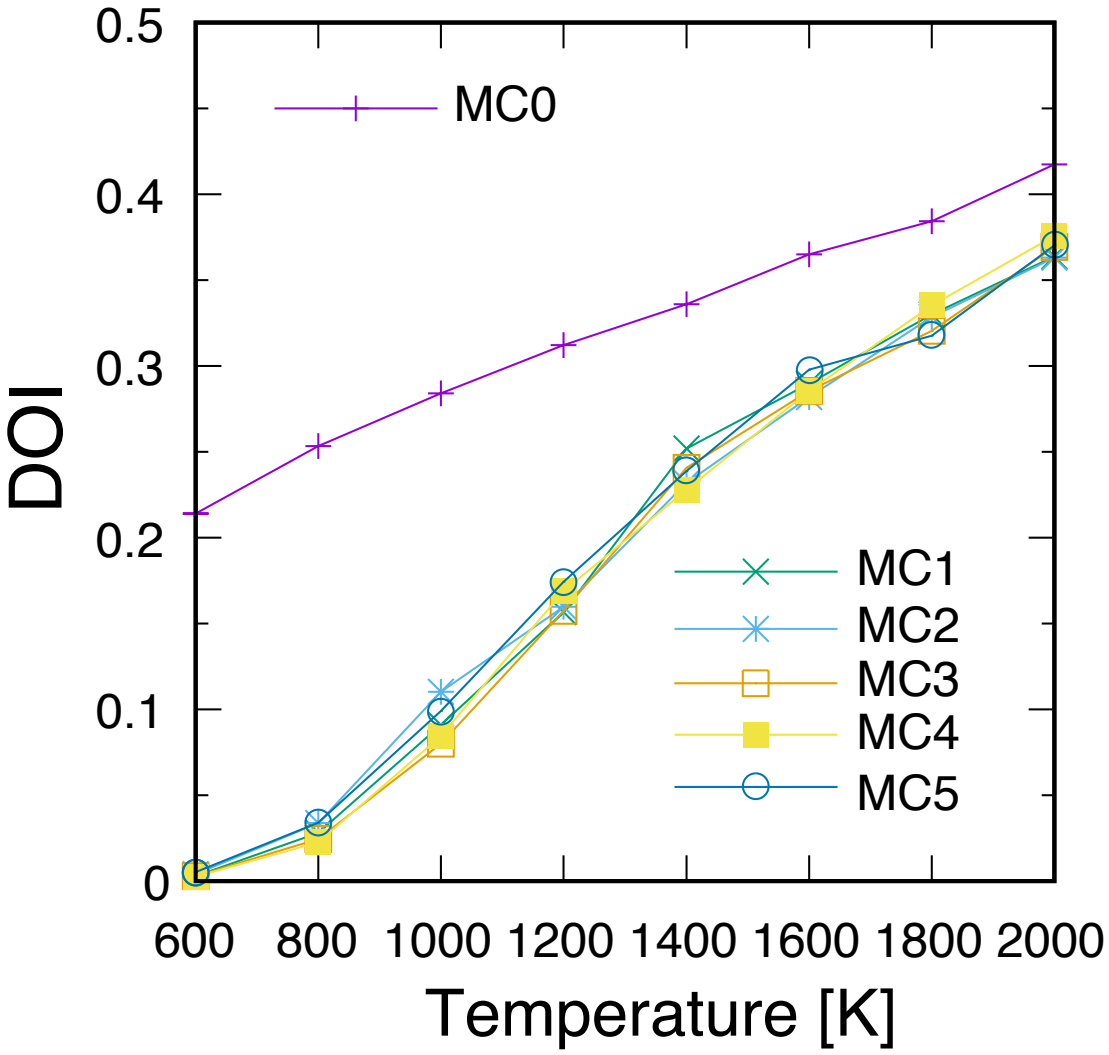
サンプルスクリプトには、計算手順を簡略化するためのスクリプト MC.sh が準備されています。MC.sh スクリプトの中身は以下の通りです。

```
#!/bin/sh
#SBATCH -p i8cpu
#SBATCH -N 1
#SBATCH -n 8
#SBATCH --time=00:30:00

srun -n 8 abics_sampling input.toml >> aenet.out

echo Done
```

abics_sampling を実行すると MCxx ディレクトリが作成されます (xx は実行回数)。abICS では、active learning 向けに、ALloop.progress ファイルから計算回数などの情報を取得する機能が追加されています。MCxx ディレクトリの下には、レプリカ数分だけのフォルダが作成され、VASP の POSCAR ファイル形式で記載された各ステップごとの原子配置 (structure.XXX.vasp)、最低エネルギーを与えた原子位置 (minE.vasp) や、各ステップごとの温度とエネルギー (obs.dat) などが出力されます。詳細については abICS マニュアルの出力ファイル を参考にしてください。

上の手続きで得られた結果は、aenet により求められたニューラルネットワークポテンシャルの精度に依存します。はじめのステップではランダムな配置をもとに学習を行ったので、低温の構造については精度が低いことが予想されます。そこで、モンテカルロで推定された構造に対して、再度第一原理計算でエネルギーを計算し再学習させます。このステップを繰り返すことで、全温度領域での精度を高めることが期待されます。

このプロセスは、AL.sh と MC.sh を順番に繰り返すことで計算できます。実際に下図に反転率 (DOI) を計算した結果を掲載します。この例では、最初の一回目の結果が MC0、その後 MC1, MC2, ..., MC5 と 5 回実行させています。最初の一回目が他のものとかなりずれていることから、精度が出ていないことが見てとれます。一度モンテカルロを行った結果を元に学習させると、その次からはほぼ同じような値が得られていることがわかります。



なお、DOI については以下の手順で計算できます。

1. MCxx ディレクトリに移動する。
2. `srun -n 8 abicsRXsepT ../input.toml` を実行して `Tseparate` ディレクトリを作成する (並列数は `abics_sampling` を実行した際の並列数に揃える。本チュートリアルでは並列数を 8 にしているので 8 に設定)。
3. sample ディレクトリにある `calc_DOI.py` と `MgAl2O4.vasp` をコピーする。
4. `srun -n 8 python3 calc_DOI.py ../input.toml` を実行して温度ごとの反転度を計算する。 (並列数の指定は 2. と同様)。

一般的には、`MCxxx/Tseparate``の中に入っている温度ごとの構造ファイルから所望の熱力学平均を計算するスクリプト（今回の場合は``calc_DOI.py`）をユーザーが用意する必要があります。

また、反転度の計算を完全に収束させるためには、例題のモンテカルロステップ数では不十分であることにご注意ください。能動学習のサイクルとは別にモンテカルロステップ数を増やした計算を行って、熱力

学平均を計算することをおすすめします。

第5章　入力ファイルフォーマット

abICS の入力ファイルは, 以下の 5 つのセクションから構成されます.

1. [sampling] セクション

   レプリカ数や温度の幅, モンテカルロステップ数など, レプリカ交換モンテカルロ部分のパラメータを指定します. また、利用するソルバーの種類 (VASP, QE, ...)、ソルバーへのパス、不変な入力ファイルのあるディレクトリなど（第一原理計算）ソルバーのパラメータを指定します.

2. [mlref] セクション

   ニューラルネットワークモデルの精度評価と訓練データの拡張などを行うため、サンプリングの結果から原子配置のみを取り出す際のオプションを設定します. `abics_mlref` のみで使用されます. また、訓練データを作成するために利用するソルバーの種類 (VASP, QE, ...)、ソルバーへのパス、不変な入力ファイルのあるディレクトリなど（第一原理計算）ソルバーのパラメータを指定します.

3. [train] セクション

   訓練データから配置エネルギー予測モデルを学習する学習器の設定を行います. `abics_train` のみで使用されます.

4. [observer] セクション

   取得する物理量の種類などを指定します.

5. [config] セクション

   合金の配位などを指定します.

以下, 順に各セクションの詳細について説明します.

5.1 [sampling] セクション

レプリカ数や温度の幅, モンテカルロステップ数など, レプリカ交換モンテカルロ部分のパラメータを指定します. 以下のようなファイルフォーマットをしています.

```
[sampling]
nreplicas = 3
nprocs_per_replica = 1
kTstart = 500.0
kTend = 1500.0
nsteps = 5
RXtrial_frequency = 2
```

(次のページに続く)

(前のページからの続き)

```
sample_frequency = 1
print_frequency = 1
```

5.1.1 入力形式

keyword = value の形式でキーワードとその値を指定します. また, #をつけることでコメントを入力することができます (それ以降の文字は無視されます).

5.1.2 キーワード

- 温度に関する指定
  - kTstart

    形式 : float 型 (>0)

    説明 : レプリカの温度の下限.
  - kTend

    形式 : float 型 (>0)

    説明 : レプリカの温度の上限.
- レプリカに関する指定
  - nprocs_per_replica

    形式 : int 型 (自然数)

    説明 : レプリカに対するプロセス数を指定します. デフォルト値 = 1.
  - nreplicas

    形式 : int 型 (自然数)

    説明 : レプリカ数を指定します.
- その他
  - nsteps

    形式 : int 型 (自然数)

    説明 : モンテカルロステップ数を指定します.
  - RXtrial_frequency

    形式 : int 型 (自然数)

    説明 : レプリカ交換を何モンテカルロステップごとに行うかを指定します. 例えば、1 に設定した場合にはモンテカルロ更新を行う度にレプリカ交換が行われ、2 に設定した場合には 2 モンテカルロステップ毎にレプリカ交換が行われます. デフォルト値 = 1

- `sample_frequency`

  形式 : int 型 (自然数)

  説明 : 物理量測定を何モンテカルロステップごとに行うかを指定します. デフォルト値 = 1

- `print_frequency`

  形式 : int 型 (自然数)

  説明 : 測定した物理量を何モンテカルロステップごとに保存するかを指定します. デフォルト値 = 1

- `reload`

  形式 : bool 型 ("true" or "false")

  説明 : 前回終了したの最後のステップから計算をやりなおすかどうかを指定します. デフォルト値 = false

5.2 [sampling.solver] セクション

ソルバーの種類 (VASP, QE, ...)、ソルバーへのパス、不変な入力ファイルのあるディレクトリなど（第一原理計算）ソルバーのパラメータを指定します. 以下のようなファイルフォーマットをしています.

```
[solver]
type = 'vasp'
path = './vasp'
base_input_dir = './baseinput'
perturb = 0.1
run_scheme = 'mpi_spawn_ready'
```

5.2.1 入力形式

`keyword = value` の形式でキーワードとその値を指定します. また, #をつけることでコメントを入力することができます (それ以降の文字は無視されます).

5.2.2 キーワード

- `type`

  形式 : str 型

  説明 : ソルバーの種類 (`OpenMX`, `QE`, `VASP`, `aenet`) を指定します.

- `path`

  形式 : str 型

  説明 : ソルバーへのパスを指定します.

- `base_input_dir`

  形式 : str 型 or str 型のリスト

  説明 : ベースとなる入力ファイルへのパスを指定します. リスト形式で複数設定した場合は、各々の入力を使った計算が順番に実行されます. 2 番目以降の計算では、前の計算の最終ステップでの構造が初期座標として用いられ, 最後の計算のエネルギーが使用されます. 例えば、1 つ目の入力ファイルで精度を犠牲にして高速な構造最適化を行い、2 番目以降の入力ファイルで高精度な設定で構造最適化を行うといったことが可能にります. あるいは、格子ベクトルの緩和を行う場合に、設定した平面波カットオフに基づいて計算メッシュをリセットするために同じ入力の計算を複数回実行するといったことも可能です.

- `perturb`

  形式 : float 型

  説明 : 対称性が良い構造を入力にしてしまうと、構造最適化が鞍点で止まってしまいがちである。これを避けるため、各原子をこのパラメータに比例するようにランダムに変位させたものを初期構造とする。0.0 あるいは false に設定することも可能. デフォルト値 = 0.0.

- `ignore_species`

  形式 : list 型

  説明 : `aenet` などのニューラルネットワークモデルで「無視」する原子種を指定します. 常に占有率が 1 のものについては、ニューラルネットワークモデルの訓練および評価時に存在を無視した方が、計算効率が高くなります.

- `run_scheme`

  形式 : str 型

  説明 : ソルバーを起動する方法を指定します. 詳細は **第一原理ソルバー利用時の注意点** を参照してください。

- `parallel_level` (QuantumESPRESSO のみ)

  形式 : 辞書型

  説明 : Parallelization levels について、各 level の並列数を指定します。長い形式のコマンドラインオプションから - を抜いたもの、すなわち、`nimage`, `npools`, `nband`, `ntg`, `ndiag` をキーとして、各 level の分割数を値とする辞書として指定します。指定した要素のみ、実際のコマンドラインオプションとして `pw.x` に渡されます。

5.3 [mlref] セクション

MC 計算の結果から原子配置のみを取り出す際のオプションを設定します. 例えばニューラルネットワークモデルの精度評価と訓練データの拡張などに利用します.

以下のようなファイルフォーマットをしています.

```
[mlref]
nreplicas = 3
ndata = 50
```

5.3.1 入力形式

keyword = value の形式でキーワードとその値を指定します. また, #をつけることでコメントを入力することができます (それ以降の文字は無視されます).

5.3.2 キーワード

- レプリカに関する指定
  - nreplicas

    形式 : int 型 (自然数)

    説明 : レプリカ数を指定します.
  - ndata

    形式 : int 型 (自然数)

    説明 : 取り出すデータ（原子配位）の数
  - sampler

    形式 : 文字列 ("linspace" or "random", デフォルトは "linspace")

    説明 : N 個生成されている MC サンプルから N_{data} のデータをどのようにして取り出すか.
    - "linspace"

      `numpy.linspace(0, N-1, num=ndata, dtype=int)` を用いて等間隔に取り出す
    - "random"

      `numpy.random.choice(range(N), size=ndata, replace=False)` を用いたランダムサンプリング

5.4 [mlref.solver] セクション

訓練データ（配置エネルギー）の計算に用いるソルバーの設定を行います。ソルバーの種類 (VASP, QE, ...)、ソルバーへのパス、不変な入力ファイルのあるディレクトリなど（第一原理計算）ソルバーのパラメータを指定します. [sampling.solver] セクションと基本的には同様で、以下のようなファイルフォーマットをしています.

```
[mlref.solver] # 参照第一原理ソルバーの設定
type = 'qe'
base_input_dir = ['./baseinput_ref', './baseinput_ref', './baseinput_ref']
perturb = 0.05
ignore_species = []
```

5.4.1 入力形式

keyword = value の形式でキーワードとその値を指定します. また, #をつけることでコメントを入力することができます (それ以降の文字は無視されます).

5.4.2 キーワード

- type

  形式 : str 型

  説明 : ソルバーの種類 (OpenMX, QE, VASP) を指定します.

- base_input_dir

  形式 : str 型 or str 型のリスト

  説明 : ベースとなる入力ファイルへのパスを指定します. リスト形式で複数設定した場合は、各々の入力を使った計算が順番に実行されます. 2 番目以降の計算では、前の計算の最終ステップでの構造が初期座標として用いられ, 最後の計算のエネルギーが使用されます. 例えば、1 つ目の入力ファイルで精度を犠牲にして高速な構造最適化を行い、2 番目以降の入力ファイルで高精度な設定で構造最適化を行うといったことが可能にります. あるいは、格子ベクトルの緩和を行う場合に、設定した平面波カットオフに基づいて計算メッシュをリセットするために同じ入力の計算を複数回実行するといったことも可能です.

- perturb

  形式 : float 型

  説明 : 対称性が良い構造を入力にしてしまうと、構造最適化が鞍点で止まってしまいがちです. これを避けるため、各原子をこのパラメータに比例するようにランダムに変位させたものを初期構造とします. 0.0 あるいは false に設定することも可能. デフォルト値 = 0.0.

- ignore_species

  形式 : list 型

  説明 : aenet などのニューラルネットワークモデルで「無視」する原子種を指定します. 常に占有率が 1 のものについては、ニューラルネットワークモデルの訓練および評価時に存在を無視した方が、計算効率が高くなります.

5.5 [train] セクション

訓練データから配置エネルギー予測モデルを学習する学習器の設定を行います。現在のところ、abICS では aenet のみに対応しています。以下のようなファイルフォーマットをしています.

```
[train] # モデル学習器の設定
type = 'aenet'
base_input_dir = './aenet_train_input'
exe_command = ['~/git/aenet/bin/generate.x-2.0.4-ifort_serial',
               'srun ~/git/aenet/bin/train.x-2.0.4-ifort_intelmpi']
ignore_species = ["O"]
```

5.5.1 入力形式

keyword = value の形式でキーワードとその値を指定します. また, #をつけることでコメントを入力することができます (それ以降の文字は無視されます).

5.5.2 キーワード

- type

  形式 : str 型

  説明 : 訓練データから配置エネルギー予測モデルを学習する学習器の設定を行います. 現在のところ、abICS では aenet のみに対応しています.

- base_input_dir

  形式 : str 型

  説明 : 設定したディレクトリの中に、学習器の設定ファイルを設置します.

- exe_command

  形式 : str の list 型

  説明 : aenet の generate.x と train.x へのパスを指定します。 train.x については MPI 並列版が利用可能で、その場合は、上の例で示すように、MPI 実行するためのコマンド（ srun 、 mpirun など）を合わせて設定してください。

- ignore_species

  形式 : list 型

  説明 : aenet などのニューラルネットワークモデルで「無視」する原子種を指定します. 常に占有率が 1 のものについては、ニューラルネットワークモデルの訓練および評価時に存在を無視した方が、計算効率が高くなります.

5.6 [observer] セクション

取得する物理量の種類などを指定します. 以下のようなファイルフォーマットをしています.

```
[observer]
type = 'default'
```

5.6.1 入力形式

keyword = value の形式でキーワードとその値を指定します. また, #をつけることでコメントを入力することができます (それ以降の文字は無視されます).

5.6.2 キーワード

- type

  形式 : str 型

  説明 : 物理量セットを指定します.

5.7 [config] セクション

合金の配位などを指定します. 以下のようなファイルフォーマットをしています.

```
[config]
unitcell = [[8.1135997772, 0.0000000000, 0.0000000000],
            [0.0000000000, 8.1135997772, 0.0000000000],
            [0.0000000000, 0.0000000000, 8.1135997772]]
supercell = [1,1,1]

[[config.base_structure]]
type = "O"
coords = [
    [0.237399980, 0.237399980, 0.237399980],
    [0.762599945, 0.762599945, 0.762599945],
    ## 中略
    [0.262599975, 0.262599975, 0.762599945],
    ]
relaxation = [
  [true, true, true],
    ## 中略
  [false, false, false],
  ]
```

(次のページに続く)

(前のページからの続き)

```
magnetization = [0.0, 0.5, ... 中略..., 0.0]

[[config.defect_structure]]
coords = [
    [0.000000000, 0.000000000, 0.000000000],
    [0.749999940, 0.249999985, 0.499999970],
    ## 中略
    [0.124999993, 0.624999940, 0.124999993],
    ]

[[config.defect_structure.groups]]
name = 'Al'
# species = ['Al']    # default
# coords = [[[0,0,0]]]  # default
# relaxation = [[true, true, true]]  # default
# magnetization = [0.0]  # default
num = 16

[[config.defect_structure.groups]]
name = 'Mg'
# species = ['Mg']    # default
# coords = [[[0,0,0]]]  # default
# relaxation = [[true, true, true]]  # default
# magnetization = [0.0]  # default
num = 8
```

5.7.1 入力形式

keyword = values の形式でキーワードとその値を指定します. また, #をつけることでコメントを入力することができます (それ以降の文字は無視されます).

5.7.2 キーワード

- 格子の指定
  - unitcell

    形式 : list 型

    説明 : 格子ベクトル $\mathbf{a}, \mathbf{b}, \mathbf{c}$ を, リスト形式で [$\mathbf{a}, \mathbf{b}, \mathbf{c}$] として指定します.
  - supercell

    形式 : list 型

    説明 : 超格子の大きさをリスト形式で [$\mathbf{a}, \mathbf{b}, \mathbf{c}$] 指定します.

- [[config.base_structure]] セクション

  type と coords によりモンテカルロ計算で動かさない原子種とその座標を指定します. 原子種が複数ある場合には, 複数の [[config.base_strucure]] セクションを指定します.

  - type

    形式 : str 型

    説明 : 原子種を指定します.

  - coords

    形式 : list の list もしくは 文字列

    説明 : 座標を指定します. 3 次元座標を表す 3 要素のリストを N 個（原子の数）だけ並べたリストか, 座標を N 行 3 列に並べた文字列で指定します.

  - relaxation

    形式 : list の list もしくは 文字列

    説明 : 原子・空間次元ごとに、構造最適化を行うかどうかを指定します. 真偽値 ("true" or "false") を表す 3 要素のリストを N 個（原子の数）だけ並べたリストか, 真偽値を N 行 3 列に並べた文字列で指定します. デフォルトはすべて "true" です。

  - magnetization

    形式 : list

    説明 : 原子ごとに磁化（up/down 電子個数差）を指定します. デフォルトはすべて 0.0 です。

- [[config.defect_structure]] セクション

  モンテカルロで更新する原子が入る座標 (coords) と入りうる原子（団） (group) を指定します. Ver. 1.0 では POSCAR や cif からの変換ツールが利用出来るようになる予定です.

  - coords

    形式 : list の list もしくは 文字列

    説明 : 原子が入る座標を指定します. 3 次元座標を表す 3 要素のリストを N 個（原子の数）だけ並べたリストか, 座標を N 行 3 列に並べた文字列で指定します.

  - [[config.defect_structure.groups]] セクション

    モンテカルロで更新する原子グループの情報を指定します.

    * name

      形式 : str 型

      説明 : 原子グループの名前を指定します.

    * species

      形式 : list 型

説明 : 原子グループに属する原子種を指定します. デフォルト値は name で指定したものがひとつだけ含まれたリストです. また, 空のリストを用いて欠陥を表現できます.

* coords

形式 : list の list の list

説明 : 原子グループ中の各原子の座標を、局所回転の向きごとに指定します. 3 次元座標を表す 3 要素のリストを N 個（原子の数）だけ並べたリストをさらに向きごとにならべた、3 重のリストとして指定します。たとえば原子の数が 2 つあり、回転の方向として x,y,z 向きの 3 種類考える場合には、

```
coords = [
[ # dir-1
[0.0, 0.0, 0.0], [0.5, 0.0, 0.0]
],
[ # dir-2
[0.0, 0.0, 0.0], [0.0, 0.5, 0.0]
],
[ # dir-3
[0.0, 0.0, 0.0], [0.0, 0.0, 0.5]
],
]
```

のように指定します。デフォルト値は *[[[0.0, 0.0, 0.0]]]* なので、原子が一つしかない場合には一般に省略可能です。

* relaxation

形式 : list の list もしくは 文字列

説明 : 原子・空間次元ごとに、構造最適化を行うかどうかを指定します. 真偽値 ("true" or "false") を表す 3 要素のリストを N 個（原子の数）だけ並べたリストか, 真偽値を N 行 3 列に並べた文字列で指定します. デフォルトはすべて "true" です。

* magnetization

形式 : list

説明 : 原子ごとに磁化（up/down 電子個数差）を指定します. デフォルトはすべて 0.0 です。

* num

形式 : int 型

説明 : この原子グループの数を指定します.

第6章　出力ファイルフォーマット

各レプリカディレクトリ以下に計算結果が出力されます。

6.1 structure.XXX.vasp

各ステップごとの原子配置が VASP の POSCAR ファイル形式で出力されます。ステップ番号がファイル名の XXX に入ります。

例:

```
Mg8 Al16 O32
1.0
8.113600 0.000000 0.000000
0.000000 8.113600 0.000000
0.000000 0.000000 8.113600
Al Mg O
16 8 32
direct
0.011208 0.995214 0.998158 Al
0.758187 0.240787 0.499981 Al
... skipped ...
0.746308 0.744706 0.233021 O
0.257199 0.255424 0.771040 O
```

6.2 minE.vasp

最低エネルギーを与えた原子位置が VASP の POSCAR ファイル形式で出力されます。

6.3 obs.dat

各ステップごとの温度とエネルギーが電子ボルト単位で出力されます。

例:

```
0       0.1034076       -41690.28269769395
1       0.1034076       -41692.06763035158
2       0.1034076       -41692.06763035158
3       0.1034076       -41691.98205990787
4       0.1034076       -41692.74143710456
```

6.4 obs_save.npy

各ステップごとのエネルギーが電子ボルト単位で出力されます。numpy.load('obs_save.npy') で、darray として読み取ることができます。

例:

```
$ python -c "import numpy; print(numpy.load('obs_save.npy'))"
[[-41690.28269769]
 [-41692.06763035]
 [-41692.06763035]
 [-41691.98205991]
 [-41692.7414371 ]]
```

6.5 kT_hist.npy

各ステップごとの温度 (電子ボルト単位) が Numpy バイナリ形式で出力されます。numpy.load('kT_hist.npy') で、 darray として読み取ることができます。

例:

```
$ python -c "import numpy; print(numpy.load('kT_hist.npy'))"
[0.1034076 0.1034076 0.1034076 0.1034076 0.1034076]
```

6.6 Trank_hist.npy

各ステップごとの温度インデックスが Numpy バイナリ形式で出力されます。numpy.load('Trank_hist.npy') で、 darray として読み取ることができます。

例:

```
$ python -c "import numpy; print(numpy.load('Trank_hist.npy'))"
[1 1 1 1 1]
```

第7章　付属ツール

abICS には、典型的なワークフローを容易にするためのツールがいくつか付属しています。

- 原子構造ファイルから abICS 入力ファイルを作成するための `st2abics`
- レプリカ交換モンテカルロ計算実行の後処理を行うための `abicsRXsepT`

以下のセクションでは、これらのツールのそれぞれの使用方法を説明します。

7.1 st2abics

abICS **入力ファイル** の *[config]* **セクション** を比較的簡単に作成するために、`st2abics` ツールを使うことができます。これは pymatgen で読める原子構造ファイルを読み取り、`[config]` セクションを埋めた abICS 入力テンプレートファイルに変換します。元の構造ファイルからどのようにして `config.base_structure` と `config.defect_structure` を構成するか指定するため、 `st2abics` 専用の制御ファイルが必要です。`st2abics` は以下のように使用します:

```
$ st2abics -h
usage: st2abics [-h] inputfi structurefi [outfi]

Prepare abICS config from structure file

positional arguments:
  inputfi      toml input file for st2abics
  structurefi  Structure file that can be read by pymatgen Structure.from_file()␣
↪method
  outfi        Output file to be used as abics input. Defaults to standard output

optional arguments:
  -h, --help   show this help message and exit
```

例がいくつか `examples/st2abics` に用意されています:

```
$ cd examples/st2abics
$ st2abics st2abics_MgAl2O4.toml MgAl2O4.vasp abics_MgAl2O4.toml # spinel
$ st2abics st2abics_CuZn.toml CuZn.vasp abics_CuZn.toml # brass
$ st2abics st2abics_BZY.toml BaZrO3.vasp abics_BZY.toml # Y-doped BaZrO3
```

結果として得られたファイル (上記の例では `abics_MgAl2O4.toml`, `abics_CuZn.toml`, `abics_BZY.toml`) は、`[sampling]`, `[mlref]`, `[train]`, `[observer]` セクションのキーワードが空になっており、これらを記

入した後、abICS の入力として使用することができます。

7.1.1 入力フォーマット

st2abics の入力ファイルの例は、 examples/st2abics にあります (上の例では st2abics_CuZn.toml , st2abics_MgAl2O4.toml, st2abics_BZY.toml)。

フォーマットは abICS 入力ファイルの [config] セクションに似ています。

7.1.2 キーワード

- supercell

  形式 : list 型

  説明 : スーパーセルの大きさをリスト形式 [$\mathbf{a}, \mathbf{b}, \mathbf{c}$] で指定します.

- [[config.base_structure]] セクション

  ここでは、モンテカルロ計算中に格子サイト間で原子を交換しない base_structure を指定します。

  - species

    形式 : 文字列の list

    説明 : base_structure の原子種。対応する座標は入力構造ファイルから自動的に抽出されます。

  - fix

    形式 : bool 型 ("true" or "false")

    説明 : base_structure の局所的な緩和を行わない場合 true、行う場合は false に設定する。

- [[config.defect_structure]] セクション

  このセクションは、配置サンプリングを行う副格子を指定します。複数の [[config.defect_structure]] セクションが存在しても構いません。例えば、陽イオンの配置サンプリングを行う副格子と、陰イオンの配置サンプリングを行う副格子を指定することができます。

  - site_center_species

    形式 : 文字列の list

    説明 : 元の構造ファイルに含まれる元素種のうち、配置サンプリングを行う格子サイトと対応するもの。

  - [[config.defect_structure.groups]] サブセクション

    このセクションでは、配置サンプリングを行う格子サイト上に配置される原子グループを指定します。もしこのセクションがない場合は、 site_center_species を基に、入力された構造ファイルから自動的に構築されます。

* name

  形式 : 文字列

  説明 : 原子グループの名前。

* species

  形式 : 文字列の list

  説明 : 原子グループに属する元素種のリスト。デフォルト値は、[name] です。元の構造ファイルに含まれない元素種も指定できます。さらに、空のリスト [] で格子欠陥を表現できます。例として、サンプルにある st2abics_BZY.toml を参照してください。

* coords

  形式 : list の list の list あるいは文字列

  説明 : 原子グループがとることのできる配向ごとの原子グループ内の各原子の座標（入力ファイルの coords 定義 **参照** ）。デフォルト値は [[[0.0, 0.0, 0.0]] です。

* num

  形式 : int

  説明 : このセクションで指定した原子グループの数。スーパーセル内のサイト数に応じた数を指定してください。

7.2 abicsRXsepT

このツールは、RXMC 実行の各サンプリングステップで得られる構造とエネルギーを温度ごとに並べ替えるためのツールです。abICS の RXMC 実行が終了した後に使用します:

```
$ mpiexec -np NPROCS abicsRXsepT input.toml NSKIP
```

NPROCS はレプリカの数と同じかそれ以上でなければならず、input.toml は abICS の実行に使用された abICS 入力ファイルに置き換える必要があります。NSKIP はオプションのパラメータで、各温度でのエネルギー平均を計算する際にスキップする初期ステップ数を指定します（デフォルト値は 0）。結果は Tseparate ディレクトリに保存され、温度ごとのエネルギー平均は Tseparate/energies_T.dat に保存されます。

第8章　アルゴリズム

abICS は、拡張アンサンブル法と任意のエネルギー計算ソフトウェアを組み合わせるために設計されています。現在は、レプリカ交換モンテカルロ法のみが実装されています。

8.1 レプリカ交換モンテカルロ法

広く使用されているメトロポリスモンテカルロ法は、局所安定配置に捕まりやすく、そこでサンプリングが行き詰まる傾向があります。レプリカ交換法は、計算対象の系のコピー（レプリカ）を使って、この問題を克服することを目的としています。レプリカ交換法は、大まかに次のように説明できます（より正確な説明については、下記の文献を参照してください)。まず、各レプリカに対して異なる温度におけるモンテカルロサンプリングを個別に実行します。事前に設定された任意の間隔で、メトロポリス基準に従ってレプリカ間で温度を交換し、サンプリングを再開します。メトロポリス基準によって、大まかには、エネルギーが低いレプリカに低い温度が割り当てられることになります。これにより、高い温度のレプリカによる大域的なサンプリングと低い温度における局所安定配置の探索を共存させることができます。

abICS では、入力ファイルの `[replica]` セクションでレプリカ交換モンテカルロ法に関連したパラメータを指定します。レプリカの温度の下限を `kTstart` 、 上限を `kTend` とし、レプリカ数を `nreplicas` とすることで、

$$T_i = \frac{\mathbf{kTend} - \mathbf{kTstart}}{\mathbf{nreplicas} - \mathbf{1}} i + \mathbf{kTstart}$$

として `nreplicas` 個の異なる温度に接触しているレプリカ系が用意されます (ただし、$i = 0 \cdots$ `nreplicas`-1 。abICS では、「nprocs_per_replica」を使用して、各レプリカで計算を実行する並列ソルバープロセスの数を指定できます)。モンテカルロステップ数は `nsteps` で指定し、 `RXtrial_frequency` ステップ毎に交換遷移確率

$$R = \exp\left[-\left(\frac{1}{T_i} - \frac{1}{T_k}\right)\left(E(X_i) - E(X_k)\right)\right]$$

が計算されます。R の確率で温度交換 $T_i \leftrightarrow T_k$ が行われます (ただし、 X_i は i 番目のレプリカ系の状態です。また、abICS では隣接する温度を持ったレプリカ同士で交換試行をしています)。なお、物理量は `sample_frequency` ステップ毎に測定が行われます。

- abICS の概要について
  - S. Kasamatsu and O. Sugino, J. Phys. Condens. Matter, 31, 085901 (2019).
- レプリカ交換モンテカルロ法について
  - K. Hukushima and K. Nemoto, J. Phys. Soc. Japan, 65, 1604 (1996).
  - R. Swendsen and J. Wang, Phys. Rev. Lett. 57, 2607 (1986).

8.2 配置と更新について

ここでは、 図 8.1 を例に ablCS での格子配置の定義とモンテカルロ法での更新の概要を説明します。(a)-(c) は `unitcell` 、 `base_structure` 、 `defect_structure` の概念図で、青丸、緑丸、黒丸はそれぞれ `base_structure` で定義される原子種、星印は `defect_structure` で定義される欠陥が入る位置を表します。(d) は `base_structure` で原子種を指定する場合の概念図です。ここでは、blue, green, black の 3 原子種を定義しています。各原子がどのように配置されるかは、各原子種ごとに `coords` で定義します。(e) は `defect_structure` で欠陥位置に入るグループを指定する場合の概念図です。orange は 2 種の原子種から構成される 4 原子で構成されるグループを定義し、 purple は 3 種の原子種から構成される 3 原子のグループを構成しています。 `defect_structure.coords` で指定される欠陥位置にこれらのグループが配置されます。 各グループ内での原子の配置は `defect_structure.groups` セクション内の `coords` で指定することができます。`defect_structure` を複数定義した場合は、それぞれの欠陥座標にそれぞれの原子グループが配置されます。(f) はモンテカルロ法のアップデートに関する概念図です。 アップデートでは欠陥の位置に入るグループを入れ替えるパターンと、配置は変えずに配位を変える 2 つのパターンがあります (どちらのアップデートを行うかは半分の確率で自動で選択されます)。提案された配置 X_{trial} から指定されたソルバーでエネルギーを計算し、採択率 $P(X_i \to X_{trial})$ を計算します。

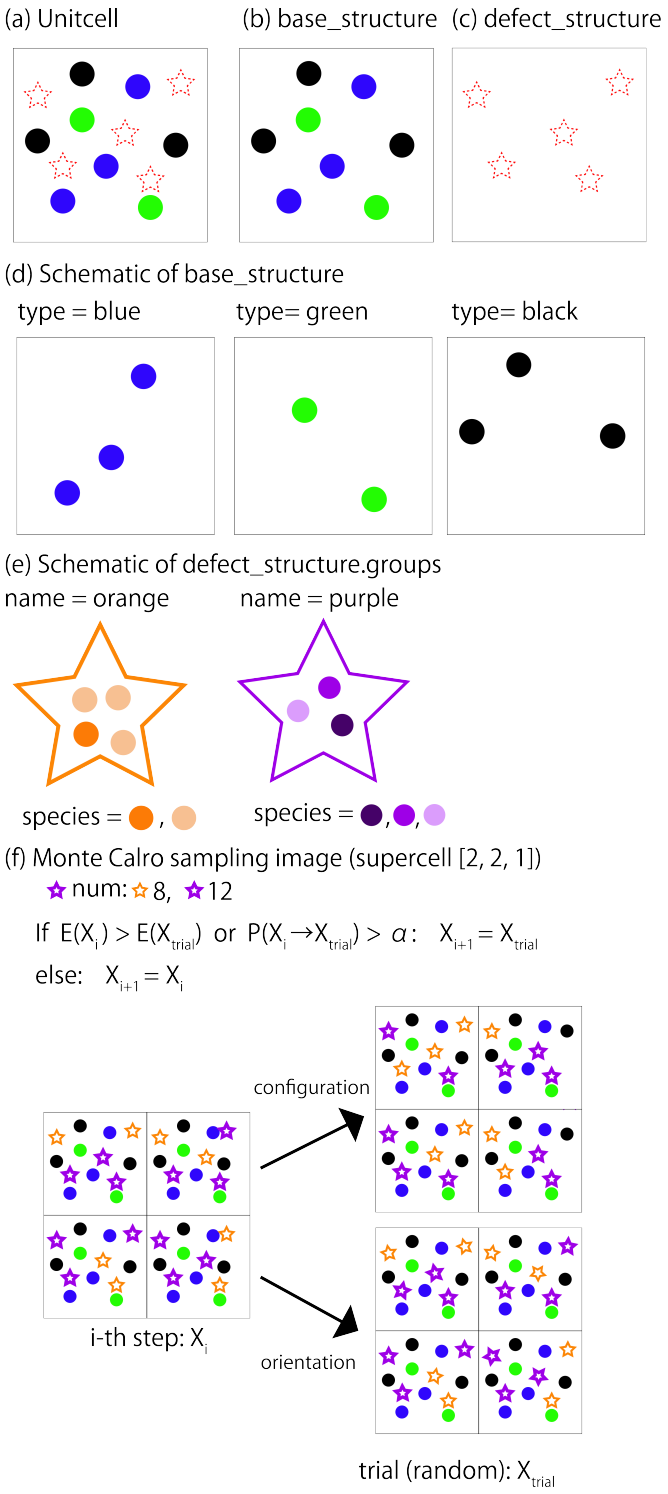


図 8.1: (a)-(e)abICS での格子の定義と (f) モンテカルロ法の概要。詳細は本文に記載。

第9章　謝辞

このソフトウェアの開発は, 様々なプロジェクトとコンピューター資源の提供によりサポートされてきました。この場を借りて感謝します。

- ポスト「京」重点課題 5
- 東京大学物性研究所スーパーコンピュータ共同利用
- 文部科学省卓越研究員事業
- 科学研究費補助金（No. JP18H05519, No. 19K15287）
- JST CREST (No. JPMJCR15Q3, No. 19K15287)
- NEDO

また, abICS 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 (2019,2022 年度) の支援を受け開発されました。この場を借りて感謝します。

第10章　お問い合わせ

abICS に関するお問い合わせはこちらにお寄せください。

- バグ報告

  abICS のバグ関連の報告は GitHub の Issues で受け付けています。

  バグを早期に解決するため、報告時には次のガイドラインに従ってください。

  - 使用している abICS のバージョンを指定してください。
  - インストールに問題がある場合には、使用しているオペレーティングシステムとコンパイラの情報についてお知らせください.
  - 実行に問題が生じた場合は, 実行に使用した入力ファイルとその出力を記載してください。

- その他

  研究に関連するトピックなど GitHub の Issues で相談しづらいことを問い合わせる際には, 以下の連絡先にコンタクトをしてください。

  E-mail: `abics-dev__at__issp.u-tokyo.ac.jp` (_at_を@に変更してください)