ablCS Documentation

Release 0.1

ablCS's team

Dec 09, 2019

CONTENTS:

CHAPTER

ONE

ABOUT ABICS

1.1 What is abICS ?

abICS is a software framework for performing configurational sampling in disordered systems, with a specific emphasis on multi-component solid state systems such as metal and oxide alloys. It couples parallel sampling methods with external codes that perform structural relaxation and energy calculations. We provide interfaces for Quantum Espresso, VASP, and aenet, with planned support for OpenMX in the near future.

1.2 Developers

abICS is developed by the following members.

- **ver. 0.1**
  - Shusuke Kasamatsu (Yamagata University)
  - Yuichi Motoyama (Institute for Solid State Physics, Univ. of Tokyo)
  - Kazuyoshi Yoshimi (Institute for Solid State Physics, Univ. of Tokyo)
  - Yoshiyuki Yamamoto (Institute for Solid State Physics, Univ. of Tokyo)
  - Osamu Sugino (Institute for Solid State Physics, Univ. of Tokyo)
  - Taisuke Ozaki (Institute for Solid State Physics, Univ. of Tokyo)

1.3 Version information

- ver. 0.1 : 2019/12/10.

1.4 License

This package is distributed under GNU General Public License version 3 (GPL v3) or later.

1.5 Copyright

This software was developed with the support of "*Project for advancement of software usability in materials science*" of The Institute for Solid State Physics, The University of Tokyo.

CHAPTER

TWO

INSTALL

2.1 Download

The source codes of abICS can be obtained from GitHub page .

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

2.2 Prerequisites

- python3
- numpy
- scipy
- toml (for parsing input files)
- mpi4py (for parallel tempering)
- pymatgen (for parsing vasp I/O)
- qe-tools (for parsing QE I/O)

2.3 Directory structure

The directory structure of abICS is given as follows:

```
- examples
   - standard
       - spinel
           - QE
           - vasp
           - openmx
       - sub-lattice
           - QE
           - vasp
           - openmx
   - expert
       - ising2D
       - 2D_hardcore
       ...
- make_wheel.sh
- py_mc
```

```
    - pymc.py
    - python module
- test
```

`examples/standard` contains samples that can be run by simple files. `examples/expert` contains examples by using python module.

A set of python modules are located in the `py_mc` directory.

2.4 Install

1. Make wheel file by typing following command:

```
$ ./make_wheel.sh
```

2. Install using the created file as follows:

```
$ pip install dist/py_mc-\*.whl
```

If you want to change the install directry, use `--user` option or `--prefix = DIRECTORY` (`DIRECTORY` is the path to the directory where you want to install) option. In the following, the case for using `--user` option is shown:

```
$ pip install --user dist/py_mc-\*.whl
```

CHAPTER

THREE

USAGE

3.1 Overview

3.1.1 Preparing the input file

3.1.2 Execution

The number of processes specified here must be the same as the number of replicas.

(Various matters derived from MPI_Comm_spawn (the actual process to be allocated, mpiexec options, etc. need to be described somewhere)

```
$ mpiexec -np 2 abics input.toml
```

CHAPTER

FOUR

FILE FORMAT

The input file of abICS is constructed by the following four sections:

1. [replica] section Specify the parameters of the replica exchange Monte Carlo part, such as the number of replicas, the temperature range, and the number of Monte Carlo steps.
2. [solver] section Specify the parameters for the (first principle calculation) solver, including the type of solver (VASP, QE,...), the path to the solver, and the directory containing immutable input files.
3. [observer] section Specify the type of physical quantity to be calculated.
4. [config] section Specify the configuration of the alloy, etc.

The following sections describe the detail of each section.

4.1 [replica] section

Specify the parameters of the replica exchange Monte Carlo part, such as the number of replicas, the temperature range, and the number of Monte Carlo steps. The example is shown as follows.

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

4.1.1 Input Format

Specify a keyword and its value in the form `keyword = value`. Comments can also be entered by adding # (Subsequent characters are ignored).

4.1.2 Keywords

- About temperatures
  - `kTstart`

  Format : float (>0)

Description : Initial temperature.

- `kTend`

Format : float (>0)

Description : Final temperature.

- `nsteps`

Format : int (nature number)

Description : Number of temperature divisions.

- About replica

  - `nprocs_per_replica`

  Format : int (nature number)

  Description : The number of processes for the replica. Default value = 1.

  - `nreplicas`

  Format : int

  Description : The number of replicas.

- Others

  - `RXtrial_frequency`

  Format : int (nature number)

  Description : The number of Monte Carlo steps for replica exchange. Default = 1.

  - `sample_frequency`

  Format : int (nature number)

  Description : The number of Monte Carlo steps for observation of physical quantities. Default value = 1.

  - `print_frequency`

  Format : int (nature number)

  Description : The number of Monte Carlo steps for saving physical quantities. Default value = 1.

4.2 [solver] section

Specify solver parameters (first-principles calculation) such as solver type (VASP, QE, ...), path to solver, directory with invariant input file. The example is shown as follows:

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

4.2.1 Input Format

Specify a keyword and its value in the form `keyword = value`. Comments can also be entered by adding # (Subsequent characters are ignored).

4.2.2 Keywords

- `type`

Format : str

Description : Specify the solver type (`OpenMX, QE, VASP`).

- `path`

Format : str

Description : Specify the path to the solver.

- `base_input_dir`

Format : str

Description : Specify the path to the base input file.

- `perturb`

Format : float

Description : If a structure with good symmetry is input, structure optimization tends to stop at the saddle point. In order to avoid this, an initial structure is formed by randomly displacing each atom in proportion to this parameter. It can also be set to 0.0 or false. Default value = 0.0.

4.3 [observer] section

Specify the type of physical quantity to be acquired. The example is shown as follows:

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

4.3.1 Input Format

Specify a keyword and its value in the form `keyword = value`. Comments can also be entered by adding # (Subsequent characters are ignored).

4.3.2 Key words

- `type`

Format : str

Description : Specify a physical quantity set.

  - "default"
    - Default setting. Get the energy and the atom group in each coordinate.

4.4 [config] section

Specify alloy coordination, etc. The example is shown as follows:

```
[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],
    ...
    [0.262599975, 0.262599975, 0.762599945],
    ]

[[config.defect_structure]]
coords = [
    [0.000000000, 0.000000000, 0.000000000],
    [0.749999940, 0.249999985, 0.499999970],
    ...
    [0.124999993, 0.624999940, 0.124999993],
    ]
[[config.defect_structure.groups]]
name = 'Al'
# species = ['Al']    # default
# coords = [[[0,0,0]]]  # default
num = 16
[[config.defect_structure.groups]]
name = 'Mg'
# species = ['Mg']    # default
# coords = [[[0,0,0]]]  # default
num = 8
```

4.4.1 Input Format

Specify a keyword and its value in the form `keyword = value`. Comments can also be entered by adding # (Subsequent characters are ignored).

4.4.2 Key words

- Specify lattice

  – `unitcell`

  Format : list

  Description : Specify lattice vector $\mathbf{a}, \mathbf{b}, \mathbf{c}$ by the list format [$\mathbf{a}, \mathbf{b}, \mathbf{c}$] .

  – `supercell`

  Format : list

  Description : Specify the size of super lattice by the list format[$\mathbf{a}, \mathbf{b}, \mathbf{c}$].

- [[config.base_structure]] section

  `type` and `coords` specify the atomic species that do not move in Monte Carlo calculation and their coordinates.
  If there are multiple atomic species, specify multiple [[config.base_strucure]] sections.

  - `type`

  Format : str

  Description : Specify atomic specie.

  - `coords`

  Format : list of lists or str

  Description : Specify coordinates. Specify a list of N elements (number of atoms) arranged in 3 elements representing 3D coordinates, or a string of coordinates arranged in N rows and 3 columns.

- [[config.defect_structure]] section

  Specify the coordinates (coords) and atoms (group) that can enter the atoms to be updated in Monte Carlo. In Ver. 1.0, conversion tools from POSCAR and cif will be available.

  - `coords`

  Format : list of lists or str

  Description : Specify the coordinates where the atom enter. Specify a list of N elements (number of atoms) arranged in 3 elements representing 3D coordinates, or a string of coordinates arranged in N rows and 3 columns.

  - [[config.defect_structure.groups]] section

    Specify the atom group information to be updated by Monte Carlo.

    * `name`

      Format : str

      Description : Specify the name of atomic group.

    * `species`

      Format : list

      Description : Specify the atomic species belonging to the atom group. The default value is a list containing only one specified by `name`.

    * `coords`

      Format : list of lists or str

      Description : Specify the coordinates of each atom in the atom group. Specify a list of N elements (number of atoms) arranged in 3 elements representing 3D coordinates, or a string of coordinates arranged in N rows and 3 columns. Default value is *[[0.0, 0.0, 0.0]]*.

    * `num`

      Format : int

      Description : Specify the number of this atom group.

CHAPTER

FIVE

ALGORITHM

T.B.A.

CHAPTER

SIX

ACKNOWLEDGEMENT

The development of this software has been supported by various funding bodies and computer resource providers over the years.

- Priority Issue on Post-K computer (Development of new fundamental technologies for high-efficiency energy creation, conversion/storage and use)
- Joint-use Supercomputer System at Institute for Solid State Physics, the University of Tokyo
- Leading Initiative for Excellent Young Researchers (LEADER) by Ministry of Education, Culture, Science, and Technology (MEXT) of Japan.
- KAKENHI (No. JP18H05519, No. 19K15287) by MEXT
- Core Research for Evolutional Science and Technology (CREST) by Japan Science and Technology Agency (No. JPMJCR15Q3, No. 19K15287)
- New Energy and Industrial Technology Development Organization

We would also like to express our thanks for the support of the "*Project for advancement of software usability in materials science*" of The Institute for Solid State Physics, The University of Tokyo, for the development of abICS.

CHAPTER

SEVEN

CONTACTS

- About Bugs

  Please report all problems and bugs on the github Issues page

  To resolve bugs early, follow these guidelines when reporting:

  - Please specify the version of abICS you are using.
  - If there are problems for installation, please inform us about your operating system and the compiler, and include the input / output.
  - If a problem occurs during execution, enter the input file used for execution and its output.

  Thank you for your cooperation.

- Others

  If you have any questions about topics related to your research that are difficult to consult at Issues on GitHub, please contact the following contact information.

  E-mail: `abics-dev__at__issp.u-tokyo.ac.jp` (replace _at_ by @)

CHAPTER

EIGHT

INDICES AND TABLES

- genindex
- modindex
- search