Getting a Crystal Structure into Abinit

J. W. Zwanziger

23 April 2013

1 Introduction

The Abinit program computes the electronic structure of a solid with periodic boundary conditions, so it always needs in the input a description of the solid. Most users use as the starting point of their calculation the crystallographic data (either experimental or from a previous computation) describing a solid. These data are typically given in the form of a `.cif` file (the standard format for exchange of crystal structure data) or from crystallographic data published in an article. The aim of these notes is to provide some hints to get users started in correctly entering these data into Abinit.

2 An Easy Case: Stishovite

Stishovite is a polymorph of SiO_2. The reason I call it an "easy case" is that its space group, number 136 in the International Tables, is $P4_2/mnm$ and therefore the conventional unit cell and the primitive unit cell are the same.

KEY POINT: A unit cell that displays all the symmetries of the space group is called the "conventional" unit cell. Some space groups also have unit cells, which are smaller than the conventional cells, that do not display all the symmetries. These are called "primitive" unit cells. Of the 236 crystallographic space groups, those with Hermann-Marguin names beginning with the letter P have conventional and primitive cells that coincide, while the others do not. Because the primitive cell always has the fewest number of atoms that can be used to construct the solid, Abinit always defaults to use the primitive cell.

The `.cif` file of stishovite contains a lot of data. Here are the parts we care about:

```
_chemical_formula_structural 'Si O2'
_cell_length_a 4.1790(4)
_cell_length_b 4.1790(4)
_cell_length_c 2.6649(4)
_cell_angle_alpha 90.
_cell_angle_beta 90.
_cell_angle_gamma 90.
```

```
_cell_volume 46.54
_cell_formula_units_Z 2
_symmetry_space_group_name_H-M 'P 42/m n m'
_symmetry_Int_Tables_number 136
loop_
_atom_site_label
_atom_site_type_symbol
_atom_site_symmetry_multiplicity
_atom_site_Wyckoff_symbol
_atom_site_fract_x
_atom_site_fract_y
_atom_site_fract_z
_atom_site_B_iso_or_equiv
_atom_site_occupancy
_atom_site_attached_hydrogens
Si1 Si4+ 2 a 0 0 0 . 1. 0
O1 O2- 4 f 0.3062(13) 0.3062(13) 0 . 1. 0
```

The first thing to look for is the space group name, which is $P4_2/mnm$ as mentioned and therefore the unit cell is both conventional and primitive already. The cell itself happens to be tetragonal: two sides are equal, with length 4.1790 Å(`.cif` files use Angstroms, not atomic units) and the third side of length 2.6649 Å. The edges of the cell all meet at 90°: these are the angles `alpha`, `beta`, `gamma` and are the angles between sides BC, AC, and AB respectively.

Next, notice that the number of formula units Z, is 2. Therefore there will be 2 units of SiO_2 in the unit cell, or 6 atoms total: 2 Si and 4 O. Where are these atoms? The loop over atom sites tells us, but notice that there are only two entries: 1 Si and 1 O. That's because the other locations can be constructed by symmetry, given the symmetry operations in space group 136. These two atoms form what is called the "asymmetric unit". Notice that it is NOT a multiple of the formula unit. The lines describing the two atoms include a lot of information, including the site multiplicity, fractional occupancy, and so forth. These notes are not a full course in crystallography so I'm not going to try to explain all that here, what you need to get started with ABINIT are the fractional positions, which are entries 5–7 in each line.

So, how does all this look in an ABINIT input file? Like this:

```
spgroup 136 # space group number
acell 4.1790 4.1790 2.6649 angstrom # cell sides, angstrom units
angdeg 90.0 90.0 90.0 # cell angles (this is the default by the way)
znucl 14 8 # atomic number of atoms,
           # will be cross checked against pseudopotential files
natom 6 # 6 atoms in the unit cell (remember Z = 2 here)
natrd 2 # only read two atoms in, this is the asymmetric unit
ntypat 2 # two types of atoms
typat 1 2 # read atom type 1 then type 2, order is set by znucl above
xred # here come the fractional coordinates from the cif file
```

```
0 0 0 # first atom type
0.3062 0.3062 0 # second atom type
```

If I'm unsure that I've input a cell correctly, I run ABINIT with `nstep 1`, `prtgeo 2` and `prtden 1`, so that it only does one step of calculation and prints geometry and density files. I then examine the geometry output for sensible bond lengths, and look at the density file using the program XCRYSDEN to get a visual image of the cell.

In the output file from an ABINIT run, we find near the top:

```
 Symmetries : space group P4_2/m n m (#136); Bravais tP (primitive tetrag.)
================================================================================
```

Thus ABINIT found the symmetries we said were there, and is using them.

A bit further we find:

```
   acell      7.8971655093E+00  7.8971655093E+00  5.0359311715E+00 Bohr
   typat      1  2  1  2  2  2
   xred      0.0000000000E+00  0.0000000000E+00  0.0000000000E+00
             3.0620000000E-01  3.0620000000E-01  0.0000000000E+00
             5.0000000000E-01  5.0000000000E-01  5.0000000000E-01
             1.9380000000E-01  8.0620000000E-01  5.0000000000E-01
             6.9380000000E-01  6.9380000000E-01  0.0000000000E+00
             8.0620000000E-01  1.9380000000E-01  5.0000000000E-01
```

Notice that `acell` is now in atomic units, and also `xred` for all 6 atoms in the unit cell have been worked out for us. The first two are the ones we entered, the rest were computed by symmetry by ABINIT. To do this yourself, you would use the Wyckoff multiplicity of each atom from the `.cif` file together with the International Tables entry for that multiplicity in space group 136, to compute the remaining, symmetry-related atoms.

3 A Harder Case: Anatase

The anatase form of TiO_2 is a little harder. Here's the relevant parts of the `.cif` file:

```
_chemical_formula_structural 'Ti O2'
_cell_length_a 3.7842(13)
_cell_length_b 3.7842(13)
_cell_length_c 9.5146(15)
_cell_angle_alpha 90.
_cell_angle_beta 90.
_cell_angle_gamma 90.
_cell_formula_units_Z 4
_symmetry_space_group_name_H-M 'I 41/a m d S'
_symmetry_Int_Tables_number 141
```

```
loop_
_atom_site_label
_atom_site_type_symbol
_atom_site_symmetry_multiplicity
_atom_site_Wyckoff_symbol
_atom_site_fract_x
_atom_site_fract_y
_atom_site_fract_z
_atom_site_B_iso_or_equiv
_atom_site_occupancy
_atom_site_attached_hydrogens
Ti1 Ti4+ 4 a 0 0 0 0.39(6) 1. 0
O1 O2- 8 e 0 0 0.2081(2) 0.61(9) 1. 0
```

It looks at first like stishovite, with a tetragonal cell, but note the space group: number 141, $I41/amd$ is indeed tetragonal but body-centered, therefore the primitive cell will be smaller than the conventional cell. Here $Z = 4$, but that means 12 atoms in the conventional cell. Here's what the ABINIT input looks like:

```
spgroup 141 # space group number
brvltt -1 # tell abinit to find the primitive cell
acell 3.7842 3.7842 9.5146 angstrom # cell sides, angstrom units
angdeg 90.0 90.0 90.0 # cell angles (this is the default by the way)
znucl 22 8 # atomic number of atoms,
           # will be cross checked against pseudopotential files
natom 6 # 6 atoms in the primitive unit cell (although Z = 4)
natrd 2 # only read two atoms in, this is the asymmetric unit
ntypat 2 # two types of atoms
typat 1 2 # read atom type 1 then type 2, order is set by znucl above
xred # here come the fractional coordinates from the cif file
0 0 0 # first atom type
0 0 0.2081 # second atom type
```

The input is pretty much like stishovite, but with two key differences: `brvltt -1`, to tell ABINIT to find the primitive cell from the conventional cell input, and `natom 6`, which is not the conventional cell value. How did I know what value of `natom` to use? Well, since the primitive cell has to be smaller than the conventional cell in this case, but still must contain an integral multiple of formula units, the only choices were 9, 6, and 3; and primitive cells are also divisors of the total, leaving only 6 and 3 as possibilities. You can work all this out in exact detail from the symmetries but it's easy to just guess and run a trial.

Again, from the output we find:

```
 Symmetries : space group I4_1/a m d (#141); Bravais tI (body-center tetrag.)
================================================================================
```

This looks about like before–ABINIT is using the space group we wanted.

But further down, things look pretty confusing at first glance:

```
acell      1.0000000000E+00  1.0000000000E+00  1.0000000000E+00 Bohr
rprim     -3.5755508160E+00  3.5755508160E+00  8.9899941320E+00
           3.5755508160E+00 -3.5755508160E+00  8.9899941320E+00
           3.5755508160E+00  3.5755508160E+00 -8.9899941320E+00
typat      1  2  1  2  2  2
xred      0.0000000000E+00  0.0000000000E+00  0.0000000000E+00
          2.0810000000E-01  2.0810000000E-01  2.0704769534E-17
          7.5000000000E-01  2.5000000000E-01  5.0000000000E-01
          9.5810000000E-01  4.5810000000E-01  5.0000000000E-01
          5.4190000000E-01  4.1900000000E-02  5.0000000000E-01
          7.9190000000E-01  7.9190000000E-01 -2.0704769534E-17
```

The `acell` values now look very different than what we input, and also there is a new variable `rprim` that didn't appear in the stishovite case. Actually, `rprim` *was* in the stishovite case, but as its default value didn't change, it didn't get printed. This would be a good time to review the ABINIT documentation for `acell` and `rprim`. Basically, ABINIT describes the unit cell in terms of three lengths `acell` that multiply three vectors `rprim` that describe the cell axes. The default for `rprim` are the three Cartesian unit vectors, and in the stishovite case, which has a tetragonal cell, these remain valid and can be multiplied by the `acell` values. In certain complex cases like the body-centered tetragonal case, ABINIT resets `acell` to `1.0 1.0 1.0` and carries both the length and orientation of the unit cell axes completely in `rprim`. In particular, a body-centered tetragonal cell with cell lengths a, a, c is described by a primitive cell with vectors $(-a/2, a/2, c/2)$, $(a/2, -a/2, c/2)$ and $(a/2, a/2, -c/2)$. These values can be found in the `rprim` output above.

4 FAQ

How to choose the `kpt` grid? The kpt mesh has to be coherent with the symmetry of the cell. The easiest way to set it up is to let ABINIT do the work for you, using the variable `kptrlen`. With this variable, ABINIT will design a series of meshes with the correct symmetry, and then choose one with characteristics close to the input value of `kptrlen`. As a rule of thumb, `kptrlen` of 30 gives a coarse mesh, and `kptrlen` of 60 is pretty fine. `kptrlen of 80` usually gives very good convergence for many properties. As always, you should do a convergence check on the kpt grid.

For more control, do a run with `kptrlen 60` (say) and also `prtkpt 1`. In this case ABINIT will stop after computing the trial grids, and only print out information about them. Then you can pick a grid you like based on its `kptrlen value` and the number of kpts it will generate (which you might be matching against the number of processors you can run on).

Then you input this mesh using the variables `kptrlatt`, `nshiftk`, and `shiftk`.

How to choose `rprim`? As shown in the above examples, using `spgroup`, `acell`, and `angdeg` lets ABINIT do the work of setting up the cell geometry and therefore `rprim`. However, if you really want to do it yourself, you have to know how the primitive cells are constructed for the different cell types. This information can be found for instance in Bradley and Cracknell (see Further Reading below), Table 3.1

What about other settings of the space group? Some space groups have multiple settings–see the ABINIT space group help file, which you can access from the documentation for the `spgroup` variable. How do you know which one to use? The easiest way is to look at the Wyckoff multiplicities of the atoms in your structure together with their positions, and compare to the International Tables. This will give you the match for the setting used. You will then have to include `spgorig` and `spgaxor` as input variables to ABINIT.

What about forces and special positions? It is not uncommon for atoms in a unit cell to lie on "special positions", which are positions in the unit cell that exhibit the point symmetry of the space group. If the point symmetry of the position is high enough, vectorial quantities, such as force, must vanish there. A typical example is AlAs, for which both the Al and As atoms lie on special positions. When you optimize the unit cell, the ions can't move and all forces vanish at them, regardless of the volume of the cell. To optimize such a cell in ABINIT you would use `ionmov 2` and `optcell 2`, but instead of `toldff` as a convergence variable, use `tolvrs`. `toldff` refers to forces and will always be "converged" in this case, because all forces are zero.

Is there an even easier way? Yes–there is a PYTHON package called CIF2CELL, which reads `.cif` files directly and can make ABINIT input files. In my experience it works really well. So, are the above notes a waste of your time? I don't think so–first, you should understand something about what you're inputting to ABINIT, and secondly, you may be getting your structure from tabulated data in a paper, not from a `.cif` file, and then you need to understand the ABINIT input variables.

5 Further Reading

- *International Tables for Crystallography* (Wiley, 2012). Volume A contains the symmetry information for each of the 236 space groups.

- *The Mathematical Theory of Symmetry in Solids*, C. J. Bradley and A. P. Cracknell (Oxford, 2010). This book is from 1972 but was recently reissued in paperback. It is very dense and very sophisticated, but well worth having.

- *Fundamentals of Crystallography*, C. Giacovazzo *et al.* (Oxford 1992). A handbook for how crystallography is actually done, with a very thorough introduction to symmetry in solids.
- *Bilbao Crystallographic Server*: http://www.cryst.ehu.es. An incredibly helpful website. If you've ever wondered how to visualize the Brillouin Zone for some unusual space group, or how to find the special points and lines in reciprocal space, this is the place to start.
- *ICSD Database*. A non-free database of `.cif` files. Very comprehensive, easy to use, and well worth the subscription fee.