CIF2Cell manual

Torbjörn Björkman

July 5, 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`.

The code was published in Computer Physics Communications **182**, 1183–1186 (2011). I hope you find it useful.

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 Basic running

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 most immediate 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]
```

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:

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

```
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
Space group number      : 227
Space group symbol (H-M): Fd-3mS
Cubic crystal system.

 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 $(\frac{1}{4}, \frac{1}{4}, \frac{1}{4})$.

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

[2] In all systems which has a space group (or Hermann-Mauguin) symbol that starts with "P", for "primitive", see the output above.

Table 1: Supported Electronic structure programs and the files output by `cif2cell`.

Code	Alloy support	Output file(s)
VASP	no	POSCAR
ABINIT	no	[compoundname].in
Siesta	no	[compoundname].fdf
CPMD	no	[compoundname].inp
CASTEP	no	[compoundname].cell
Crystal09	no	[compoundname].d12
RSPt	no	symt.inp
Fleur	no	inp_[compoundname]
cellgen	no	cellgen.inp
elk	no	GEOMETRY.OUT
exciting	no	input.xml
spacegroup	no	spacegroup.in
ncol	no	[spacegroupname/compoundname].dat for bstr and ncol.
emto	yes	[spacegroupname/compoundname].dat for kstr, bmdl, shape, kgrn and kfcd in separate directories.