Cif2x Documentation

Release 1.1.0

ISSP, University of Tokyo

Sep 10, 2024

Contents

Chapter 1

Introduction

1.1 What is cif2x?

In recent years, the use of machine learning for predicting material properties and designing substances (known as materials informatics) has gained considerable attention. The accuracy of machine learning depends heavily on the preparation of appropriate training data. Therefore, the development of tools and environments for the rapid generation of training data is expected to contribute significantly to the advancement of research in materials informatics.

Cif2x is a tool that generates input files for first-principles calculations from cif files. It constructs parts that vary depending on the type of material and computational conditions from crystal structure data, using input parameters as a template. It is capable of generating multiple input files tailored to specific computational conditions. Currently, it supports VASP, Quantum ESPRESSO, OpenMX, and AkaiKKR.

1.2 License

The distribution of the program package and the source codes for cif2x follow GNU General Public License version 3 (GPL v3) or later.

1.3 Contributors

This software was developed by the following contributors.

- Developers
  - Kazuyoshi Yoshimi (The Instutite for Solid State Physics, The University of Tokyo)
  - Tatsumi Aoyama (The Instutite for Solid State Physics, The University of Tokyo)
  - Yuichi Motoyama (The Instutite for Solid State Physics, The University of Tokyo)
  - Masahiro Fukuda (The Instutite for Solid State Physics, The University of Tokyo)
  - Kota Ido (The Instutite for Solid State Physics, The University of Tokyo)
  - Tetsuya Fukushima (The National Institute of Advanced Industrial Science and Technology (AIST))
  - Shusuke Kasamatsu (Yamagata University)
  - Takashi Koretsune (Tohoku University)

- Project Corrdinator
  - Taisuke Ozaki (The Instutite for Solid State Physics, The University of Tokyo)

1.4 Release history

ver.1.0.1
Released on 2024/03/31

ver.1.0.0
Released on 2024/03/19

ver.1.0-alpha
Released on 2023/12/28

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.

1.6 Operating environment

This tool was tested on the following platforms:

- Ubuntu Linux + python3

Chapter 2

Installation and basic usage

Prerequisite

Input file generator for first-principles calculation `cif2x` included in HTP-tools requires the following programs and libraries:

- python 3.x
- pymatgen module
- ruamel.yaml module
- f90nml module
- qe-tools module
- numpy module
- pandas module
- monty module
- OpenBabel module (optional)
- AkaiKKRPythonUtil module

For A tool to retrieve crystallographic data from databases `getcif`, the additional library is required:

- mp-api module

Official pages

- GitHub repository

Downloads

cif2x can be downloaded by the following command with git:

```
$ git clone https://github.com/issp-center-dev/cif2x.git
```

Installation

Once the source files are obtained, you can install cif2x by running the following command. The required libraries will also be installed automatically at the same time.

```
$ cd ./cif2x
$ python3 -m pip install .
```

The executable files `cif2x` and `getcif` will be installed. You may need to add `--user` option next to `install` keyword above in case you are not allowed to install packages system-wide.

AkaiKKRPythonUtil module need to be installed separately. The source package is available from the repository. Then follow the steps below to install the module along with the required seaborn module:

```
$ git clone https://github.com/AkaiKKRteam/AkaiKKRPythonUtil.git
$ cd ./AkaiKKRPythonUtil/library/PyAkaiKKR
$ python3 -m pip install .
$ python3 -m pip install seaborn
```

Directory structure

```
.
|-- LICENSE
|-- README.md
|-- pyproject.toml
|-- docs/
|   |-- ja/
|   |-- en/
|   |-- tutorial/
|-- src/
|   |-- cif2x/
|       |-- __init__.py
|       |-- main.py
|       |-- cif2struct.py
|       |-- struct2qe.py
|       |-- qe/
|       |   |-- __init__.py
|       |   |-- calc_mode.py
|       |   |-- cards.py
|       |   |-- content.py
|       |   |-- qeutils.py
|       |   |-- tools.py
|       |-- struct2vasp.py
|       |-- struct2openmx.py
|       |-- openmx/
|       |   |-- __init__.py
|       |   |-- vps_table.py
|       |-- struct2akaikkr.py
|       |-- akaikkr/
|       |   |-- make_input.py
|       |   |-- read_input.py
|       |   |-- run_cif2kkr.py
|       |-- utils.py
|   |-- getcif/
|       |-- __init__.py
|       |-- main.py
|-- sample/
```

Basic usage

`cif2x` is a tool to generate a set of input files for first-principles calculation software. It takes an input parameter file as a template, and generates parameter items that may vary by materials and calculation conditions from crystallographic data. In the present version, `cif2x` supports Quantum ESPRESSO, VASP, OpenMX, and AkaiKKR.

1. Prepare input parameter file

   First, you need to create an input parameter file in YAML format that describes contents of the input file to be generated for the first-principles calculation software.

2. Prepare crystal structure files and pseudo-potential files

   The crystal structure data need to be prepared for the target materials. The file format is CIF, POSCAR, xfs, or those supported by pymatgen.

   For Quantum ESPRESSO, the pseudo-potential files and the index file in CSV format need to be placed. Their locations are specified in the input parameter file.

   For VASP, the location of the pseudo-potential files will be specified in a file `~/.config/.pmgrc.yaml` or by an environment variable. It may be specified in the input parameter file.

3. Run command

   Run `cif2x` command with the input parameter file and the crystal structure data as arguments. To generate input files for Quantum ESPRESSO, the target option `-t QE` should be specified. The option turns to `-t VASP` for VASP, `-t OpenMX` for OpenMX, and `-t AkaiKKR` for AkaiKKR.

   ```
   $ cif2x -t QE input.yaml material.cif
   ```

Chapter 3

Tutorial

The procedure to use the input file generator `cif2x` for first-principles calculation software consists of preparing an input parameter file, crystal structure data, and pseudo-potential files, and running the program `cif2x`. In the current version, the supported software includes Quantum ESPRESSO, VASP, OpenMX, and AkaiKKR. In this tutorial, we will explain the steps along a sample for Quantum ESPRESSO in `docs/tutorial/cif2x`.

3.1 Prepare an input parameter file

An input parameter file describes the content of input files for the first-principles calculation software. An example is presented below. It is a text file in YAML format that contains options to crystal structure data, and contents of the input file used as an input for the first-principle calculation. See *file format* section for the details of specification.

In YAML format, parameters are given in dictionary form as `keyword:  value`, where `value` is a scalar such as a number or a string, or a set of values enclosed in `[ ]` or listed in itemized form, or a nested dictionary.

```
structure:
  use_ibrav: false
  tolerance: 0.05

optional:
  pseudo_dir: ./pseudo
  pp_file: ./pseudo/pp_psl_pbe_rrkjus.csv

tasks:
  - mode: scf
    output_file: scf.in
    output_dir: scf
    template: scf.in_tmpl
    content:
      namelist:
        control:
          prefix: pwscf
          pseudo_dir:
          outdir: ./work
        system:
          ecutwfc:
```

(continues on next page)

(continued from previous page)

```
        ecutrho:
      CELL_PARAMETERS:
      ATOMIC_SPECIES:
      ATOMIC_POSITIONS:
        option: crystal
      K_POINTS:
        option: automatic
        grid: [8,8,8]
```

The input parameter file consists of `structure`, `optional`, and `tasks` sections. The `structure` section specifies options to the crystal structure data. The `optional` section holds global settings concerning the pseudo-potentials.

The `tasks` section describes inputs for the first-principles calculations. In case of generating multiple files for a series of calculations, the `tasks` section takes a list of parameter sets. For each set, the calculation type is specified by the `mode` parameter: `scf` and `nscf` are supported as modes, as well as arbitrary modes for generic output.

The content of the output is given in `content` section. The input files of Quantum ESPRESSO are composed of the parts in namelist format of Fortran90 starting from `&keyword`, and the blocks called cards that start with keywords such as `K_POINTS` and end with blank lines. The `content` block holds namelist and cards in a form of nested dictionary. Basically, the specified items are written to the input files as-is, except for several cases. If a keyword is left blank, its value will be obtained form the crystal structure data or other sources.

Besides, templates of the input files can be used. The content of the file given by the `template` keyword is considered as input data along with the entries in `content` block. When the entries of the same keywords appear both, those of the input parameter files will be used. Therefore, it is possible to use template files and overwrite some entries by the input parameter file as needed. In the present example, the file (`scf.in_tmpl`) shown below is read as a template, and the entries on cutoff parameters as well as cards of CELL_PARAMETER, ATOMIC_SPECIES, ATOMIC_POSITIONS, K_POINTS are generated from the crystal structure data and pseudo-potential files. It is noted that the values of `ecutwfc` and `ecutrho` are overwritten by the empty lines.

```
&control
    calculation = 'scf'
    prefix = 'pwscf'
    pseudo_dir = './pseudo'
    outdir = './work'
    tstress = .true.
    tprnfor = .true.
/

&system
    ibrav = 0
    nat = 7
    ntyp = 3
    ecutwfc = 36.0
    ecutrho = 180.0
    occupations = 'smearing'
    smearing = 'm-p'
    degauss = 0.01
    noncolin = .true.
    nspin = 2
/

&electrons
```

(continues on next page)

(continued from previous page)

```
    missing_beta = 0.1
    conv_thr = 1e-08
/
```

3.2 Generating input files

The program `cif2x` is executed with the input parameter file (`input.yaml`) and crystal structure data (`Co3SnS2_nosym.cif`) as follows.

```
$ cif2x -t QE input.yaml Co3SnS2_nosym.cif
```

The required pseudo-potential files should be placed in the directory `./pseudo`, and the index file for the pseudo-potential should be prepared as `./psudo/pp_psl_pbe_rrkjus.csv`.

Run `cif2x` and a set of input files for Quantum ESPRESSO will be created. The output file is specified by `output_file` parameter of the input parameter file, and stored in the directory given by `output_dir`. In this example, the input file for SCF calculation is created as `./scf/scf.in`.

3.3 Specifying parameter sets

In some cases, a series of input files should be generated with varying their parameter values. For example, the convergence is examined by modifying the cutoff values or grid resolution of k points. The input parameter can be given a list or a range of values, and the input files for every combination from the choices of parameter values are generated and stored in separate directories. To specify parameter set, a special syntax `${...}` is adopted.

```
content:
  K_POINTS:
    option: automatic
    grid:   ${ [ [4,4,4], [8,8,8], [12,12,12] ] }
```

When `K_POINTS` is given as above, the input files having the `grid` value to be `[4,4,4]`, `[8,8,8]`, `[12,12,12]` will be generated in the sub-directories, `4x4x4/`, `8x8x8/`, `12x12x12/`, respectively.

Chapter 4

Command reference

4.1 cif2x

Generate input files for first-principles calculation software

SYNOPSIS:

```
cif2x [-v][-q] -t target input_yaml material.cif
cif2x -h
cif2x --version
```

DESCRIPTION:

This program reads an input parameter file specified by `input_yaml` and a crystal data file specified by `material.cif`, and generates a set of input files for first-principles calculation software. In the current version, the supported software includes Quantum ESPRESSO, VASP, and OpenMX. It takes the following command line options.

- `-v`

  increases verbosity of the runtime messages. When specified multiple times, the program becomes more verbose.

- `-q`

  decreases verbosity of the runtime messages. It cancels the effect of `-v` option, and when specified multiple times, the program becomes more quiet.

- `-t` *target*

  specifies the target first-principles calculation software. The supported software for *target* is listed as follows:

  - `QE`, `espresso`, `quantum_espresso`: generates input files for Quantum ESPRESSO.
  - `VASP`: generates input files for VASP.
  - `OpenMX`: generates input files for OpenMX.
  - `AkaiKKR`: generates input files for AkaiKKR.

- `input_yaml`

  specifies an input parameter file in YAML format.

- `material.cif`

  specifies crystal structure data file. It is in CIF (Crystallographic Information Framework) format, or other format supported by pymatgen.

- `-h`

  displays help and exits.

- `--version`

  displays version information.

Chapter 5

File format

5.1 Input parameter file

An input parameter file describes information necessary to generate input files for first-principles calculation software by `cif2x`. It should be given in YAML format, and consist of the following sections.

1. structure section: describes how to handle crystal structure data.
2. optional section: describes pseudo-potential files, and symbol definitions for reference feature of YAML.
3. tasks section: describes contents of input files.

5.1.1 structure

`use_ibrav` (default value: `false`)

> This parameter specifies whether `ibrav` parameter is used for Quantum ESPRESSO as the input of the crystal structure. When it is set to `true`, the lattice is transformed to match the convention of Quantum ESPRESSO, and the lattice parameters `a`, `b`, `c`, `cosab`, `cosac`, and `cosbc` are written to the input file as needed.

`tolerance` (default value: 0.01)

> This parameter specifies the tolerance in the difference between the reconstructed Structure data and the original data when `use_ibrav` is set to `true`.

`supercell` (default value: none)

> This parameter specifies the size of supercell, when it is adopted, in the form of $[n_x, n_y, n_z]$.

5.1.2 optional

This section contains global settings needed for the first-principles calculation software. The available parameters are described in the corresponding sections below.

5.1.3 tasks

This section defines contents of the input files. It is organized as a list of blocks, each corresponding to an input file, to allow for generating a set of input files for an input. The terms described in each block are explained in the following.

`mode` (Quantum ESPRESSO)

> This parameter specifies the type of calculation. In the current version, the supported mode includes `scf` and `nscf` for pw.x of Quantum ESPRESSO. If an unsupported mode is specified, the settings in `content` will be exported as is.

`output_file` (Quantum ESPRESSO)

> This parameter specifies the file name of the output.

`output_dir`

> This parameter specifies the directory name of the output. The default value is the current directory.

`content`

> This parameter describes the content of the output. For Quantum ESPRESSO, it contains the namelist data (blocks starting from `&system`, `&control`, etc.) in `namelist` block, and other card data (such as `K_POINTS`) as individual blocks. Some card data may take parameters.

`template` (Quantum ESPRESSO)

`template_dir` (VASP)

> These parameters specifies the template file and the template directory for the input files, respectively. If they are not given, templates will not be used. The content of the template file is merged with those of `content`. The entries in the template file will be superseded by those of `content` if the entries of the same keys appear both.

5.1.4 Specifying parameter set

An input parameter may be given a list or range of parameters. In this case, a separate directory is created for every combination of parameters to store the generated input files. A special syntax `${...}` is used to specify the parameter set as follows:

- a list: `${[ A, B, ... ]}`

  a set of parameter values is described as a Python list. Each entry may be a scalar value, or a list of values.

- a range: `${range(N)}`, `${range(start, end, step)}`

  a range of parameter is given by the keyword `range`. The former specifies the values from `0` to `N-1`, and the latter from `start` to `end` with every `step`. (If `step` is omitted, it is assumed to be 1.)

5.2 Parameters for Quantum ESPRESSO

The entries of `optional` section and `content` part of the `tasks` section specific to Quantum ESPRESSO are explained below. In the current version, `scf` mode and `nscf` mode of `pw.x` are supported.

5.2.1 optional section

`pp_file`

This parameter specifies the index file in CSV format that relates the element type and the pseudo-potential file. This file contains the following columns: element name, type of pseudo-potential, nexclude, orbitals. An example line is given as:

```
Fe,pbe-spn-rrkjus_psl.0.2.1,4,spd
```

The name of the pseudo-potential file corresponding to the above example reads Fe.pbe-spn-rrkjus_psl.0.2.1.UPF .

`cutoff_file`

This parameter specifies the index file in CSV format that relates the pseudo-potential file and the cutoff values. This file contains the following columns: name of pseudo-potential file, `ecutwfc` value, `ecutrho` value.

`pseudo_dir`

This parameter specifies the name of the directory that holds pseudo-potential files. It is used when the cutoff values are obtained from the pseudo-potential files. It is indenepent from the `pseudo_dir` parameter in the input files for Quantum ESPRESSO.

5.2.2 content

namelist

- The lattice specifications in `&system` block will be superseded according to `use_ibrav` parameter in the `structure` section.
  - `use_ibrav = false`: `ibrav` is set to `0`, and the lattice parameters including `a`, `b`, `c`, `cosab`, `cosac`, `cosbc`, `celldm` are removed.
  - `use_ibrav = true`: `ibrab` is set to the index of Bravais lattices obtained from the crystal structure data. The Structure data will be reconstructed to match the convention of Quantum ESPRESSO.
- `nat` (the number of atoms) and `ntyp` (the number of element types) will be superseded by the values obtained from the crystal structure data.
- The cutoff values `ecutwfc` and `ecutrho` are obtained from the pseudo-potential files if these parameters are left blank.

CELL_PARAMETERS

- This block will not be generated if `use_ibrav` is set to `true`. Otherwise, the lattice vectors are exported in units of angstrom.
- The information of the lattice vectors are obtained from the crystal structure data. When the `data` field is defined and contains a 3x3 matrix, that value will be used for the set of lattice vectors instead.

ATOMIC_SPECIES

- This block exports a list of atom species, atomic mass, and the file name of the pseudo-potential data.
- The information of the atoms are obtained from the crystal structure data. The file names of the pseudo-potential data are referred from the CSV-formatted index file specified by `pp_list` parameter.
- When the `data` field is defined and contains the required data, these values will be used instead.

ATOMIC_POSITIONS

- This block exports the atomic species and their fractional coordinates.
- When `ignore_species` is given to specify an atomic species or a list of species, the values of `if_pos` for these species will be set to `0`. It is used for MD or structure relaxations.
- When the `data` field is defined and contains the required data, these values will be used instead.

K_POINTS

- This block exports the information of k points. The type of the output is specified by the `option` parameter that takes one of the following:
  - `gamma`: uses Γ point.
  - `crystal`: generates a list of k points in mesh pattern. The mesh width is given by the `grid` parameter, or derived from the `vol_density` or `k_resolution` parameters.
  - `automatic`: generates a mesh of k points. It is given by the `grid` parameter, or derived from the `vol_density` or `k_resolution` parameters. The shift is obtained from the `kshifts` parameter.
- The mesh width is determined in the following order:
  - the `grid` parameter, specified by a list of n_x, n_y, n_z, or a scalar value n. For the latter, $n_x = n_y = n_z = n$ is assumed.
  - derived from the `vol_density` parameter.
  - derived from the `k_resolution` parameter, whose default value is 0.15.
- When the `data` field is defined and contains the required data, these values will be used.

5.3 Parameters for VASP

The entries of `optional` section and `content` part of the `tasks` section specific to VASP are explained below.

5.3.1 optional

The type and the location of pseudo-potential files are specified.

According to pymatgen, the pseudo-potential files are obtained from `PMG_VASP_PSP_DIR`/*functional*/POTCAR.{element}(.gz) or `PMG_VASP_PSP_DIR`/*functional*/{element}/POTCAR, where `PMG_VASP_PSP_DIR` points to the directory and it is given in the configuration file `~/.config/.pmgrc.yaml` or by the environment variable of the same name. *functional* refers to the type of the pseudo-potential, whose value is predefined as `POT_GGA_PAW_PBE`, `POT_LDA_PAW`, etc.

`pseudo_functional`

This parameter specifies the type of the pseudo-potential. The relation to the *functional* value above is defined in the table of pymatgen, for example, by `PBE` to `POT_GGA_PAW_PBE`, or by `LDA` to `POT_LDA_PAW`, or in a similar manner.

When the `pseudo_dir` parameter is specified, it is used as the directory that holds the pseudo-potential files, ignoring the convention of pymatgen.

`psuedo_dir`

> This parameter specifies the directory that holds the pseudo-potential files. The paths to the pseudo-potential file turn to `pseudo_dir`/POTCAR.{element}(.gz), or `pseudo_dir`/{element}/POTCAR.

5.3.2 tasks

The template files are assumed to be placed in the directory specified by the `template_dir` parameter by the names `INCAR`, `KPOINTS`, `POSCAR`, and `POTCAR`. The missing files will be ignored.

5.3.3 content

incar

- This block contains parameters described in the INCAR file

kpoints

- `type`

  The `type` parameter describes how KPOINTS are specified. The following values are allowed, with some types accepting parameters. See pymatgen.io.vasp manual for further details.

  - `automatic`

    parameter: `grid`

  - `gamma_automatic`

    parameter: `grid`, `shift`

  - `monkhorst_automatic`

    parameter: `grid`, `shift`

  - `automatic_density`

    parameter: `kppa`, `force_gamma`

  - `automatic_gamma_density`

    parameter: `grid_density`

  - `automatic_density_by_vol`

    parameter: `grid_density`, `force_gamma`

  - `automatic_density_by_lengths`

    parameter: `length_density`, `force_gamma`

  - `automatic_linemode`

    parameter: `division`, `path_type` (corresponding to the `path_type` parameter of HighSymmKpath.)

5.4 Parameters for OpenMX

The entries of `optional` section and `content` part of the `tasks` section specific to OpenMX are explained below.

5.4.1 optional

`data_path`

This parameter specifies the name of directory that holds files for pseudo-atomic orbitals and pseudo-potentials. It corresponds to the `DATA.PATH` parameter.

5.4.2 content

`precision`

This parameter specifies the set of pseudo-atomic orbitals listed in Tables 1 and 2 of Section 10.6 of the OpenMX manual. It is one of `quick`, `standard`, or `precise`. The default value is `quick`.

5.5 Parameters for AkaiKKR

The entries of `optional` section and `content` part of the `tasks` section specific to AkaiKKR are explained below.

5.5.1 optional

`workdir`

This parameter specifies the directory in which temporal files are stored. If it is not given, `/tmp` or the value of the environment variable `TMPDIR` is used.

5.5.2 content

The `content` part contains the input parameters of AkaiKKR. A blank is written to the input file for an unspecified parameter, to which the default value defined in AkaiKKR will be assumed. The parameter values listed below are replaced by the values obtained from the crystal structure data.

- `brvtyp`, except when it is set to `aux` (or a string that contains `aux`).
- lattice parameters, `a`, `c/a`, `b/a`, `alpha`, `beta`, `gamma`, `r1`, `r2`, `r3`.
- type information, `ntyp`, `type`, `ncmp`, `rmt`, `field`, `mxl`, `anclr`, `conc`.
- element information, `natm`, `atmicx`, `atmtyp`.

For `rmt` and `field`, the values specified in the input parameter file will be used only when they are lists having the same number of elements as `ntyp`.

Chapter 6

Extension guide

6.1 Adding modes of Quantum ESPRESSO

In order to add supports to modes of Quantum ESPRESSO, the mapping between the modes and the transformation classes should be added to `create_modeproc()` function in `src/cif2x/qe/calc_mode.py`.

```
def create_modeproc(mode, qe):
    if mode in ["scf", "nscf"]:
        modeproc = QEmode_pw(qe)
    else:
        modeproc = QEmode_generic(qe)
    return modeproc
```

The transformation functionality for each mode is provided as a derived class of `QEmode_base` class. This class implements methods `update_namelist()` for updating the namelist block, and `update_cards()` for generating data of card blocks. In the current version, two classes are provided: `QEmode_pw` class for scf and nscf calculations of pw.x, and `QEmode_generic` class for generating output as-is.

```
class QEmode_base:
    def __init__(self, qe):
    def update_namelist(self, content):
    def update_cards(self, content):
```

For the namelist, the transformation class generates values for blank entries from crystal structure data and other sources. It may also force to set values such as the lattice parameters that are determined from the crystal structure data, or those that must be specified consistently with other parameters. The functions are provided for each mode separately.

For card blocks, a function is provided for each card, and the mapping between the card type and the function is given in the `card_table` variable. The method `update_cards()` in the base class picks up and runs the function associated to the card, and updates the content of the card. Of course, a new `update_cards()` function may be defined.

```
self.card_table = {
    'CELL_PARAMETERS': generate_cell_parameters,
    'ATOMIC_SPECIES': generate_atomic_species,
    'ATOMIC_POSITIONS': generate_atomic_positions,
    'K_POINTS': generate_k_points,
}
```

The functions for cards are gathered in `src/cif2x/qe/cards.py` with the function names as `generate_{card name}`. These functions takes parameters for card blocks as argument, and returns a dictionary containing the card name, the options, and the data field.

Chapter 7

A tool to retrieve crystallographic data from databases (getcif)

7.1 Introduction

`getcif` is a tool to retrieve crystallographic information and other properties of materials from databases. The latest version of getcif provides access to Materials Project database. Users can search database and obtain information by specifying symmetry, composition, or physical properties of materials.

7.2 Tutorial

In this tutorial, the procedure to use the database query tool `getcif` is described for searching and obtaining crystallographic information from databases for the materials science. It consists of getting an API key, preparing an input parameter file, and running the getcif program. We will explain the steps along an example of searching and obtaining information for ABO3-type materials provided in the `docs/tutorial/getcif` directory.

7.2.1 Getting an API key

In order to access the Materials Project database via API, users need to register to the Materials Project and obtain an API key. Visit the Materials Project website https://next-gen.materialsproject.org, create an account and do Login. An API key is automatically generated on registration and shown in the user dashboard. The API key should be kept safe and not shared with others.

The API key is made available to getcif by one of the following ways:

(a) storing in the pymatgen configuration file by typing in as follows:

```
$ pmg config --add PMG_MAPI_KEY <API_KEY>
```

or editing the file `~/.config/.pmgrc` to include the following:

```
PMG_MAPI_KEY: <API_KEY>
```

(b) setting to an environment variable by:

```
$ MP_API_KEY="<API_KEY>"
$ export MP_API_KEY
```

(c) storing the API key to a file located in the directory where getcif is run. The default value of the file name is `materials_project.key`. Otherwise, it is given in the input parameter file. The file name must end with `.key`.

```
database:
  api_key_file: materials_project.key
```

Comment: it will be recommended to exclude files with `.key` as a suffix from version control system. (e.g. for Git, add `*.key` in `.gitignore` file.)

7.2.2 Prepare an input parameter file

An input parameter file describes search conditions and data items to retrieve from databases.

An example is presented below. It is a text file in YAML format that contains information for accessing the database, search conditions, and types of data to obtain. See *file format* section for the details of specification.

In YAML format, parameters are given in dictionary form as `keyword: value`, where `value` is a scalar such as a number or a string, or a set of values enclosed in `[ ]` or listed in itemized form, or a nested dictionary. For the search conditions and data fields, a list may be given by space-separated items without brackets as a special notation.

```
database:
  target: materials project

option:
  output_dir: result
  # dry_run: false

properties:
  band_gap: < 1.0
  is_stable: true
  is_metal: false
  formula: "**O3"
  spacegroup_symbol: Pm-3m

fields: |
  structure
  band_gap
  symmetry
```

The input parameter file consists of `database`, `option`, `properties`, and `fields` sections. The `database` section describes settings about connecting to databases. In the example, `target` is set to Materials Project, though this term is not considered at present. `api_key` can be used to set the API key. The key may also be set in the pymatgen configuration file or in the environment variable. The latter is assumed in the tutorial.

The `option` section describes optional settings for the command execution. `output_dir` specifies the directory to place the obtained data. The default is the current directory. If `dry_run` is set to `true`, getcif does not connect to the database; instead, it just prints the search conditions and exits. `dry_run` may be specified in the command-line option.

The `properties` section describes search conditions. They are given in the form of `keyword: value` and treated as AND conditions. In the example, the search condition is specified to find materials with band gap less than or equal to 1.0, stable insulator, having composition formula of ABO3 (where A and B are arbitrary species), that belong to the space group `Pm-3m` (perovskite). The `band_gap` takes a pair of values for the lower and upper limits, as well as the description such as `< 1.0`. The available terms for specifying search conditions are listed in the Appendix.

The `fields` section describes the data items to obtain. It is given as a YAML list, or a space-sparated list. `structure` specifies the crystal structure data that will be stored in CIF format. `band_gap` specifies the value of band gap, and `symmetry` specifies the information on the symmetry. `material_id` that refers to the index of material data in the Materials Project, and `formula_pretty` that refers to the composition formula are automatically obtained. The available items are listed in the Appendix, or can be found in the help message of getcif command.

7.2.3 Obtaining data

The program `getcif` is executed with the input parameter file (`input.yaml`) as follows.

```
$ getcif input.yaml
```

Then it connects to the Materials Project database, and obtains the data that match the specified conditions. The summary including the material IDs, the composition formulas, and other data items is printed to the standard output as follows.

```
material_id  formula  band_gap  symmetry  formula_pretty
mp-861502  AcFeO3  0.9887999999999995  crystal_system=<CrystalSystem.cubic: 'Cubic'>␣
↪symbol='Pm-3m' number=221 point_group='m-3m' symprec=0.1 version='2.0.2'  AcFeO3
mp-977455  PaAgO3  0.915  crystal_system=<CrystalSystem.cubic: 'Cubic'> symbol='Pm-3m'␣
↪number=221 point_group='m-3m' symprec=0.1 version='2.0.2'  PaAgO3
mp-11775  RbUO3  0.45420000000000016  crystal_system=<CrystalSystem.cubic: 'Cubic'>␣
↪symbol='Pm-3m' number=221 point_group='m-3m' symprec=0.1 version='2.0.2'  RbUO3
mp-3163  BaSnO3  0.37239999999999984  crystal_system=<CrystalSystem.cubic: 'Cubic'>␣
↪symbol='Pm-3m' number=221 point_group='m-3m' symprec=0.1 version='2.0.2'  BaSnO3
mp-4126  KUO3  0.44540000000000024  crystal_system=<CrystalSystem.cubic: 'Cubic'> symbol=
↪'Pm-3m' number=221 point_group='m-3m' symprec=0.1 version='2.0.2'  KUO3
mp-865322  UTlO3  0.27360000000000007  crystal_system=<CrystalSystem.cubic: 'Cubic'>␣
↪symbol='Pm-3m' number=221 point_group='m-3m' symprec=0.1 version='2.0.2'  UTlO3
mp-753781  EuHfO3  0.47959999999999996  crystal_system=<CrystalSystem.cubic: 'Cubic'>␣
↪symbol='Pm-3m' number=221 point_group='m-3m' symprec=0.1 version='2.0.2'  EuHfO3
```

The obtained data are placed in the directory specified by `output_dir` with the subdirectories of the material ID for each material. In this example, seven subdirectories with names from mp-3163 to mp-977455 are created within `result` directory, and each subdirectory contains the following files:

- **band_gap**
  the value of band gap
- **formula**
  the composition formula (that corresponds to the field `formula_pretty`)
- **structure.cif**
  the crystal structure data in CIF format
- **symmetry**
  the information about symmetry

If an option `--dry-run` is added as a command-line option to `getcif`, the program prints the search condition as follows, and exits. It will be useful for checking the search parameters.

```
$ getcif --dry-run input.yaml
{'band_gap': (None, 1.0), 'is_stable': True, 'is_metal': False, 'formula': '**O3',
 'spacegroup_symbol': 'Pm-3m', 'fields': ['structure', 'band_gap', 'symmetry', 'material_
→id', 'formula_pretty']}
```

7.3 Command reference

7.3.1 getcif

Retrieve crystallographic and other data from databases.

SYNOPSIS:

```
getcif [-v][-q] [--dry-run] input_yaml
getcif -h
getcif --version
```

DESCRIPTION:

This program reads an input parameter file specified by `input_yaml`, and connects to the database to submit a query and obtain the crystallographic data of materials. It takes the following command line options.

- `input_yaml`

  specifies an input parameter file in YAML format.

- `-v`

  increases verbosity of the runtime messages. When specified multiple times, the program becomes more verbose.

- `-q`

  decreases verbosity of the runtime messages. It cancels the effect of `-v` option, and when specified multiple times, the program becomes more quiet.

- `--dry-run`

  displays search parameters and exits without connecting to the database. It allows to confirm the search conditions. This option supersedes the `dry_run` parameter in the input file.

- `-h`

  displays help and exits.

- `--version`

  displays version information.

7.4 File format

7.4.1 Input parameter file

An input parameter file describes information to search for crystallographic and other data from Materials Project database by getcif. It should be given in YAML format, and consist of the following sections.

1. database section: describes information on the database to connect.
2. option section: describes output directory and other parameters for command execution.
3. properties section: describes search conditions.
4. fields section: describes types of data to be retrieved.

database

`target`

This parameter specifies the database to connected to. At present this parameter is ignored.

`api_key_file` (default value: `materials_project.key`)

This parameter specifies a name of a file that contains the API key to access to the database. The suffix of the file name must be `.key`. If the file does not exist or it does not contain a valid value, the API key is obtained from the environment variable `MP_API_KEY`, or from the parameter `PMG_MAPI_KEY` of the pymatgen configuration file in `~/.config/.pmgrc`.

The API key file is a text file. A line starting with `#` is regarded as a comment. The heading and trailing spaces are ignored. When the file contains more than one line, the API key is taken from the first valid line.

option

This section contains global settings needed for the first-principles calculation software. The available parameters are described in the corresponding sections below.

`output_dir` (default value: `""`)

This parameter specifies the directory name to store the data. The retrieved data are placed in this directory under the subdirectories by the material ID for each material. The default value is the current directory.

`dry_run` (default value: `False`)

When this parameter is set to True, getcif prints the search conditions and exists without connecting to the database. It is useful to check the content of the query.

`symprec` (default value: 0.1)

This parameter specifies the tolerance in calculating the symmetry of a crystal structure when the structure data are written to a CIF file. By default, 0.1 is specified. When `symprec` is set to 0.0, it is treated as if `symprec` is unspecified, in which case a CIF file is generated without considering symmetry.

`symprec` is a parameter that specifies the tolerance used to determine the symmetry of a crystal structure. When calculating the symmetry of a crystal structure, it is essential to consider the slight displacements of atomic positions and the precision of numerical calculations. `symprec` controls the allowable range of these displacements and serves as a threshold for deciding whether a symmetry operation should be applied.

If `symprec` is set to a smaller value (e.g., 0.01), the symmetry determination becomes more stringent, and even minor displacements in the crystal structure may prevent the application of symmetry operations. This can result in the identification of a lower-symmetry space group. Conversely, if `symprec` is set to a larger value (e.g., 1.0), the symmetry determination is more lenient, allowing small displacements to be ignored, which may lead to the recognition of a higher-symmetry space group.

When the `symmetry` field is specified in the fields section, the symmetry information determined using the default `symprec=0.1` in the Materials Project is obtained and written to a text file (`symmetry`).

properties

This section defines the search conditions. The conditions such as the element types, the crystal symmetry, or the values of physical properties are specified in the `keyword: value` format. They are treated as AND condition. The available terms, based on the Materials Project API, conform to the parameters of the `materials.summary.search` method in the mp-api library. The list of terms are summarized in the Appendix, and can be seen by `getcif --help`.

The format of the parameter values is shown below. It follows the YAML specification with several extension for brief description.

- a number, a string

  describe as-is.

- a boolean value

  describe as `true` or `false`.

- a list of numbers or strings

  describe in the indented style (block style) or in the comma-separated list enclosed by the bracket (flow style) in YAML notation. It is also available that it is described as a space-separated list, for example:

  ```
  element: Sr Ti
  ```

- a range of numerical value

  described as a list of two numbers such as `[ min, max ]`, or a pair of two numbers separated by a space as `min max`. The following formats are also available.

  `<= max`
  less than or equal to `max`.

  `< max`
  less than `max`. (For a real number, it is equivalent to <= `max`. For an integer, it is treated as <= `max`-1.)

  `>= min`
  more than or equal to `min`.

  `> min`
  more than `min`. (For a real number, it is equivalent to >= `min`. For an integer, it is treated as >= `min`+1.)

  `min ~ max`
  between `min` and `max`.

  N.B.:

  - A space must be placed between the symbol and the number.

  - Due to the YAML syntax that the symbol ">" at the beginning of a term is treated as a special character, `> min` and `>= min` should be enclosed by quotes as `"> min"` and `">= min"`, respectively.
  - In list notations, `<= max` and `>= min` are denoted as `[ None, max ]` and `[ min, None ]`, respectively.

- wild card symbols

  The term `formula` accepts wild card symbols * for elements. In this case, the whole value is enclosed by `" "`. For example,

  ```
  formula: "**O3"
  ```

  for ABO_3-type materials.

fields

This section defines the types of data to be retrieved. A list of types is described in the YAML format, or as a space-sparated strings. In the latter format, it can be given in multiple-line format using the “ `|` ” notation of YAML.

The available types of data conform to the `field` parameter of the Materials Project API. They are listed in the Appendix, and can be viewd by `getcif --help`.

The types `material_id` and `formula_pretty` are retrieved automatically.

The obtained data are placed in the directory specified by `output_dir` parameter under the subdirectories of the material_id for each material. Each item is stored as a separate file of the item name. The crystal structure data (`structure`) is stored in a file `structure.cif` in CIF format.

7.5 Parameter List

7.5.1 Search conditions (properties)

Table *Search criteria* summarizes condition terms available in the properties section.

`getcif` uses the `mp-api` library provided by Materials Project as a client for accessing the database via Materials Project API. The condition terms correspond to the parameters for the `materials.summary.search` method of MPRester class in `mp-api`. (The content of the table is taken and reformatted from the comments of the source file in `mpi-api`.)

The types of the parameter values denote as follows:

- `str`: a string
- `List[str]`: a list of strings
- `str | List[str]`: a string or a list of strings
- `int`: an integer
- `bool`: a boolean value (`true` or `false`)
- `Tuple[float,float]`: a pair of two floating point numbers (as a list)
- `Tuple[int,int]`: a pair of two integers (as a list)
- `CrystalSystem`: a string representing the crystal system, one of the following: Triclinic, Monoclinic, Orthorhombic, Tetragonal, Trigonal, Hexagonal, Cubic

- `List[HasProps]`: a list of strings representing the properties defined in `emmet.core.summary`. The available terms include:

  absorption, bandstructure, charge_density, chemenv, dielectric, dos, elasticity, electronic_structure, eos, grain_boundaries, insertion_electrodes, magnetism, materials, oxi_states, phonon, piezoelectric, provenance, substrates, surface_properties, thermo, xas

- `Ordering`: a string representing the magnetic ordering, one of the following: FM, AFM, FiM, NM

A list of the values is described in an indented style or in a comma-separated bracketted style in YAML notation. It is also available that it is described as a space-separated list.

A `Tuple` is used to denote a range of values by `min` and `max`. It is described by a list of two numbers, as well as by a space-separated list as `min max`. The following notation is also available:

`< max`
: less than or equal to `max`

`> min`
: more than or equal to `min`

`min ~ max`
: between `min` and `max`

Table 7.1: Search criteria

Keyword	Type	Description
band_gap	Tuple[float,float]	Minimum and maximum band gap in eV to consider.
chemsys	str | List[str]	A chemical system, list of chemical systems (e.g., Li-Fe-O, Si-*, [Si-O, Li-Fe-P]), or single formula (e.g., Fe2O3, Si*).
crystal_system	CrystalSystem	Crystal system of material.
density	Tuple[float,float]	Minimum and maximum density to consider.
deprecated	bool	Whether the material is tagged as deprecated.
e_electronic	Tuple[float,float]	Minimum and maximum electronic dielectric constant to consider.
e_ionic	Tuple[float,float]	Minimum and maximum ionic dielectric constant to consider.
e_total	Tuple[float,float]	Minimum and maximum total dielectric constant to consider.
efermi	Tuple[float,float]	Minimum and maximum fermi energy in eV to consider.
elastic_anisotropy	Tuple[float,float]	Minimum and maximum value to consider for the elastic anisotropy.
elements	List[str]	A list of elements.
energy_above_hull	Tuple[int,int]	Minimum and maximum energy above the hull in eV/atom to consider.
equilibrium_reaction_energy	Tuple[float,float]	Minimum and maximum equilibrium reaction energy in eV/atom to consider.
exclude_elements	List[str]	List of elements to exclude.
formation_energy	Tuple[int,int]	Minimum and maximum formation energy in eV/atom to consider.
formula	str | List[str]	A formula including anonymized formula or wild cards (e.g., Fe2O3, ABO3, Si*). A list of chemical formulas can also be passed (e.g., [Fe2O3, ABO3]).
g_reuss	Tuple[float,float]	Minimum and maximum value in GPa to consider for the Reuss average of the shear modulus.
g_voigt	Tuple[float,float]	Minimum and maximum value in GPa to consider for the Voigt average of the shear modulus.

continues on next page

Table 7.1 – continued from previous page

Keyword	Type	Description
g_vrh	Tuple[float,float]	Minimum and maximum value in GPa to consider for the Voigt-Reuss-Hill average of the shear modulus.
has_props	List[HasProps]	The calculated properties available for the material.
has_reconstructed	bool	Whether the entry has any reconstructed surfaces.
is_gap_direct	bool	Whether the material has a direct band gap.
is_metal	bool	Whether the material is considered a metal.
is_stable	bool	Whether the material lies on the convex energy hull.
k_reuss	Tuple[float,float]	Minimum and maximum value in GPa to consider for the Reuss average of the bulk modulus.
k_voigt	Tuple[float,float]	Minimum and maximum value in GPa to consider for the Voigt average of the bulk modulus.
k_vrh	Tuple[float,float]	Minimum and maximum value in GPa to consider for the Voigt-Reuss-Hill average of the bulk modulus.
magnetic_ordering	Ordering	Magnetic ordering of the material.
material_ids	List[str]	List of Materials Project IDs to return data for.
n	Tuple[float,float]	Minimum and maximum refractive index to consider.
num_elements	Tuple[int,int]	Minimum and maximum number of elements to consider.
num_sites	Tuple[int,int]	Minimum and maximum number of sites to consider.
num_magnetic_sites	Tuple[int,int]	Minimum and maximum number of magnetic sites to consider.
num_unique_magnetic_sites	Tuple[int,int]	Minimum and maximum number of unique magnetic sites to consider.
piezoelectric_modulus	Tuple[float,float]	Minimum and maximum piezoelectric modulus to consider.
poisson_ratio	Tuple[float,float]	Minimum and maximum value to consider for Poisson's ratio.
possible_species	List[str]	List of element symbols appended with oxidation states. (e.g. Cr2+,O2-)
shape_factor	Tuple[float,float]	Minimum and maximum shape factor values to consider.
spacegroup_number	int	Space group number of material.
spacegroup_symbol	str	Space group symbol of the material in international short symbol notation.
surface_energy_anisotropy	Tuple[float,float]	Minimum and maximum surface energy anisotropy values to consider.
theoretical	bool	Whether the material is theoretical.
total_energy	Tuple[int,int]	Minimum and maximum corrected total energy in eV/atom to consider.
total_magnetization	Tuple[float,float]	Minimum and maximum total magnetization values to consider.
to-tal_magnetization_normalized	Tuple[float,float]	Minimum and maximum total magnetization values normalized by formula units to consider.
to-tal_magnetization_normalized	Tuple[float,float]	Minimum and maximum total magnetization values normalized by volume to consider.
uncorrected_energy	Tuple[int,int]	Minimum and maximum uncorrected total energy in eV/atom to consider.
volume	Tuple[float,float]	Minimum and maximum volume to consider.
weighted_surface_energy	Tuple[float,float]	Minimum and maximum weighted surface energy in J/m^2 to consider.
weighted_work_function	Tuple[float,float]	Minimum and maximum weighted work function in eV to consider.

7.5.2 Data to retrive (fields)

The items available for the `fields` section for retrieving from the database are listed below.

```
band_gap
bandstructure
builder_meta
bulk_modulus
cbm
chemsys
composition
composition_reduced
database_IDs
decomposes_to
density
density_atomic
deprecated
deprecation_reasons
dos
dos_energy_down
dos_energy_up
e_electronic
e_ij_max
e_ionic
e_total
efermi
elements
energy_above_hull
energy_per_atom
equilibrium_reaction_energy_per_atom
es_source_calc_id
formation_energy_per_atom
formula_anonymous
formula_pretty
grain_boundaries
has_props
has_reconstructed
homogeneous_poisson
is_gap_direct
is_magnetic
is_metal
is_stable
last_updated
material_id
n
nelements
nsites
num_magnetic_sites
num_unique_magnetic_sites
ordering
origins
possible_species
property_name
```

(continues on next page)

(continued from previous page)

```
shape_factor
shear_modulus
structure
surface_anisotropy
symmetry
task_ids
theoretical
total_magnetization
total_magnetization_normalized_formula_units
total_magnetization_normalized_vol
types_of_magnetic_species
uncorrected_energy_per_atom
universal_anisotropy
vbm
volume
warnings
weighted_surface_energy
weighted_surface_energy_EV_PER_ANG2
weighted_work_function
xas
```