The `cif2cell` manual

Torbjörn Björkman

July 10, 2012

Introduction – Of CIF's and Cells

An electronic structure program needs atomic positions and, at least in the case of band structure programs, unit cell vectors to calculate the electronic structure of a crystal. Experimental data on the other hand tends to be given as a space group and a set of irreducible, or wyckoff positions, and are often distributed in the form of CIF files. There is a wide range of sources of CIF files – it is actually used for communication of crystallographic data by virtually everyone outside of the electronic structure community! Yet far from all electronic structure programs can read the CIF format, and from this circumstance a small set of my convenience scripts one day turned into the program `cif2cell`.

What the program does is simply to take the crystal structure information in a CIF file, typically space group information and a set of irreducible coordinates, and then set up a calculation cell which is then output to your favourite electronic structure program (ESP). As you are probably aware, the choice of calculation cell for a given crystal is arbitrary and infinitely many possibilities exist for any given crystal. Perhaps the simplest example is when a problem require us to set up a supercell, which is just describing the same crystal in a different cell than the primitive one, but even the primitive cell can often be chosen in many different ways. `cif2cell` contains working solutions for choices of primitive cells for (almost) any possible crystal and will make a choice automatically. You may choose either the primitive cell for the given structure, such as the rhombohedral cell with one atom for fcc Cu, or the conventional cell, a cubic cell with four atoms for fcc Cu. These cells can then be used as building blocks for generating supercells by multiplying them up and/or inputting extra vacuum in the structure.

This manual is intended as a document for computational physicists and materials scientists. The problems that you solve when generating a computational cell from crystallographic data belongs to the field of crystallography and the program is not intended to replace a working knowledge of basic crystallography. Remember this, and refer to your textbook in crystallography or solid state physics/chemistry when you get confused and your life with `cif2cell` will be happy.

The code was published in Computer Physics Communications **182**, 1183–1186 (2011), please cite generously. Happy computing!

1 Installation

1.1 Requirements

`cif2cell` requires Python 2.4 or higher and the PyCIFRW python package[1]. Note however that the output may be slightly different (but equivalent) with Python 2.4 than with later versions due to differences in the internal sorting routines of different Python distributions.

To install the program in your systems standard location, type:

```
python setup.py install
```

To choose a different location, add

```
--prefix=where/you/want/it
```

to the above line. For help and more options type

```
python setup.py --help
```

2 Getting started

This section is meant to give you a quick overview of how you generate cells to get you started as quickly as possible. Running `cif2cell` is quick, so the best way to learning to work with it is to simply run it over and over, testing different settings and options until you have what you want. The program comes with a small set of sample CIF files which can be found either in the source distribution or in the directory `[prefix]/lib/cif2cell/sample_cifs`, where `[prefix]` is the path where you installed `cif2cell`. The examples below assume that you stand in a directory containing these sample files. In the following, some arbitrary command line input will be specified within brackets `[like this]`.

2.1 Help!

The quickest way to get help in `cif2cell` is to type

```
cif2cell -h
```

This will list all available input options along with a description.

2.2 The first run

There is only one required piece of input, and that is the CIF file itself which is given either as the first argument to the program:

```
cif2cell Si.cif [options]
```

[1] Available from `http://pycifrw.berlios.de`.

or anywhere in the argument list with the `-f` flag:

```
cif2cell [someoptions] -f Si.cif [otheroptions]
```

and if no output program is specified, the cell information is output to screen:

```
CIF2CELL 0.4.0
2011-03-25 16:35
Output for Si (Silicon)
CIF file exported from Inorganic Crystal Structure Database.
Database reference code: 51688.

 BIBLIOGRAPHIC INFORMATION
Toebbens, D.M. et al., Materials Science Forum 378, 288-293 (2001)

 SYMMETRY INFORMATION
Cubic crystal system.
Space group number     : 227
Hall symbol            : F 4d 2 3 -1d
Hermann-Mauguin symbol : Fd-3mS

 INPUT CELL INFORMATION
Lattice parameters:
          a           b           c
  5.4305300   5.4305300   5.4305300
      alpha        beta       gamma
 90.0000000  90.0000000  90.0000000
Representative sites :
Atom           a1          a2          a3
Si      0.0000000   0.0000000   0.0000000

 OUTPUT CELL INFORMATION
Bravais lattice vectors :
  0.5000000   0.5000000   0.0000000
  0.5000000   0.0000000   0.5000000
  0.0000000   0.5000000   0.5000000
All sites, (lattice coordinates):
Atom           a1          a2          a3
Si      0.0000000   0.0000000   0.0000000
Si      0.2500000   0.2500000   0.2500000
```

First there is some information about the program itself and a description of the compound derived from information in the CIF file. If the file comes from a databases known to `cif2cell`, information about that is also printed. Then follows any bibliographic information found in the file. Then starts the actual crystal information with data about the space group. If the `-v` or `--print-symmetry-operations` flags are given, all symmetry operations will also be printed here. Next comes the lattice parameters and representative sites (occupied wyckoff positions), with the the chemical elements and, in case of an alloy, another column with the site occupancies.

Then follows the things we actually want – the Bravais lattice vectors and all positions of the atoms. In the case of Si in the diamond structure the result is probably familiar: the standard fcc lattice vectors and two atoms in the basis, one at $(0, 0, 0)$ and another in $(\frac{1}{4}, \frac{1}{4}, \frac{1}{4})$. For the fcc lattice the choice of primitive lattice vectors is straightforward, but in many systems of lower symmetry, in particular monoclinic systems, the choice is not always as simple. `cif2cell` has default choices for the Bravais lattice vectors that were selected to produce calculation cells similar to those listed at the Naval Research Laboratory site `http://cst-www.nrl.navy.mil/lattice/`.

By default, `cif2cell` will reduce the cell to the primitive cell, anticipating that the user wants to calculate something as small as possible, but we can also get the conventional cell. The diamond structure is a cubic system, and the primitive cell has only two atoms. Now try to also give the program the option `--no-reduce`. You should now get 8 atoms and cubic lattice vectors. Note that the conventional and primitive cell are the same in many systems.[2]

2.3 Output to your electronic structure program

Having generated the basic cell we want to export that to our favourite ESP. This is done by simply giving the option `-p espname` or `--program=espname`. The possible choices for `espname` can be found in Table 1, where you also find what files will be generated. You may also decide the output file name yourself by giving the option `-o outputfilename` or `---outputfile=outputfilename`. If you wish to append the output from `cif2cell` at the end of some existing file, give the option `-a` or `--append`, but note that this requires that you also specify the output file name. Appending to existing files is useful for example if you have a template file with your magic settings and just wish to add the cell with the atoms. You will notice that for nearly all ESP's, `cif2cell` generates *only* the geometric part of the setup, unless you specify the `--setup-all` flag, which not supported for many ESP's right now.

Generation of cells for alloy theory calculations is possible for ESP's that implement such features. For other ESP's, you may use the `--force-alloy` option to generate as much of the geometric input as possible.

Warning! Output generated with `--force-alloy` *always* needs further editing. The

[2] In all systems which has a space group, or Hermann-Mauguin (H-M), symbol that starts with "P" (for "primitive"). See the example for fcc Si above, where the H-M symbol is 'F', (from German "Flächenzentriert", meaning face-centered).

partially occupied sites will get some placeholder string where the element should be specified, no attempt is made by `cif2cell` to guess what you want to do.

Table 1: Supported electronic structure programs (ESP) and the files output by `cif2cell`. Also shown is whether the ESP implements some alloy theory and whether `cif2cell` currently supports full output.

ESP	Alloy support	Full setup	Output file(s)
ABINIT	no	no	[compoundname].in
CASTEP	no	no	[compoundname].cell
cellgen	no	no	cellgen.inp
CPMD	no	no	[compoundname].inp
Crystal09	no	no	[compoundname].d12
Fleur	no	no	inp_[compoundname]
RSPt	no	no	symt.inp
Elk	no	no	GEOMETRY.OUT
EMTO	yes	no	[spacegroupname/compoundname].dat for kstr, bmdl, shape, kgrn and kfcd in separate directories.
Exciting	no	no	input.xml
ncol	no	no	[spacegroupname/compoundname].dat for bstr and ncol.
Siesta	no	no	[compoundname].fdf
Spacegroup	no	no	spacegroup.in
SPRKKR	yes	no	[compoundname].sys (via XBAND)
VASP	no	yes	POSCAR
XBAND	yes	no	[compoundname].sys