AIRSS Manual

Yang Li
lyang.1915@gmail.com

2020.10.24 — 2020.10.28

Contents

1 About AIRSS

AIRSS(**A**b **I**nitio **R**andom **S**tructure **S**earching) is a first-principles structure searching software developed by professor Chris Pickard at Cambridge University. It is an open-source software protected by the **GPL2 license**. You can visit its official website, `https://www.mtg.msm.cam.ac.uk/Codes/AIRSS`, to get the source package.

The so-called "structure searching" refers to such a process: for a system with unknown atomic structure but under certain physical conditions (such as atomic distance, distribution density, element composition, element ratio, etc.), we widely guessed its configuration, relax them using some DFT software, calculating its energy, and finally obtaining the most stable atomic structure. Obviously, the manual guessing or blind traversal search is extremely awkward, time-consuming, and even difficult to achieve. Therefore, it is necessary to use a set of mature structure search software, systematically and cleverly capture the stable configuration of the system.

AIRSS is exactly such kind of software! Professor Chris's introduction about this software can refer to the relevant video. There are also another two more commonly used structure searching software: USPEX and CALYPSO.

Unlike the genetic algorithm used in USPEX or CALYPSO, AIRSS is based on a completely random structure searching strategy, and the different structures it produces are completely random and independent. Such an algorithm is very conducive to the parallel implementation of search tasks.[1] The effectiveness of this random method is also discussed in Professor Chris' articles on AIRSS.[2] The most powerful feature of this random method of AIRSS lies in its many flexible adjustable parameters, which can be said to be the most outstanding feature of this project. The user experience of AIRSS gives a feeling of "point-and-shoot camera vs professional SLR camera". The former is simple and practical, you can submit various tasks with one click and get some results not bad. The latter has many buttons, adjustable parameters, and can be customized to produce a variety of more professional and refined results. The former sacrifices flexibility for ease of operation; the latter is the opposite. AIRSS can highly customize the characteristics of research objects according to user needs. For example, we can even use this software to verify the stability of an atomic structure, or to calculate the potential energy of the lattice surface.

Regrettably, as a powerful structure searching software, AIRSS's official manual is still to be developed. In addition, the relevant documentation or tutorials on the Internet are also very scarce, so most people are not familiar with such an extremely excellent program in the material design.

In order to make up for this shortcoming, the author decided to write this article. The following content is not a guide or tutorial, but only some records of learning AIRSS. Most of the conclusions of this article are derived from analyzing the source code, actually running the program, and trying to fumble. [3] **Limited by various factors, the errors in understanding and interpretation may be inevitable. If you find any questions or errors, please feel free to correct me.**

[1] For example, we can use AIRSS to randomly generate 1000 structures, and then relax them independently

[2] [1] PRL 97, 045504 (2006); [2] JPCM 23, 053201 (2011)

[3] Although the official does not give a complete user manual, it does provide a large number of usage examples, located in the example directory of the program, for users to learn from.

2 Prepare Work

2.1 Linux Operating System

Before reading this record, you need to have a certain understanding of the Linux operating system. For example, to understand the meaning of the following commands:

```
user@machine_name$ ls | grep *.cell
```

And catastrophic accidents that can be caused by the following instructions:

```
root@machine_name# rm -rf / home/user_name/trash_directory
```

Another thing that needs special reminder is that certain characters will be converted into symbols that are not recognized by the command line when the PDF document is compiled (such as the 'minus' in ls −1, although they look extremely the same as the 'minus' we typed in the command line). **Please be careful when directly copying commands or characters in this document.**

2.2 Program Installation and Remove

Appendix A takes `airss-0.9.0` as an example to briefly record the installation process of AIRSS. The latest `airss-0.9.1` has made a lot of simplifications in the installation process, you can read the included README file to install this version.

The main running script of AIRSS is written in perl language, and can only be installed in the `*nix` system, used in the command line (Command Line). Before installing this software, you'd better understand the usage of GNU make.

2.3 Run AIRSS

For you first time using AIRSS, you can enter the command '`airss.pl`' in terminal to view the software welcome interface.

```
user@machine_name$ airss.pl

      .o.       ooooo oooooooooo.    .oooooo..o  .oooooo..o
     .888.      '888' '888   'Y88. d8P'    'Y8 d8P'    'Y8
    .8:888.      888   888   .d88' Y88bo.      Y88bo.
   .8' '888.     888   888ooo88P'   ':Y8888o.   ':Y8888o.
  .88ooo8888.    888   888'88b.         ':Y88b      ':Y88b
 .8'     '888.   888   888  '88b.  oo     .d8P oo     .d8P
o88o     o8888o o888o o888o  o888o 8::88888P'  8::88888P'

     Ab Initio Random Structure Searching
     Chris J. Pickard   (cjp20@cam.ac.uk)
            Copyright (c) 2005-2018

Please cite the following:

[1] C.J. Pickard and R.J. Needs, PRL 97, 045504 (2006)
[2] C.J. Pickard and R.J. Needs, JPCM 23, 053201 (2011)

Usage: airss.pl [-pressure] [-build] [-pp0] [-pp3] [-gulp]
                [-lammps] [-gap] [-psi4] [-cluster] [-slab]
                [-dos] [-workdir] [-max] [-num] [-amp] [-mode]
                [-minmode] [-sim] [-symm] [-nosymm] [-mpinp]
                [-steps] [-best] [-track] [-keep] [-seed]
  -pressure f  Pressure (0.0)
  -build       Build str. only (false)
  -pp0         Use pair potentials rather than Castep (0D) (false)
  -pp3         Use pair potentials rather than Castep (3D) (false)
  -gulp        Use gulp rather than Castep (false)
  -lammps      Use LAMMPS rather than Castep (false)
  -gap         Use GAP through QUIP/QUIPPY/ASE (false)
  -ps4         Use psi4 (false)
  -vasp        Use VASP (false)
  -cluster     Use cluster settings for symmetry finder (false)
  -slab        Use slab settings (false)
  -dos         Calculate DOS at Ef (false)
  -workdir  s  Work directory ('.')
  -max      n  Maximum number of str. (1000000)
  -num      n  Number of trials (0)
  -amp      f  Amplitude of move (-1.5)
  -mode        Choose moves based on low lying vmodes (false)
  -minmode  n  Lowest mode (4)
  -sim      f  Threshold for structure similarity (0.0)
  -symm     f  Symmetrise on-the-fly (0.0)
  -nosymm   f  No symmetry (0)
  -mpinp    n  Number of cores per mpi Castep (0)
  -steps    n  Max number of geometry optimisation steps (400)
  -best        Only keep the best str. for each compos. (false)
  -track       Keep the track of good str. during RESH(false)
  -keep        Keep intermediate files (false)
  -seed     s  Seedname ('NONE')
user@machine_name$
```

`airss.pl` is the main command to perform structure search using AIRSS. The usage of this command has been systematically and briefly explained in the welcome interface. The table in the usage explanation has three columns. The first column is the name of the incoming parameter; the second is The data type of the incoming parameter, '`f`' represents a floating point number, '`n`' represents an integer, '`s`' represents a string, and "empty" represents a logical string `true` or `false`. The third column is a brief description of the corresponding parameter.

3 Custom Random Structure

The core component of AIRSS is named **buildcell**. The function of this component is to generate a series of initial atomic configurations with random structures but meeting the constraints of given physical conditions based on the *.cell file given by the user. In addition, we can also take out this module separately to adapt it to other programs (such as VASP). Learning to write highly customized *.cell files is the foundation of learning AIRSS.

In order to facilitate the subsequent description, we first briefly introduce the basic logic of the buildcell component.

The execution of the buildcell block is roughly divided into the following steps:

1. Read the configuration information in the *.cell file through the cell.f90 module.
2. Generate a structure that meets the requirements through build.f90, opt.f90 etc.
   (a) First determine the reasonable lattice constant according to the given atom and its radius, if there is a lattice tag[4] #FIX, ignore this step.
   (b) Under the premise of meeting the requirements, generated an atom at a random position.
   (c) Choose according to constraints: accept this position, reject this position, or **PUSH** the current atom. **PUSH** means: move two atoms that are too close to each other in the direction of their connection, and move them away Same distance. When there is FIX or NOMOVE in the atom label, ignore this operation.
   (d) Traverse all the atoms in the system until the complete structure is updated.
   (e) Depending on the user's setting, use the phenomenological pair potential (pp3) to simply relax the structure.
3. Output the above random structure that meets the requirements through the buildcell.f90 module.

3.1 File Structure of *.cell

The *.cell file is the "seed file" of the structure search. You can set search constraints in this file. The asterisk (*) before the file name represents a wildcard under the Linux system, that is, it represents any strings. In the AIRSS file system, we refer to the part before the suffix as the "seed name". Seed name can be used to distinguish the same kind of files of different calculations. All wildcards will be seen later in the text. Unless otherwise specified, they all refer to the seed name.

***.cell file has the following characteristics:**

1. Since AIRSS was written by some people who participated in the development of CASTEP, the program is extremely friendly to CASTEP, and the *.cell file here is fully compatible with the *.cell file in CASTEP.
2. When setting parameters in the *.cell file, you need to use certain keywords to indicate the meaning of the set parameters.
3. AIRSS completely reuses the keywords of CASTEP for structural declaration, such as LATTICE_CART, etc.
4. The built-in keywords of AIRSS usually start with #, such as #RASH. (To make the *.cell file of AIRSS fully compatible with CASTEP)
5. *.cell file is mainly composed of two parts: "structure data" and "global parameters". Correspondingly, keywords in the file can also be divided into the above two categories.
6. There is no restriction on the order of writing between any keywords.
7. All keywords set in *.cell need to be written in **capital letters**, and be careful **not to add spaces** between keywords, equal signs and parameter values. At the same time, any blank lines will be automatically ignored.
8. This file uses double pound signs (##) to indicate the content of the comments.
9. Although the program allows multiple keywords to appear in a single line, it is best to write only one keyword per line for the purpose of standardization and readability of the written file.

Next, we will introduce the specific writing method of *.cell file. Whether you can use AIRSS independently and efficiently depends on whether you can master the content of the following pages. Compared with those "about AIRSS", "no one will read it carefully" 'Foreword chapter, here can be said to be the core position of the article, so I want to use this "golden rank" to discuss some issues of the rationality of the AIRSS program design, which is quite beneficial for the better use of the software.

The problem that the structure search wants to solve is actually quite well-defined. We imagine a N-dimensional phase space, where different dimensions of the space represent different variable parameters of the structure (such as atomic ratio, lattice basis vector, atomic position, etc.)), each point in the space represents a different atomic configuration, and each configuration corresponds to an energy value. So there will be a N+1-dimensional energy surface in our system (such as our phase space If it is two-dimensional, then there will be a three-dimensional surface about energy). What the structure search does is to find some/all energy minimum points of this energy surface, and then determine the structure corresponding to the global minimum energy (That is what we call the most stable configuration).

One way to solve this problem is to use some very clever algorithms (such as genetic algorithms or machine learning) to make the system "evolve" based on different metastable configurations, and find the most stable configuration globally. The implementation steps of the algorithm are as follows: Initially generate a series of random mechanism types; then use first-principle calculation software to relax

these configurations, inherit/learn the characteristics of low-energy structures from the relaxation results; then use the information learned from the previous generation Generate a next-generation structure; iterate in this way, and finally expect to find a global most stable state. The advantage of this algorithm is that it can quickly locate the energy minimum point in a large phase space, which does not rely on any prior information , It is only a purely theoretical prediction of a certain structure is very effective. But at the same time, due to the causal relationship between the previous generation and the next generation structure, the structure of different generations can only be calculated in sequence (serial). In addition, use This type of algorithm will also face a big challenge, that is, the so-called "phase space collapse" (that is, due to the limitations of the initial selection with the organization type, the subsequent new structures are all localized in certain sub-regions. Near steady state). In order to solve this problem, we have to introduce an adjustable parameter: "mutation probability" (that is, it is believed that the structure inherits the parent structure, and there is a certain probability that it will change in certain positions). But in some Under circumstances (such as setting the mutation probability too high or too low), this parameter is not as effective as we thought.

AIRSS uses another structure searching strategy. Its *.cell file provides a large number of adjustable parameters, and each parameter corresponds to a number of constraints, which can achieve **precise restrictions on the phase space**. The system is meeting these The initial configuration is **randomly** set in the phase space of the constraints, and then a (meta)stable state near the initial configuration is found through the relaxation of the first-principles calculation software, which is regarded as a "structure search". Through large-scale (such as thousands or tens of thousands of times) "search", combined with the **structural duplicate check algorithm**, we can finally find a global most stable state in the constrained phase space. Such a steady state probably has two characteristics: **the energy is lower (the lowest)**, and **the number of repetitions is more (the most)**.

The strategy of "random selection" of the initial structure at first sounds unreasonable and extremely inefficient, but if the phase space obtained after "precision constraints" is not very large, then even if it is not used, it is very complicated The genetic algorithm, which relies solely on simple random dots, can also obtain good results. At the same time, the random setting of the initial structure ensures the **efficiency of parallel computing** while avoiding the occurrence of "phase space collapse".

Perhaps seeing this, you will argue that if we set the first generation in the genetic algorithm as many as in the AIRSS, such as 10,000, the final result is not much better than the AIRSS algorithm, after all, AIRSS only It is equivalent to the first step of the genetic algorithm? But the fact is that in a small phase space, we use 10,000 configurations per generation to iterate for 100 generations, and directly use 1,000,000 random structure relaxation calculations. The difference is actually Not big, and sometimes the latter is more accurate. And just saying "10,000 configuration iterations for 100 generations per generation" would feel ridiculous, because this kind of calculation requires massive computing power. Generally speaking. In other words, we will take the number of configurations in each generation on the order of 100. So I prefer to think that these two algorithms are complementary, and one is suitable for serially finding some low-energy extreme points in a super large phase space. , The other is suitable for finding all extreme points in a small phase space that is tightly restricted in parallel. And according to my experience, the performance of AIRSS in a larger phase space is not as bad as we thought Be-

cause for the phase space that actually meets the physical constraints, there are a small number of extreme points (otherwise the structure may have a large number of metastable states in the experiment), and it can completely rely on the relaxation process calculated by first principles. Find these final extreme points from an initial point that looks random but is actually within the range of DFT relaxation ability.

Using a strategy search structure that **accurately limits the size of the phase space** plus **random search** can be said to be a very strange but effective idea. Because most of the time, we do not need to predict a structure completely theoretically, but cooperate with experiments (such as Experiments have already known the element ratio and crystal shape, and sometimes even the approximate atomic position can be known through electron microscopy), and jointly study the atomic structure of a certain material. At this time, the variable phase space of atoms may not be very large, so "How to define the phase space under study more precisely" has become a more important issue than "how to optimize the algorithm to get the minimum point faster". Of course, the size of the phase space is also relative. For example, if we choose 10 random organization types, it is obviously not possible to cover any so-called "small phase space".Only when this number increases to, for example, 10,000, can we reluctantly say that for some systems. But here comes another benefit of randomly generating structures, that is, it is inherently suitable for parallel computing. We know that almost all the time in structure search is spent on structure relaxation. If we have 8000 cores in our hands, use 8 Each core optimizes a configuration, then at the same time we can search for 1000 structures at the same time. If there are 80,000 cores (of course, this is almost impossible, face smile.jpg), then 10,000 structures can be calculated at the same time This simple and rude and efficient parallel method is unmatched by other algorithms. AIRSS also combines the two characteristics of **precise limit phase space size** and **random structure generation**. In many cases (especially when working with experiments to search for atomic structures), It is very efficient.

At present, the functions of the parameters available in the *.cell file can be divided into the following categories:

1. **Declare the initial lattice structure.** For example, declare the initial unit cell basis vector, initial atomic position, etc.
2. **Constraints on unit cell parameters.** Such as fixing the initial lattice base vector, constraining the unit cell system, constraining the degree of change of the unit cell, constraining the crystal system of the unit cell, adding a vacuum layer to the unit cell, whether to expand the cell, etc.
3. **Requirements for the chemical formula of the system.** For example, how many yuan the system is, whether the electrons can be balanced (whether there are dangling bonds), etc.
4. **Restrictions on the position of atoms.** Such as the shortest distance between two atoms, the maximum/minimum distance of an atom from the initial position, the coordination number of an atom, etc.
5. **Restrictions on the symmetry of the system.** For example, the system has several symmetry operations, in what space group, the degree of fineness of symmetry search, whether to introduce symmetry breaking disturbances, etc.
6. **Adjustment to the search algorithm of the program itself.** For example, accept the upper limit of the number of search failures, whether to introduce PSUH, whether to introduce TPSD, whether to introduce RASH, etc.
7. (Less commonly used) Add a force field covering the entire unit cell in the crystal. For example, add a spherical force field, an ellipsoidal force field, a strip force field, a plane force field, etc.

The following is the `Al.cell` file used to predict the structure of metal aluminum:

```
0  user@machine_name$ cat Al.cell
1  %BLOCK LATTICE_CART
2  2 0 0
3  0 2 0
4  0 0 2
5  %ENDBLOCK LATTICE_CART
6
7  %BLOCK POSITIONS_FRAC
8  Al 0.0 0.0 0.0 # Al1 % NUM=8
9  %ENDBLOCK POSITIONS_FRAC
10
11 #MINSEP=1.5
12 user@machine_name$
```

The structure of the above *.cell file can be summarized as follows:

1. The first 9 lines are data reading blocks defined by the format %BLOCK [keywords], this format is very common in CASTEP. These blocks define the basic **structure data** of the crystal.
2. The [keywords] of the 1 to 5 rows of data is LATTICE_CART, which declares the unit cell base vector defined by the Cartesian absolute coordinate system.
3. The [keywords] used in lines 7 to 9 is POSITIONS_FRAC. The meaning of this in CASTEP is "the position of the atom defined in fractional coordinates". In AIRSS, the data module can not only specify the initial position of the atom, but also Define the constraint conditions for a single atom in the search process. If the specific position of the atom is not known initially, it can be set to any value, such as (0, 0, 0), and the program will find the atomic position that meets the requirements. Atom position More specific details of the settings will be given later.
4. #MINSEP on line 11 is **global parameter** in AIRSS. #MINSEP=1.5 indicates that the distance between any two atoms must not be less than 1.5 $\mathring{(}$A).

As we can see, the structure of the *.cell file is roughly divided into two parts: **structure data** and **global parameters**.

3.2 Parameter Details of *.cell File

3.2.1 structure data

This part follows the data pattern defined in the CASTEP structure file. The structure data consists of two parts: the lattice parameter data block and the atomic position data block.

The specific mode of the data block is as follows:

```
%BLOCK [keywords]
[...]
[structure data]
[...]
%BLOCKEND [keywords]
```

The data block keywords that can be used in AIRSS are listed in **Table 1**.

Table 1: AIRSS Cheat Sheet – Data BOLCK

Parameters	Description
LATTICE_CART	Use the Cartesian coordinate system to define the base vector of **unit cell** with a matrix of 3×3. By appending the following tags to the block: #FIX,#CFIX, #ABFIX, can be implemented separately: fix the entire lattice shape, fix the lattice constant c, and fix the lattice constant ab. In addition, it should be emphasized that if you want to ensure that the lattice parameters are unchanged, except in the lattice Add #FIX to the parameter data block to ensure that it is maintained as a constant value when guessing the lattice structure, and it is also necessary to make corresponding settings in the DFT relaxation software to ensure that it is also in the structure relaxation. constant.
LATTICE_ABC	Use a 2×3 matrix to define the lattice parameters of the unit cell. The first row of the matrix is the parameter abc, and the second row is the three angles. In the same .cell, it is the same as |LATTICE_CART|Chooseone of the two. By adding the following tags to the block: #FIX,#CFIX,#ABFIX, can be achieved separately: fix the entire lattice shape, Fixed lattice constant c, fixed lattice constant ab.
POSITIONS_FRAC	The atomic position is defined by the fractional coordinates in the lattice base vector coordinate system.
POSITIONS_ABS	Define the atomic position with absolute numerical coordinates in Cartesian coordinate system. Choose one from this or POSITIONS_FRAC.
SYMMETRY_OPS	Define the symmetry operations that the generated unit cell must meet, a group of four, the first three lines define rotation operations, and the fourth line defines translation operations. For specific usage, please refer to CASTEP:SYMMETRY OPS.

The following briefly introduces the detailed writing method of **lattice param-**

eter data block and **atomic position** data block.

Lattice parameter First introduce the setting of lattice parameters, take `LATTICE_CART` as an example.
Example 1.

```
%BLOCK LATTICE_CART
20 0 0
0 20 0
0 0 20
#FIX
%BLOCKEND LATTICE_CART
```

The above fields construct a cube of $20 \times 20 \times 20$ $(\mathring{A})^3$ as the unit cell of the crystal. Here `#FIX` is called **lattice tag**.

It states that the lattice constant cannot be changed during the search (generating the initial random structure). If you want to ensure that the lattice constant of the system remains unchanged during DFT relaxation, you need to set it in the input file of the corresponding calculation software .

Atomic position Then introduce the setting method of atomic position, take `POSITIONS_FRAC` as an example.
Example 2.

```
%BLOCK POSITIONS_FRAC
Al   0.0 0.0 0.0 # Al1 % NUM=2
Mg   0.0 0.0 0.0 # Mg1 % NUM=4
O    0.4 0.2 0.3 # O1  % NUM=1 POSAMP=0 FIX
O    0.1 0.1 0.1 # O2  % NUM=1 POSAMP=0 UNMOVE
H    0.3 0.3 0.6 # free_H
H    0.0 0.0 0.0 # H-set % ANGAMP=0 POSAMP=0
H    0.0 0.0 0.0 # H-set % ANGAMP=0 POSAMP=0
%BLOCKEND POSITIONS_FRAC
```

As can be seen from this example, the basic format of atomic position structure data is:

```
[element] [x] [y] [z] # [atoms_set_name] % [tag1] [tag2] [tag3]
```

每一行内部的元素含义如下:

1. 第一项是元素名称, 如果**将元素名称设置为 Z**, 则表示此原子是个空位, 空位与真实原子一样具有体积和 PUSH 属性, 但不会在最终结果中输出.
2. 二三四项是原子位置坐标.
3. 井号 (#) 后的第一项是原子所在原子集的名称, 原子集名称可以被设置成任意字符. **原子集名称相同的原子组成一个原子集, 原子集是由一个或多个原子构成的基本结构单元, 同一个原子集中的原子做相同的随机移动, 且不受 PUSH 影响**. 如无特殊需求, 为了实现自由度相对最大化, 一般不推荐将不同原子放入同一原子集中.
4. 百分号 (%) 之后的内容都是**原子标签** (Atom Tags). 同一行原子可以指定多个原子标签, 中间用空格隔开, 他们共同指定了该原子应该满足的若干约束条件.

Table 2 是 AIRSS 中全部可用的原子标签的具体说明.

Table 2: AIRSS Cheat Sheet – Atom Tags (distance in Å)

参数名称	输入类型	默认值	功能说明
NUM=n NUM=$n_{min} - n_{max}$	int	1	定义该行原子实际代表的原子个数. 使用 NUM 定义的重复原子会被逐个单独分配到不同的**原子集**中.
POSAMP=d	float	-1.0	定义该行原子在结构搜索过程中, 能偏离初始位置的最大距离. 若为负值则无限制.
MINAMP=d	float	0.0	定义该行原子在结构搜索过程中, 偏离初始位置的最小距离.
XAMP=d	float	-1.0	定义该行原子在结构搜索过程中, 在 X 轴方向上移动的最大距离. 若为负值则无限制. 设置该项会令 POSAMP 和 MINAMP 失效
YAMP=d	float	-1.0	定义该行原子在结构搜索过程中, 在 Y 轴方向上移动的最大距离. 若为负值则无限制. 设置该项会令 POSAMP 和 MINAMP 失效.
ZAMP=d	float	-1.0	定义该行原子在结构搜索过程中, 在 Z 轴方向上移动的最大距离. 若为负值则无限制. 设置该项会令 POSAMP 和 MINAMP 失效.
ANGAMP=θ	float [0, 360]	-1.0	原子绕自身所在**原子集**中心旋转角度的最大值 (与 POSAMP 互相独立, 分开作用). 存在晶格标签 #FIX 时, ANGAMP=0, POSAMP=0 与 NOMOVE(\FIX) 共用, 可保证原子在原始位置不动. 如果 cell 文件中所有原子集都只有 1 个原子, 则该参数失效, ANGAMP 强制设为 0. 若为负值则无限制.
RAD=d	float	0.0	设置原子的半径, 用于判断两个原子间的最小距离, MINSEP. 其优先级低于直接设置全局参数 #MINSEP=.

Continued on next page...

Table 2 – *Continued from previous page*

参数名称	输入类型	默认值	功能说明
FIX	void	off	该原子在产生结构时不受 PUSH 影响. 任何接近该原子的可移动原子都会以 2 倍的 PUSH 步长被 PUSH 回去. 同时向 cell 文件中写入指令使其在 CASTEP 弛豫时也不动. 这一指令仅在晶格被 #FIX 时有效. 原子标签中的 FIX 与晶格标签中的 #FIX, #CFIX, #ABFIX 名称相似, 但作用不同.
NOMOVE	void	off	该原子在产生结构时不受 PUSH 影响, 任何接近该原子的可移动原子都会以 2 倍的 PUSH 步长被 PUSH 回去. 这一指令仅在晶格被 #FIX 时有效.
COORD=n, COORD=$n_{min} - n_{max}$	int	-1	约束该行原子的配位数 (最近邻原子数). 若为负值则无限制.
NN=±elm	string	null	规定原子最近邻元素种类, '+' 代表必须近邻该元素, '−' 代表不能近邻该元素. 一个原子只能指定一种元素近邻或不近邻 (NN 参数仅会被读入一次). 若为空, 则表示无限制
OCC=p	float [0, 1]	1.0	该点位占据原子的几率, 同种元素在不同位置的占据几率之和必须为大于 0 的整数. 接收分数形式的输入, 如 OCC=1/3. 若该值设为小于 0 的数, 则强行将该原子的 OCC 设置为 1, 同时不再输出该原子可能因对称性衍生的其他原子位置. 若原子同时有 FIX 或 NOMOVE 标签, 则 OCC 强制设为 1. 若该原子 OCC 大于 1, 则将其设置为空位 (与将元素名称设为 Z 的效果相同). 系统存在对称性时, 使用更小的 OCC 可强制某一原子位于对称性更高的点位上. 但更为自然的用法是使用 MULT 代替 OCC 设置. 若设置了全局参数#NFORM=n(n 大于 0): 在没有对称性条件下, OCC 参数失效; 在有对称性条件下, 请参考 MULT 中的说明.

Continued on next page...

Table 2 – *Continued from previous page*

参数名称	输入类型	默认值	功能说明
MULT=m	float	-1.0	设置该原子点位的 multiplicities. 这一数值为正时, 强制将 OCC 参数的值覆盖为 MULT/SYMM_NUM. 其中 SYMM_NUM 为晶格对称群元的个数. 若设置了全局参数#NFORM=n(n 大于 0), 则等效于将所有原子的 MULT 的值设为 n, 同时不设置#NFORM=. 若该行原子有标签 FIX 或 NOMOVE, 则强制将该原子的 MULT 设为-1. 使用这一参数, 可以限制某一行原子由晶体对称性实际生成原子的个数, 以及所处位置对称性的高低. 关于格点 multiplicities 与对称性的关系可参考 BCS: Wyckoff Position.
PERM	void	off	在完成原子位置选定后, 重排 (按一定概率互相交换) 指定原子位置间的元素种类. 需要配合全局参数 #PERMUTE 使用, 否则该参数将被关闭.
ADATOM	void	off	表明该行原子是等, 未加该标签的所有原子位置生成结束之后, 再加入的原子.
ATHOLE	void	off	表明该原子位于被挖掉的孔洞之中, 输出结构中将自动删除该原子. 常与全局变量 #HOLE=, #HOLEPOS= 联用.

3.2.2 全局参数

结构搜索过程中整个体系应遵守的条件由全局参数指定. AIRSS 中的全局参数均以井号（#）开头, 对体系中晶格以及全部的原子作用. 另外, 有些全局参数和原子标签有相同的作用, 此时, 原子标签设置的优先级要高于全局参数.

AIRSS 中全部可用的全局参数的功能和详细使用方法如 **Table 3** 所示.

Table 3: AIRSS Cheat Sheet – Global Pragma (distance in Å)

参数名称	输入类型	默认值	功能说明
#CELLCON=a b c α β γ	int	off	定义晶胞应满足的条件: a, b, c 是晶格常数; α, β, γ 是三个晶角. 晶格常数可选填的值有 $\{0, -1\}$, 晶角可选填的值有 $\{0, -1, \theta\}$. 0 表示无约束, −1 表示相等. 如 #CELLCON=-1-1 -1 90 90 90 表示立方晶系.
#SYSTEM=sys	string	off	设置结构所在的晶系, 可选的输入有 [Tric, Mono, Hexa, Rhom(/Trig), Orth, Tetr, Cubi]. 该参数实际为 '#CELLCON=' 参数的若干特殊组合.
#CONS=p	float $(0, 1]$	0.4	约束单胞边长. 接近 0 表示完全没有约束, 等于 1 表示尽一切可能约束晶胞 abc 三边等长.
#ACONS=p	float $(0, 1]$	0.5	约束晶胞晶角, 使其远离某一晶格参数极小、晶胞扁平的情况. 接近 0 表示完全没有约束, 等于 1 表示尽一切可能约束三个键角严格等于 90 度.
#CELLAMP=p	float	-1.0	在一定限度内, 随机变化晶格形状. 设为负值时采用晶格形状将完全随机变化. p 越小, 晶格偏离 cell 文件中给定值越小. 若要启用该参数, 建议不要设置超过 1.0 的值.[5] 使用该参数后, 晶格标签 #ABFIX 与 #CFIX, 原子标签 #CONS 与 #ACONS 都将被弃用. 若为负数则关闭该参数设置.
#VACUUM=d	float	0.0	在晶格 (Z 方向) 中加入 d Å 的真空层.
#NFORM=n	int	-1	声明单胞中实际存在的原子个数是的 %BLOCK POSITIONS_* 或 #SPECIES 中定义的 n 倍. 该选项不可与 #NATOM 联用. 若为负值则此参数被关闭.

Continued on next page...

[5] 该参数使用方法详见 airss-0.9.1/src/buildcell/src/build.f90 第 100 行 cellamp 变量的使用.

Table 3 – *Continued from previous page*

参数名称	输入类型	默认值	功能说明
#SUPERCELL=n #SUPERCELL=n_a n_b n_c #SUPERCELL=a_x a_y a_z b_x b_y b_z c_x c_y c_z	int	off	定义超胞的尺寸. 可以使用超胞中单胞的的个数 n, 超胞晶格基矢在三个方向的数值 n_a n_b n_c, 或是超胞与单胞晶格基矢之间的变换矩阵 $[[a_x, a_y, a_z], [b_x, b_y, b_z], [c_x, c_y, c_z]]$ 定义新的超胞. 超胞的构建策略是: 首先用单胞构建符合条件的原子构型, 而后根据扩胞参数复制粘贴得到超胞后直接输出.
#SLAB	void	off	声明该结构是一个二维平面结构, 取超胞时将不再向 c 方向扩胞, 同时检查结构对称性时也将忽略 c 方向.
#SURFACE	void	off	声明该结构是一个表面结构, 表面结构继承 SLAB 的全部特性, 同时增加一条与表面相关的对称性限制条件.[6]
#MOLECULES	void	off	声明同一个原子集中的原子构成了一个基本分子构型, 检查配位时将不再对同一个原子集内的原子做检查.
#CLUSTER	void	off	声明预测的结构是无周期性的团簇.
#FOCUS=n	int	0	约束最终结构必须是 n 组分的 (由 n 种不同的元素构成). 该参数一般用于变组分分析. 小于等于 0 表示无约束.
#OCTET	void	off	检查化学配比是否电子配平 (全部电子数是否可以被 8 整除).
#POSAMP=d	float	-1.0	与原子标签中的定义相同, 结构搜索中原子偏离初始点的最大位置. 负值为无限制.
#MINAMP=d	float	-1.0	与原子标签中的定义相同, 结构搜索中原子偏离初始点的最小位置. 负值为无限制.

Continued on next page...

[6] 见源码 airss-0.9.1/src/buildcell/src/cell.90 第 3216 行

Table 3 – *Continued from previous page*

参数名称	输入类型	默认值	功能说明
#XAMP=d	float	-1.0	与原子标签中的定义相同, 结构搜索中原子 X 方向偏离初始点的最大振幅. 负值为无限制. 设置该项会令 POSAMP 和 MINAMP 失效.
#YAMP=d	float	-1.0	与原子标签中的定义相同, 结构搜索中原子 Y 方向偏离初始点的最大振幅. 负值为无限制. 设置该项会令 POSAMP 和 MINAMP 失效.
#ZAMP=d	float	-1.0	与原子标签中的定义相同, 结构搜索中原子 Z 方向偏离初始点的最大振幅. 负值为无限制. 设置该项会令 POSAMP 和 MINAMP 失效.
#ANGAMP=θ	float $[0,360]$	-1.0	与原子标签中的定义相同, 原子绕自身所在**原子集**中心旋转角度的最大值. 负值为无限制.
#MINBANGLE=θ	float $(0,360]$	0.0	设置搜索结构中, 原子键角的最小值.
#MAXBANGLE=θ	float $(0,360]$	180.0	设置搜索结构中, 原子键角的最大值.
#MINSEP=d #MINSEP=d X-X=d_{XX} X-Y=d_{XY} ...	float	0.0	两原子间最小距离, 也可以用来定义两原子距离固定是多少. 比如 #MINSEP= 2.0 Li-Li=2.6 Ge-Ge=2.51
#RAD=d	float	0.0	与原子标签中的定义相同, 定义原子的半径大小.
#COORD=n	int	-1	与原子标签中的定义相同, 设置原子的配位数限制. 负值为无限制.

Continued on next page...

Table 3 – *Continued from previous page*

参数名称	输入类型	默认值	功能说明
#SPIN=S_{real} S_{mod} #SPIN=S_{real} S_{mod} "elm_1 elm_2... "	float, string	off	随机设置体系每个原子上共线自旋的取值. 并要求 $S_{t,real}/N_{ions} = S_{real}$, $S_{t,mod}/N_{ions} = S_{mod}$, 其中 $S_{t,real}$ 是全部指定原子自旋的和, $S_{t,mod}$ 是全部指定原子自旋绝对值的和. 未被指定的原子自旋保持为 0. 可以用被双引号包裹的带空格的元素符号指明哪些是指定原子. 若未指定任何原子, 则默认所有原子都是指定原子. 例如 #SPIN=0 5 "Fe Co " 或者 #SPIN=1 4
#SPECIES=elm_1%$tags_1$, elm_2%$tags_2$, ... #SPECIES=elm_1, elm_2...	string	off	使用简化记号定义体系原子组分. 例如, #SPECIES=Si%NUM=1 COORD=2,O%NUM=2. 该参数不能与BLOCK POSITIIONS_* 同时出现.
#NATOM=n #NATOM=$n_{min} - n_{max}$	0	int	与 #SPECIES 联用, 定义单胞中总原子数, 一般用于变组分分析 (变胞预测). 且使用此全局参数会使 #SPECIES 指令包含的% 后的原子标签全部失效. [7] 若 #SPECIES 中含有多中元素, 则每种元素随机数目, 保持总和为 NATOM. 该参数设为 0 时自动失效.
#TARGVOL=V #TARGVOL=$V_{min} - V_{max}$	float	init. cell vol.	固定晶格体积为 V, 或 V_{min} 到 V_{max} 之间的随机数值. 存在晶格标签 #FIX 时该参数失效. 默认为初始给定原胞的体积. 该参数常在不引入 BLOCK POSITION_* 区块时使用.
#VARVOL=V	float	init. cell vol.	与 #TARGVOL= 作用相同, 该参数会覆盖 #TARGVOL= 的设置. 默认为初始给定原胞的体积.
#SLACK=p	float $[0, 1)$	0.0	使用此参数可整体降低体系对原子间成键 (原子相对位置) 的限制, p 越大对原子间间距和角度的要求越低. 默认为 0, 推荐值 0.1–0.3.

Continued on next page...

[7] 详细原因请参见源码 “airss-0.9.1/src/buildcell/src/cell.f90” 第 493 行与 514 行区别.

Table 3 – *Continued from previous page*

参数名称	输入类型	默认值	功能说明
#AUTOSLACK=p	float $[0,1)$	off	使用此参数可整体降低体系对原子间成键 (原子相对位置) 的限制, p 越大对原子间间距和角度的要求越低. 若给定的初始值 p 不合适, 则以 0.01 的步长递增 SLACK, 直到找到合适的 SLACK 值.
#FLIP	void	off	搜索结构时, 对**原子集**引入随机翻转操作. 若体系中所有原子集中均只有一个原子, 则该参数失效.
#REMOVE	void	off	删除 (PUSH 后) 重叠的原子 (之一). 可以用于高初始原子密度的结构预测.
#TIGHT	void	off	使得生成的结构更加紧密.
#SYMMOPS=n #SYMMOPS=~n #SYMMOPS=$n_{min}-n_{max}$	int	off	声明生成的结构中必须含有 n 种对称操作. 若体系是周期性晶体结构, 推荐从下述整数中选取: 1,2,3,4,5,6,8,12,16,24,48. 若输入中含有波浪号 (~) 则表明, 结构搜索时将只在 general positions 上放置原子, 对于对称性更高但数量[8]更少的 special position 不予考虑.[9] 另外需要注意的是, AIRSS 中实现锁定对称性的方法是: 接将原子按对称性复制 n 份. 例如原子有 C_4 对称性, 则最终输出的总原子数将是设定的 4 倍. 但如果结合原子标签 OCC, MULT 等的使用, 可实现在更高对称的点位安放更少原子的操作. 在含有波浪号 (~) 的模式下不推荐使用MULT 等参数限制原子对称复制的个数.

Continued on next page...

[8]这里的 "数量" 是指, 满足对称操作所需的最少同类原子数量, 也即该点位的 multiplicities.
[9]源代码中这样描述: Symmetry is only be approximately applied (filling general positions only)

Table 3 – *Continued from previous page*

参数名称	输入类型	默认值	功能说明
#SYMM=spg #SYMM=~spg	string	off	生成的结构必须在 spg 空间群中, spg 是空间群的名称. 若输入中含有波浪号 (~) 则表明, 结构搜索时将只在 general positions 上放置原子, 对于对称性更高但数量更少的 special position 不予考虑, 在此模式下不推荐使用MULT 等参数限制原子对称复制的个数.
#SYMMNO=n #SYMMNO=~n	int	off	生成的结构必须在第 n 号空间群中, n 是空间群的序号. 若输入中含有波浪号 (~) 则表明, 结构搜索时将只在 general positions 上放置原子, 对于对称性更高但数量更少的 special position 不予考虑, 在此模式下不推荐使用MULT 等参数限制原子对称复制的个数.
#SYMMORPHIC	void	off	只检查体系是否存在点式对称操作.
#SGRANK=n	int	230	设置空间群寻找/锁定的序号上限 n. 此值设为 230 时, 接受任何空间群的对称性锁定.
#ADJGEN=n #ADJGEN=$n_{min} - n_{max}$	int	0	调整晶胞中使用 general positions(GP) 的个数. 该值为 0 时, 会最大程度地使用 GP 点位. 增大该值则将逐渐更多地使用 Special positions(SP) 点位. 如果尝试后发现难以生成符合条件结构, 则程序将动态增加该值. 关于 SP/GP 与晶体空间群的关系请参考: Wiki: Wyckoff Position.
#BREAKAMP=d	float	0.0	在晶格矢量 a 方向随机移动原子破坏原有对称性, 原子分数坐标移动距离: $d_a^{(frac)} = (random(0,1) \times d)^{1/3}$
#NOPUSH	void	off	对于距离过近的原子不引入 PUSH 步, 直接拒绝该构型. 该关键字不会关闭 #SPHERE 等关键字引入的晶体中心势场的 PUSH 步.
#PUSHSTEP=p	float	0.25	每一步 PUSH 移动距离大小 (step-size) 的比例参数.

Continued on next page...

Table 3 – *Continued from previous page*

参数名称	输入类型	默认值	功能说明
#PUSHMAX=n	int	100	设置最大 PUSH 步数 (在 buildcell 的输出中, 两个 X 之间所夹的 * | : - 符号的个数).
#OVERLAP=cov	float	off	结束 PUSH 之后, 再附加采用 TPSD[10]对势对晶体结构进行简单弛豫. cov 为收敛判据, 其值越小对晶格收敛限制越高, 推荐值为 0.1–0.2.
#RASH	float	off	在使用 TPSD 对势弛豫结束后再引入原子集之间的随机位移和旋转 (SHAKE step). 设置过 #OVERLAP 之后此参数才有效.
#RASH_POSAMP=d	float	1.0	设置由 RASH 引入的 SHAKE 步移动原子集的最大距离, 与 POSAMP 类似. 该项必须为一个小的正值, **不可设为负数**.
#RASH_ANGAMP=θ	float $(0, 360]$	30.0	设置由 RASH 引入的 SHAKE 步原子集绕自身中点转动的最大角度, 与 ANGAMP 类似. 该项必须为一个小的正值, **不可设为负数**.
#CELLADAPT	void	off	在设置#OVERLAP 的情况下附加设置此项, 可强制要求 TPSD 简单结构弛豫时同时尝试在保持体积不变的情况下改变单胞的形状. 体系默认不会在 TPSD 结构优化步改变单胞形状. 若存在晶格标记#FIX 或者设置了 #CELLAMP=0 则此参数失效.
#THREE=p	float	TODO	使用三体势能代替 TPSD 弛豫结构, **该参数对应的功能尚未在 airss-0.9.1 中实现**.
#COMPACT	void	on	对最终生成的单胞进行 niggli reduce 操作. 在晶格形状没有被锁定时 (未引入晶格标签#FIX, #CFIX, #ABFIX), 该项默认打开. R. W. Grosse-Kunstleve, N. K. Sauter and P. D. Adams, Acta Cryst., A60, 1-6 (2004)

Continued on next page...

[10]Two-point Step Size Gradient Methods - Barzilai and Borwein, IMA Journal of Numerical Analysis (1988) 8, 141-148

Table 3 – *Continued from previous page*

参数名称	输入类型	默认值	功能说明
#NOCOMPACT	void	off	强制关闭 COMPACT 操作.
#PERMUTE	void	off	在完成原子位置选定后, 重排 (按一定概率互相交换) 指定原子位置间的元素种类. 可以联合原子标签 PERM 使用.
#PERMFRAC=p	float $[0, 1]$	1.0	设置重排发生的概率.
#HOLE=d	float	-1.0	设置在晶格上切割球洞的半径. 设为负数时不对成型的结构做任何处理.
#HOLEPOS=f_a f_b f_c	float	random	设置在晶格上切割球洞的位置 (分数坐标). 默认为随机位置. 可与原子标签中的ATHOLE 联用.
#VACANCIES=n@elm	float, string	off	等待结构生成完毕后, 选取 n 个元素种类为 elm 的原子替换为空位. [11]
#MAXTIME=t	float	1.0	设置对一个猜测结构 PUSH 步使用时间的上限, 超过该时间程序将停止 PUSH 并重新猜测新结构. 默认 t = 1s.
#NFAILS=n	int	0	每个结构允许的失败次数 (在 buildcell 命令输出中出现 X 的次数). 若其值为 0, 则无限制.
#SPHERE=r	float	off	在单胞中心处引入一球状势能. 设置球势能的吸引半径为 r. 当原子与晶格中心距离大于 r 时, 会受到指向晶格中心的 PUSH.
#ELLIPSOID=r ε	float	off	在单胞中心处引入一椭球状吸引势. r 为椭球势能半长轴长度, ε 为形变程度. 当原子与晶格中心距离大于 r 时, 会受到指向晶格中心的 PUSH. $\varepsilon = 0$ 为球形, ε 越大畸变越严重.

Continued on next page...

[11] 该参数本来还设计了一种不加 @ 符号的输入, 但目前在 airss-0.9.1 中, 这种输入在后续处理中存在一些 BUG, 因此在此未予说明.

Table 3 – *Continued from previous page*

参数名称	输入类型	默认值	功能说明
#PANCAKE=r ε	float	off	在单胞中心处引入一圆饼状吸引势. r 为圆饼半径, ε 为形变程度. 当原子与晶格中心距离大于 r 时, 会受到指向晶格中心的 PUSH. $\varepsilon = 0$ 为偏平圆饼, $\varepsilon = 1$ 为接近球形的势能.
#CIGAR=r ε	float	off	在单胞中心处引入一雪茄状吸引势. r 为雪茄长度, ε 为形变程度. 当原子与晶格中心距离大于 r 时, 会受到指向晶格中心的 PUSH. $\varepsilon = 0$ 为针尖状势能, $\varepsilon = 1$ 为接近球形的势能.
#CYLINDER=r	float	off	在单胞中心处引入一圆筒状势能 (原子在 Z 方向不受力). r 为圆筒势能吸引半径. 当原子与晶格中心距离大于 r 时, 会受到指向晶格中心的 XY 方向的 PUSH.
#CORE=r	float	off	对定义的球状 (椭球状, 圆饼状, 雪茄状, 圆筒状) 吸引势附加排斥核心. 设置排斥核心半径 (长轴长度, 半径, 雪茄长度, 圆筒半径) 为 r. 当原子与晶格中心距离小于 r 时, 会受到远离晶格中心方向的 PUSH.
#WIDTH=l	float	off	使用平面状势能 (原子在 X 和 Y 方向均不受力), 附加计算 PUSH 步的距离. l 为平面状势能吸引长度. 当原子与原点 (origin) 距离大于 l 时, 会受到指向原点的 Z 方向的 PUSH.
#SHIFT=h	float	0.0	将平面势移动至 Z=h 的位置 (origin 的位置), 默认 $h = 0$.

现在您应该可以轻松读懂下述内容了:

```
%BLOCK LATTICE_CART
20 0 0
0 20 0
0 0 20
#FIX
%ENDBLOCK LATTICE_CART

%BLOCK POSITIONS_FRAC
Al 0.0 0.0 0.0 # Al1 % NUM=7-13 COORD=4
%ENDBLOCK POSITIONS_FRAC

#MINSEP=1.5
#CLUSTER
#OVERLAP=0.2
#RASH
#POSAMP=3.0
#MINBANGLE=80
#MAXBANGLE=120
```

另外, *.cell 文件中还可以完全不出现结构数据对应的数据区块. 比如您只对晶体大概的结构有一些模糊的认识 (比如, 只知道其晶体原子构成, 晶格大概的体积大小等), 仍然可以使用 AIRSS 进行结构搜索. 下面就是这样两则符合规范且十分简洁的 AIRSS 结构种子文件.

例 *4.1*

```
#VARVOL=15
#SPECIES=A%NUM=4,B%NUM=1
#NFORM=2
#MINSEP=1.5
```

例 *4.2*

```
#VARVOL=15
#SPECIES=A,B,C
#NATOM=2-8
#MINSEP=1.5
```

这里用 #VARVOL 代替了晶格参数数据区块 BLOCK LATTICE_CART, 用 #SPECIES 代替了原子位置数据区块 BLOCK POSITIONS_FRAC.

4 结构弛豫与能量计算

4.1 联合 airss-pp3 模块弛豫

使用 AIRSS 联合 `airss-pp3` 计算模块预测结构时, 除了要准备一个 *.cell 的文件外, 还需要准备一个名为 *.pp 文件.

`airss-pp3` 是 AIRSS 自带的 pp3 对势 (pair potential) 计算模块. 其功能是使用化学上经典的分子势场 (如 6-12 势) 弛豫原子结构并输出体系能量. 由于这一模块使用的是维象的晶体能量模型, 不涉及任何第一性原理的复杂运算, 因此其实现简单、效果稳定、计算速度极快, 适用于简单组分的小体系. 更具体的, `airss-pp3` 采用了如下势能计算原子的受力和体系总能:

$$E_{ij} = 4\varepsilon_{ij}\left[\left(\frac{\sigma_{ij}}{r_{ij}}\right)^{m} - \beta_{ij}\left(\frac{\sigma_{ij}}{r_{ij}}\right)^{n}\right] \tag{1a}$$

$$E = \sum_{i<j}^{\text{all ions}} E_{ij} \tag{1b}$$

其中 i, j 标定了不同原子位置, 当元素种类不同时, $\varepsilon_{ij}, \sigma_{ij}$ 等变量对应不同的值.

使用该模块时, 需要首先配置名为 *.pp 的参数文件. 这个文件中存储了对势的相关参数, 其内部书写形式如下:

```
n_spec m n range
specs
# Epsilon
eps_11 eps_12 ...
eps_22 ...
...
# Sigma
sgm_11 sgm_11 ...
sgm_11 ...
...
# Beta
beta_11 beta_12 ...
beta_22 ...
...
```

上述参数中,

1. 第 1 行各项分别是: 元素个数; 对势中指数 m, n 的数值; 对势中能量极小值 (力为 0 处) 处距离原子中心的距离 d_{min} 与 σ 的比值, 也即 $d_{min}^{(ij)} = \sigma_{ij} * d_{range}$.
2. 第 2 行指明了体系中的元素种类, 不同元素间用空格隔开.
3. 第 3 行是注释行, 在程序中无意义, 但必须存在.
4. 第 4 至 $(n_{spec} + 3)$ 行声明了元素之间的 varepsilon 值对应的矩阵.
5. 第 $(n_{spec} + 4)$ 行是注释行, 在程序中无意义, 但必须存在.
6. 第 $(n_{spec} + 5)$ 至 $(2n_{spec} + 4)$ 行声明了元素之间的 sigma 值对应的矩阵.
7. 第 $(2n_{spec} + 5)$ 行必须为书写含有 “Beta" 字符的注释. 若此字符未出现, 则系统将强制把全部 beta 值设为 1.
8. 第 $(2n_{spec} + 6)$ 至 $(3n_{spec} + 5)$ 行声明了元素之间的 beta 值对应的矩阵.

例如,

```
2 12 6 5
A B
# Epsilon
1.00 1.50
0.50
# Sigma
2.00 1.60
1.76
```

下面通过一个例子来演示 AIRSS 联合 pp3 的计算过程.

```
user@machine_name$ ls
Al.cell Al.pp
user@machine_name$ cat Al.cell
%BLOCK LATTICE_CART
2 0 0
0 2 0
0 0 2
%ENDBLOCK LATTICE_CART

%BLOCK POSITIONS_FRAC
Al 0.0 0.0 0.0 # Al1 % NUM=8
%ENDBLOCK POSITIONS_FRAC

#MINSEP=1.5
user@machine_name$ cat Al.pp
1 12 6 2.5
Al
# Epsilon
1
# Sigma
2
# Beta
1
user@machine_name$ airss.pl -pp3 -max 3 -seed Al
user@machine_name$ ls -1
Al-43867-3302-1.res
Al-43867-3302-2.res
Al-43867-3302-3.res
Al-43867-3302.cell
Al.cell
Al.pp
user@machine_name$
```

4.2 联合 CASTEP 弛豫

官方推荐 AIRSS 结合 CASTEP 使用, 且构建了 AIRSS 和 CASTEP 间十分完善的接口.

要使用 CASTEP 联合 AIRSS 计算, 首先需要将成功安装的 CASTEP 可执行文件 `castep.serial` 或 `castep.mpi` 复制到 AIRSS 的 `bin` 目录中, 并重命名为 `castep`.

使用 CASTEP 联合 AIRSS 计算时, 除了 *.cell 外, 还需要准备 *.param 文件. *.param 是 CASTEP 的配置文件, 您可以在其中定义 CASTEP 计算过程中的必要配置参数, 包括, 计算的类型 (结构优化, 自洽, 光学性质计算, 能带计算等), 电

荷, 自旋取向, 截断能, 收敛标准等. 该文件通常由若干行组成, 每一行包含一个 keyword 及其相应的赋值.

***.param 文件主要有以下特点:**

1. 任何两个 keywords 之间没有书写顺序上的限制.
2. 您可以使用 # 或; 或! 又或者单词 COMMENT 来添加注释.
3. *.param 中设定的所有的 keywords 和数据均不区分大小写, 同时, 任何标点符号 (除了标明注释内容的符号), 多余的空格和任何空行都将被自动忽略.
4. 文件的任何一行中最多只能出现一个 keywords 及其对应参数.

*.param 文件每一行的基本格式均为:

```
[keywords] : [value]
```

其中的 ':' 是为了书写美观便于区分内容所加, 程序实际执行时会自动忽略, 您也可以完全不加入这一符号转而用空格代替.

CASTEP 中[keywords] 的定义和使用方法, 可参考: CASTEP cell keywords and data blocks.

AIRSS 默认联合 CASTEP 计算, 因此运行下述命令即可启动结构搜索.

```
1 user@machine_name$ airss.pl -max 3 -seed Al
```

4.3 联合 VASP 弛豫

AIRSS 自带了 airss.pl -vasp 选项用于联合 VASP 弛豫计算. 但官方自带的接口在并行计算方面还有待完善. 基于 AIRSS, 笔者用一套 bash 命令集重新编写了 AIRSS 至 VASP 的接口, 将其命名为airss4vasp(a4v). 后虽考虑过将此命令集使用 python 重新编写, 但 AIRSS 本身无法在 Windows 上运行, 而复写工程又过于庞大且收效甚微, 因此 a4v 目前仍然主要基于 bash 实现.

a4v 可以看做是 AIRSS 的改版, 无需安装原生 AIRSS 便可独立运行. 其主要基于 AIRSS 原生的 buildcell 模块, 同时内嵌了 PBS, NSCC, Slurm 等作业提交系统指令, 真正做到了一键提交 AIRSS+VASP 任务的功能, 同时对计算的并行也有较好的支持. 其具体用法可参见项目内部的 README.md 文件. [12]

在 *.cell 文件设置方面: a4v 增加了对应原子位置弛豫固定的SD-XYZ, SD-XY, SD-X 等原子标签; 同时删减了原先 buildcell 中设置共线自旋数值的全局参数, #SPIN=.

[12] 事实上, 该说明手册也在此项目中.(笑

5 数据后处理

5.1 *.res 文件结构

AIRSS 的计算结果全部储存在了 *.res 文件中. 这种 *.res 结构文件最早在 SHELX 中使用. 由于一些历史原因被 CASTEP 和 AIRSS 复用. 由于 SHELX 本身是对 Windows 友好的程序, 因此其输入输出文件的书写格式也沿袭了部分 Windows 文档的特点, 如倾向使用大写字母, REM 代表注释行, 文件使用 END 结尾等.

```
user@machine_name$ cat Al-43867-3302-1.res
TITL Al-43867-3302-2 0.0000000004 60.4852769773 -53.2712053113 0 0
    8 (P63/mmc) n - 1
REM
REM in /Users/alex/Documents/ProgramCode/MaterialCalculateProgram/
    AIRSS/airss-0.9/examples/1.1
REM
CELL 1.54180 2.2 5.2 5.2  86.6 90.0 90.0
LATT -1
SFAC Al
Al      1  0.2544637028970  0.9316224149716  0.6657635302849 1.0
Al      1  0.7544640475988  0.0982890099295  0.3324301203388 1.0
Al      1  0.2544640470150  0.3482890078890  0.5824301202379 1.0
Al      1  0.2544640470479  0.8482890103459  0.0824301202324 1.0
Al      1  0.7544640476367  0.5982890097930  0.8324301190566 1.0
Al      1  0.7544637023180  0.1816224159838  0.9157635306900 1.0
Al      1  0.7544637024482  0.6816224143044  0.4157635299253 1.0
Al      1  0.2544637030384  0.4316224167828  0.1657635292340 1.0
END
user@machine_name$
```

AIRSS 输出的 *.res 文件各行的含义如下:

1. 第一行 TITL 中的第一项是软件分配给该结构的名称标签, 第二项是系统外加静水压 (GPa), 第三项是单胞体积, 第四项是每个单胞总的焓 (能量), 第五项是原子自旋值的平均值, 第六项是原子自旋绝对值的平均值, 第七项是体系的总原子数, 第八项是体系所在空间群名称, 最后一项是固定字符 n - 1.
2. 之后若干以 REM 开头的行是注释行, 记录了文件生成的基本信息, 删除后不会有任何影响.
3. 紧接着以 CELL 开头的行记录了基本的晶胞信息. 其中, 第一项是一个无意义的小数, 这个小数在原来的 SHELX 程序中是用来记录得到相关结构所用衍射谱的波长, 在 AIRSS 中锁定为了一个无意义的小数. 第二至四项是晶格常数 a b c, 接下来五至七项是晶角 α β γ.
4. 下面一行是 LATT -1. 这一行在 SHELX 中用于标定晶格的对称性, 在 AIRSS 中锁定为固定值-1.
5. 接下来以 SFAC 开头的一行记录了构成体系的全部元素名称, 不同的元素用一个空格隔开.
6. 最后若干行标定了单胞中原子的位置. 这些行中的第一列是元素符号, 第二列指明了该元素在 SFAC 行中出现的次序, 第三到五列是该行储存了原子的分数坐标, 最后一列是原子的占据数, 一般设为 1.
7. 文件最终以 END 行结尾.

5.2 数据批量化处理

有了计算数据 *.res 文件后, 就可以使用ca 指令进行数据处理了. ca 是对 AIRSS 中基本分析套件 cryan 的封装.

```
user@machine_name$ ca
ca [-R(recurcsive)] [command line arguments for cryan]
user@machine_name$
```

cryan 的使用方法如下:[13]

```
user@machine_name$ cryan

Usage: cryan [OPTIONS]
The str. are read from STDIN, for example:

     cat *.res | cryan -s
     gunzip -c lots.res.gz | cryan -f H2O
     find . -name "*.res" | xargs cat | cryan -m

cryan options (note - the order matters):

 -r,  --rank                          Rank all str.
 -s,  --summary                       Summary
 -e,  --enthalpy <length_scale>       Plot enthalpy vs. pressure
 -f,  --formula <formula>             Select str. of a given com.
 -fc,  --formula_convert <formula>    Attempt to convert.
 -t,  --top [num]                     Output top few results
 -u,  --unite <thresh>                Unite similar str.
 -dr, --distance <rmax>               Distance threshold
 -de, --delta_e <energy>              Ignore str. above energy
 -sd, --struc_dos <smear>             Plot a structural DOS
 -p,  --pressure <pressure>           Additional pressure
 -m,  --maxwell                       Extract the stable com.
 -ph, --pressure_hull                 Ext. the stable str. with P
 -<n>                                 Component <n>
 -xg, --xmgrace                       Plot output with xmgrace
 -c,  --compare <thresh> <structure>  Compare structures
      --delete                        Delete unwanted str.
 -g,  --geometry [thresh]             Calculate the atomic geometry
 -n,  --num_units                     Report n separate str.
 -d,  --dimensionality                Report dD str.
 -cl, --cluster                       No periodic boundary
 -bl, --bondlength                    Maximum bond length
 -bs, --bondscale                     Bond length scaling
 -dm, --deltamodularity               Modularity bias parameter
 -wt, --weight                        Weight the adjacancy matrix
 -ns, --notsymm                       Clusters  point group off
 -sc, --struct_comm <thresh>          Determine the community str.
 -cm, --community                     Output the community str.
 -am, --adjacancymatrix               Output the adjacancy matrix
 -x,  --xyz                           Output clusters in XYZ format
 -o,  --off                           Output polyhedra in OFF
 -al, --alpha                         Construct alpha shapes
 -l,  --long                          Long names for str.
 -h,  --help, -?                      Print usage information
user@machine_name$
```

[13]为了行文简洁, 对参数的说明做了少许简化, 请自行运行上述指令查看更详细的信息.

使用ca 就可以对之前的计算结果进行分析.

```
user@machine_name$ ca -r > analysis.data
user@machine_name$ cat analysis.data
Al-43867-3302-2    0.00   7.561   -6.659    8 Al   P63/mmc   1
Al-43867-3302-1   -0.00   7.561    0.000    8 Al   P63/mmc   1
Al-43867-3302-3    0.00   7.564    0.005    8 Al   Fm-3m     1
user@machine_name$
```

上述输出结果中,

1. 第一列是 AIRSS 软件分配给该结构的名称标签
2. 第二列压力值 (GPa)
3. 第三列是每个化学式结构单元 (fu) 的体积
4. 第四列第一行是一个化学式结构单元 (fu) 的焓值, 之后的几行是不同结构下相对于第一行的焓值
5. 第五列是单胞中化学式结构单元 (fu) 的总个数 (单胞中 fug 的个数乘以一个 fug 中 fu 的个数.)
6. 第六列是化学式结构单元 (fu) 的化学式
7. 第七列是空间群名称
8. 第八列是所有搜索结果中出现该结构的次数

如果您认为所列结果过多, 可以使用-u 选项, 但是要注意, -u 一定要排在-r 之前使用.[14] 如果您仔细看过 cryan 的 help 信息就不难发现这样一个提醒: “note - the order matters”.

```
user@machine_name$ ca -u 0.01 -r > analysis.data
user@machine_name$ cat analysis.data
Al-43867-3302-2    0.00   7.561   -6.659    8 Al   P63/mmc   2
Al-43867-3302-3    0.00   7.564    0.005    8 Al   Fm-3m     1
user@machine_name$
```

指令 ca -u 后所跟的数字是一个无量纲的比例. 可以将这一参数简单的做如下理解: 他标定了晶格相似度阈值. 该数值越大, 容忍度越高, 最终展示出来的不同结构越少. 更详细的, 这一参数标定的距离, 是原子结构内部最接近的两个原子之间距离的倍数. 例如, 现有两个结构十分相似, 晶格中最短键长为 1.5 Å, 则 ca -u 0.1 就意味着, 依次比较两结构对应原子的两两距离, 如果未能发现这些距离差存在大于 0.15 Å 的情况, 则标定这两个结构一致. [15] 该值可根据需求调整, 建议在 0.1–0.01 之间选择.

[14] 事实上, 所有排在 rank (-r) 任务之后的参数都会被自动忽略.

[15] 这一段描述只是一个粗浅直观的解释, 实际使用的算法要更加复杂且稳定. 详细见源码 “airss-0.9.1/src/cryan/src/cryan.f90”2133 行.

A AIRSS 安装日志

下面将以 airss-0.9.0 版本为例, 简要记录 AIRSS 的安装.

AIRSS 只支持在命令行 (Command Line) 使用, 且仅能安装在 *nix 系统中. 安装此软件前, 您最好已经了解 GNU make 的使用方法. 当然, 如果您实在对此不感兴趣, 这不是必须的. 前提是您能完全按照以下步骤操作.

A.1 软件主体安装

具体的安装分为以下几步, 非必须步骤已使用* 标出:

***(I) 建立安装包文件管理系统** 在开始一切安装之前, 建议作为非root 用户但是有sudo 权限的您: 在自己能进行任意操作的家目录~/中建立一个安装包管理文件夹, 如~/install_package; 同时在系统目录/usr/local 中建立一个存放 airss 和其他程序二进制可执行文件的目录, 如/usr/local/airss-0.9/bin.

之所以这样建议, 是为了减少您安装过程中在系统目录下需要进行的操作, 降低由此可能引发的事故的概率, 同时让安装过程更简洁 (避免每个命令都要使用前缀sudo ..., sudo sh -c "...>...").

当然, 您也可以完全不将软件安装在系统目录, 一切都凭您的个人喜好.

```
user@machine_name$ cd /usr/local/
user@machine_name$ sudo mkdir -p airss-0.9/bin
Password:
user@machine_name$ cd airss-0.9
user@machine_name$ ls -F
bin/
user@machine_name$ cd
user@machine_name$ mkdir -p  install_package/AIRSS
user@machine_name$ cd install_package
user@machine_name$ ls -F
AIRSS/
```

(II) AIRSS 安装包下载 您可以访问前文所述官方网站下载 airss-0.9.0.

您可以选择在浏览器上下载, 也可以使用wget 指令.

```
user@machine_name$ wget -P ~/Downloads https://www.mtg.msm.
cam.ac.uk/files/airss-0.9.tgz
```

(III) 拷贝并解压安装包 将您下载的airss-0.9-2.tag 拷贝到安装包管理文件夹中, 并使用tar 解压.

```
user@machine_name$ cd AIRSS
user@machine_name$ cp ~/Downloads/airss-0.9-2.tgz .
user@machine_name$ tar -zxvf airss-0.9-2.tgz
x airss-0.9/.hg_archival.txt
x airss-0.9/.hgignore
x airss-0.9/LICENCE
x airss-0.9/README
x airss-0.9/VERSION
...
...
```

(IV) 使用 GNU make 指令安装 AIRSS 使用make 等指令安装编译安装 AIRSS.

```
user@machine_name$ cd airss-0.9
user@machine_name$ make
(cd src/pp3/src; make)
gfortran -O3 -c ../../common/constants.f90
gfortran -O3 -c cell.f90
gfortran -O3 -c pp.f90
gfortran -O3 -c opt.f90
gfortran -O3 -c pp3.f90
...
...
user@machine_name$ make install > make_install.log 2>&1
user@machine_name$
user@machine_name$ cat make_install.log
(cp src/pp3/src/pp3 bin/)
(cp src/cabal/src/cabal bin/)
(cp src/buildcell/src/buildcell bin/)
(cp src/cryan/src/cryan bin/)
user@machine_name$
```

十分鼓励您今后使用make isntall 指令时, 将其输出重定向到一个记录文件中, 这样会给您卸载软件时提供便利.

***(V) 安放可执行文件** ~/install_package/AIRSS/airss-0.9/bin 存放了安装完毕的可执行文件, 将其拷贝至系统目录下.

```
user@machine_name$ sudo cp -r bin/ /usr/local/airss-0.9/bin
Password:
user@machine_name$ ls /usr/local/airss-0.9/bin
airss.pl      cabal          cell2lammps   crud.pl
despawn       gulp_relax mc  pp3_relax     psi4_relax
spawn-slow    tidy.pl        buildcell     castep2res
check_airss   cryan          gap_relax     lammps2cell
niggli        press          run.pl        stopairss
ca            castep_relax   comp2minsep   csymm
gencell       lammps_relax   pp3           prim
spawn         symm
```

(VI) 设置系统环境变量 完成以上所有设置后, 您实际上就可以通过使用使用命令/usr/local/airss-0.9/bin/airss.pl -[option] [parameter] ... 来运行 AIRSS 了. 为了简便, 可以考虑在~/.bash_profile 文件中加入如下内容

```
## Setting PATH for AIRSS
export PATH="/usr/local/airss-0.9/bin:${PATH}"
```

修改储存并退出后, 请重新登入终端, 或运行source 指令完成环境变量的更新.

```
user@machine_name$ source ~/.bash_profile
```

这样您就可以在系统中的任何路径上执行airss.pl 等 AIRSS 的指令了.

(VII) 检查安装情况 设置好环境变量后, 您可以在~/install_package/AIRSS/airss-0.9/下输入make check 指令检查 AIRSS 安装情况.

```
user@machine_name$ make check
(sh bin/check_airss)
Essential:

airss.pl +
run.pl +
crud.pl +
castep2res +
buildcell +
cryan +
pp3 +
cabal +
cellsym - Install cellsym: http://www.tcm.phy.cam.ac.uk/sw
/check2xsf/cellsym.html
symmol - Patch and install symmol: http://www.
ccp14.ac.uk/ccp/web-mirrors/symmol/~pila/symmol.zip
bob - Get Bob!

Recommended:

castep - Install castep: http://www.castep.org/
optados - Install optados: http://www.tcm.phy.cam.ac.uk/
~ajm255/optados/index.html
qhull - Install qhull from package manager, or:
http://www.qhull.org/
qconvex - Install qhull from package manager, or:
http://www.qhull.org/
xmgrace - Install grace from package manager or:
http://plasma-gate.weizmann.ac.il/Grace/
Rscript - Install R/Rscript and ggtern from package manager
or: https://cran.r-project.org/

Optional:

gulp - Install gulp: http://projects.ivec.org/gulp/
cif2cell - Install cif2cell from: http://cif2cell.
sourceforge.net/

Very optional:

lammps - Install lammps: http://lammps.sandia.gov/
hull - Install hull: http://www.netlib.org/voronoi/
hull.html
off_util - Install antiprism: http://www.antiprism.com/
files/antiprism-0.24.1.tar.gz

Pseudopotentials:

pspot - set $PSPOT_DIR to location of the CASTEP pspot
directory

Spawn file:

.spawn -

--------------------
Tests run in .check:
--------------------

Running example 1.1 (Crystals):

Al-9002-4643-1    -0.00    7.561    -6.659    8 Al      n/a    1
```

```
Al-9002-4643-2     0.00    7.564      0.005    8 Al      n/a    1

Running example 1.2 (Clusters):

Al-9274-4255-2     0.00    615.385   -3.014   13 Al      n/a    1
Al-9274-4255-1     0.00    615.385    0.019   13 Al      n/a    1

Skipping example 3.1 (Gulp)
Skipping example 2.1a (Castep)
```

如果您仔细阅读了上述输出文件, 会发现必要的组件中还有cellsym 和symmol 没有安装. 这直接导致了晶体和团簇空间群符号输出为n/a.

A.2 辅助插件安装

AIRSS 支持的全部插件信息可查询~/install_package/AIRSS/airss-0.9/README 文件. 下面只演示最核心的cellsym 和symmol 插件的安装过程.

(I) 下载插件安装包 cellsym 的安装包官方网站是:

http://www.tcm.phy.cam.ac.uk/sw/check2xsf/cellsym.html

需要注意的是, cellsym 源码是使用 C 语言编写的, 安装此程序前, 需要下载并安装库文件spglib.h.

spglib.h 的下载地址是:

http://www.tcm.phy.cam.ac.uk/sw/check2xsf/spglib-1.9.4.tar.gz

cellsym 的下载地址是:

http://www.tcm.phy.cam.ac.uk/sw/check2xsf/cellsym.tgz

symmol 插件安装包的下载地址是:

http://www.ccp14.ac.uk/ccp/web-mirrors/symmol/~pila/symmol.zip

您可以通过浏览器下载上述文件, 也可以使用wget 指令下载.

```
user@machine_name$ wget -P ~/Downloads
www.tcm.phy.cam.ac.uk/sw/check2xsf/spglib-1.9.4.tar.gz
www.tcm.phy.cam.ac.uk/sw/check2xsf/cellsym.tgz
www.ccp14.ac.uk/ccp/web-mirrors/symmol/~pila/symmol.zip
```

(II) 拷贝并解压插件 将您下载的三个压缩包拷贝到安装包管理文件夹中, 并使用tar 和unzip 解压.

```
user@machine_name$ cd ~/Downloads
user@machine_name$ cp cellsym.tar spglib-1.9.4.tar
symmol.zip ~/install_package/AIRSS
user@machine_name$ cd ~/install_package/AIRSS
user@machine_name$ tar -xvf cellsym.tar
...
user@machine_name$ tar -xvf spglib-1.9.4.tar
...
user@machine_name$ unzip symmol.zip -d symmol
...
user@machine_name$ ls -F
airss-0.9/      airss-0.9-2.tgz    cellsym-0.16a/
cellsym.tar     spglib-1.9.4/      spglib-1.9.4.tar
symmol/         symmol.zip
```

(III) 编译插件 将解压好的插件按如下顺序操作.

首先安装库文件 spglib. 使用 GNU make 指令.

```
user@machine_name$ cd spglib1.9.4/
user@machine_name$ ./configure
...
user@machine_name$ make
...
user@machine_name$ sudo sh -c 'make install > make_install.log
     2>&1'
Password:
user@machine_name$ cat make_install.log
Making install in src
.././install-sh -c -d '/usr/local/lib'
/bin/sh ../libtool   --mode=install /usr/bin/install -c
    libsymspg.la '/usr/local/lib'
libtool: install: /usr/bin/install -c .libs/libsymspg.0.dylib
    /usr/local/lib/libsymspg.0.dylib
libtool: install: (cd /usr/local/lib && { ln -s -f libsymspg
    .0.dylib libsymspg.dylib || { rm -f libsymspg.dylib && ln
    -s libsymspg.0.dylib libsymspg.dylib; }; })
libtool: install: /usr/bin/install -c .libs/libsymspg.lai /usr
    /local/lib/libsymspg.la
libtool: install: /usr/bin/install -c .libs/libsymspg.a /usr/
    local/lib/libsymspg.a
libtool: install: chmod 644 /usr/local/lib/libsymspg.a
libtool: install: ranlib /usr/local/lib/libsymspg.a
/Applications/Xcode.app/Contents/Developer/Toolchains/
    XcodeDefault.xctoolchain/usr/bin/ranlib: file: /usr/local/
    lib/libsymspg.a(debug.o) has no symbols
.././install-sh -c -d '/usr/local/include/spglib'
/usr/bin/install -c -m 644 arithmetic.h cell.h debug.h
    delaunay.h hall_symbol.h kgrid.h kpoint.h mathfunc.h
    niggli.h pointgroup.h primitive.h refinement.h
    site_symmetry.h sitesym_database.h spacegroup.h
    spg_database.h spglib.h spin.h symmetry.h version.h '/usr/
    local/include/spglib'
make[2]: Nothing to be done for `install-exec-am'.
make[2]: Nothing to be done for `install-data-am'.
user@machine_name$
user@machine_name$
user@machine_name$ make install check
...
...
...
PASS: spglib_test
====================================
Testsuite summary for spglib 1.9.4
====================================
# TOTAL: 1
# PASS:  1
# SKIP:  0
# XFAIL: 0
# FAIL:  0
# XPASS: 0
# ERROR: 0
====================================
make[1]: Nothing to be done for `check-am'.
user@machine_name$
```

使用`make install check` 检查 PASS 后, 就可以开始编译`cellsym` 了.

```
user@machine_name$ cd ../cellsym-0.16a/
user@machine_name$ make
...
user@machine_name$ ls -all cellsym
-rwxr-xr-x  1 user  groups  53628 Jan 25 12:28 cellsym
user@machine_name$
```

顺利编译完成后, 会生成一个名为cellsym 的可执行文件. 注意, make 执行过程中可能会出现编译警告, 但这并不影响程序执行, 可忽略.

编译并确认生成了cellsym 文件后, 就可以开始编译另一个插件symmol 了. symmol 是使用 Fortran 写成的. 在网站上下载的是其源码, 需要编译使其变为可执行文件. 需要注意的是, 原版的symmol.f 并不兼容 AIRSS, 需要为其打上~/install_package/AIRSS/airss-0.9/misc 中提供的symmol.patch 补丁.

```
user@machine_name$ cd ../airss-0.9/misc/
user@machine_name$ cp ../../symmol/symmol.f .
user@machine_name$ ls
symmol.f      symmol.patch
user@machine_name$ patch -p0 symmol.f symmol.patch
patching file symmol.f
user@machine_name$ gfortran symmol.f -o symmol
user@machine_name$ ls
symmol        symmol.f      symmol.patch
user@machine_name$ echo '-o 后跟的文件名一定要是 symmol'
-o 后跟的文件名一定要是 symmol
user@machine_name$ ls -all symmol
-rwxr-xr-x  1 user  group  106800 Jan 25 12:41 symmol
user@machine_name$
```

至此, 我们完成了所有插件的编译. 生成了symmol 和cellsym 两个可执行文件.

(IV) 将插件导入 AIRSS 这一步的操作十分简单, 将编译好的两个插件复制到系统目录下的bin/文件夹即可. 为了以防万一, 可以在安装包管理文件夹保存一个bin/的备份

```
user@machine_name$ pwd
/home/user_name/install_package/AIRSS/airss-0.9/misc
user@machine_name$ cp symmol ../bin/
user@machine_name$ sudo cp symmol /usr/local/airss-0.9/bin
Password:
user@machine_name$ cd ../../cellsym-0.16a/
user@machine_name$ cp cellsym ../airss-0.9/bin/
user@machine_name$ sudo cp cellsym /usr/local/airss-0.9/bin
```

(V) 安装最终检查 回到airss-0.9 中执行make 的文件夹. 重新输入make check 检查安装情况.

```
user@machine_name$ cd ../airss-0.9
user@machine_name$ make check
(sh bin/check_airss)
Essential:

airss.pl +
run.pl +
```

```
crud.pl +
castep2res +
buildcell +
cryan +
pp3 +
cabal +
cellsym +
symmol +
bob - Get Bob!

Recommended:

castep - Install castep: http://www.castep.org/

...
...
...

--------------------
Tests run in .check:
--------------------

Running example 1.1 (Crystals):

Al-14776-403-2  -0.00   7.784  -6.398   8 Al    C2/m    1
Al-14776-403-1   0.00   7.820   0.066   8 Al    P21/m   1

Running example 1.2 (Clusters):

Al-15054-7410-1  0.00   615.385  -3.190  13 Al    Cs    1
Al-15054-7410-2  0.00   615.385   0.006  13 Al    Cs    1

Skipping example 3.1 (Gulp)
Skipping example 2.1a (Castep)
user@machine_name$
```

成功输出了晶体的空间群名称!

至此, 我们完成了 AIRSS 的基本安装, 您现在已经可以使用 AIRSS 的 pp3 模块 (默认是 CASTEP) 进行结构搜索了.

AIRSS 是受 GPL 许可证保护的开源软件. 对此程序您有以下三种权利:

* 以任何目的运行此程序
* 再复制
* 改进此程序, 并公开发布改进

A.3 卸载软件

AIRSS 卸载可分为三步:

(I) 卸载 spglib 进入安装包管理文件夹, 使用`make uninstall` 卸载 spglib.

```
user@machine_name$ cd ~/install_package/AIRSS/spglib-1.9.4
user@machine_name$ sudo make uninstall
...
user@machine_name$
```

(II) 删除相关文件夹 删除系统目录中的 bin 文件. 您可以选择保留安装文件. 保留安装文件可以在您试图恢复使用 AIRSS 时提供便利.[16]

```
user@machine_name$ cd /usr/local/
user@machine_name$ sudo rm -ri airss-0.9
Password:
user@machine_name$ cd ~/install_package/
user@machine_name$ rm -r AIRSS
```

(III) 恢复 PATH 变量 进入~/.bashrc 文件, 删除修改环境变量的语句即可.

```
###Setting PATH for AIRSS
export PATH="/usr/local/airss-0.9/bin:${PATH}"
```

[16]强烈建议您对文件进行删除时, 在离此文件较近的路径上操作, 并杜绝使用绝对路径, 以免打出文章开头提到的的毁灭性指令.