AIRSS 说明手册

Yang Li
lyang.1915@gmail.com

2020.10.24 — 2020.10.28

Contents

1 关于 AIRSS

AIRSS(**A**b **I**nitio **R**andom **S**tructure **S**earching) 是一款由英国剑桥大学 Chris Pickard 教授[1]等人开发的第一性原理结构搜索软件. 该软件是开源的, 受 **GPL2 许可证**保护. 访问其官方网站, `https://www.mtg.msm.cam.ac.uk/Codes/AIRSS`, 可获取安装源码.

所谓结构搜索是指: 对于一原子结构未知的体系, 在一定物理条件 (如原子间距, 分布密度, 元素构成, 元素配比等) 的限制下, 广泛地猜测其构型, 进行结构弛豫并计算其能量, 最终得到 (DFT 计算上的) 全局最稳定原子结构的过程. 显然人工手动猜测或是计算机盲目地遍历式搜寻是极为笨拙、耗时、甚至难以实现的. 因此, 就需要使用一套成熟的结构搜索软件, 系统且巧妙地捕捉体系的稳定构型.

AIRSS 正是这样一款软件. Chris 教授本人关于此软件的介绍可参考有关视频. 另外还有两个较为常用的结构搜索软件: USPEX 和 CALYPSO.

与 USPEX 或 CALYPSO 中使用的遗传算法不同, AIRSS 基于完全随机的结构搜索策略, 其产生的不同结构是完全随机且独立的, 这样的算法非常有利于搜索任务的并行实现.[2] 而这种随机方法的有效性在 Chris 教授关于 AIRSS 文章中也有所证明.[3] AIRSS 最强大之处在于其众多灵活的可调参数, 这可以说是此项目最出众的特点. AIRSS 的用户体验给人一种 “傻瓜相机 vs 专业单反相机” 的感觉. 前者简单实用, 可以一键提交各种不同类型的任务并获得不错的结果. 后者按键众多参数可调, 能自定义产生各种更加专业精细的结果. 前者是牺牲灵活性换取操作简易; 后者则相反. AIRSS 可以根据用户需求高度自定义研究对象的特征. 例如, 我们甚至可以使用此软件验证某个原子结构的稳定性, 或是进行晶格表面势能的计算.

令人遗憾的是, AIRSS 作为一个功能强大的结构搜索软件, 其官方手册却仍待开发. 再加上网络上相关说明文档或教程十分匮乏, 所以大部分人对 AIRSS 这样一个材料设计领域异常优秀的程序并不熟悉.

为了弥补这一缺憾, 笔者决定编写此文. 以下内容算不上指南或教程, 仅仅是学习 AIRSS 的一些记录. 本文绝大部分结论是自行分析源码、实际运行程序、尝试摸索所得.[4] **受各种因素限制, 理解和解释上的错误或不可避免. 如有问题, 还望您不吝指正.**

[1] Chris 教授同时还参与研发过 DFT 计算软件 CASTEP.

[2] 比如, 我们可以用 AIRSS 随机产生 1000 个结构, 而后独立弛豫

[3] [1] PRL 97, 045504 (2006); [2] JPCM 23, 053201 (2011)

[4] 官方虽然没有给出完整的用户手册, 但是提供了大量的使用范例, 位于程序的 example 目录中, 供使用者参考学习.

2 准备工作

2.1 Linux 系统

阅读本记录前, 您需要对 Linux 操作系统有一定的了解. 例如, 能理解以下指令的含义:

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

以及下述指令所能引起的灾难性事故:

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

另外需要特别提醒的是, 由于 PDF 文档编译时会将某些字符转化为命令行不能识别的符号 (如 ls −1 中的减号, 虽然他们看起来与命令号输入的减号并无不同), 所以**在直接复制粘贴本文档中的命令或字符使用时, 请加以小心**.

2.2 程序的安装与卸载

附录 A 中以 `airss-0.9.0` 版本为例, 简要记录了 AIRSS 的安装过程. 最新的 `airss-0.9.1` 在安装过程上做了大量简化, 您可以自行阅读其内包含的 `README` 文件, 安装此版本.

AIRSS 主运行脚本使用 perl 语言写成, 仅能安装在 `*nix` 系统中, 且只支持在命令行 (Command Line) 使用. 安装此软件前, 您最好已经了解 GNU make 的使用方法.

2.3 初次运行 AIRSS

初次接触 AIRSS, 您可以在终端输入 `airss.pl` 指令查看软件欢迎界面.

```
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` 是使用 AIRSS 执行结构搜索的主要指令. 欢迎界面中已经系统且简要地说明了该指令的用法. 用法解释中的表格有三列. 第一列是传入参数的名称; 第二列是传入参数的数据类型, `f` 代表浮点数, `n` 代表整数, `s` 代表字符串, “空” 代表逻辑字符串 `true` 或 `false`. 第三列是对相应参数的简单描述.

3 自定义随机结构

AIRSS 的核心组件名为 **buildcell**. 这个组件的作用就是依据用户给出的 *.cell 文件, 生成一系列结构随机但符合给定物理条件约束的初始原子构型. 另外, 我们也可以单独拿出此模块, 使其与其他程序 (如 VASP 等) 适配. 学会书写出高度自定义化的 *.cell 文件是学习 AIRSS 的根本.

为了方便后续的说明, 首先简要介绍 buildcell 组件运行的基本逻辑.

buildcell 区块的执行大致分为以下几步:

1. 通过 cell.f90 模块读取 *.cell 文件中的配置信息.
2. 通过 build.f90, opt.f90 等模块生成满足要求的结构.
   (a) 首先根据给定的初始原胞, 以 0.95-1.05 的比例调整原胞体积, 同时生成满足该体积的晶格参数. 若存在晶格标记 #FIX 则忽略此步骤.
   (b) 按要求在随机位置生成某个原子.
   (c) 根据约束选择: 接受这一位置, 拒绝这一位置, 或者 **PUSH** 当前原子. **PUSH** 是指: 将原子位置过近的两原子在其连线方向上, 反向移开相同距离. 当原子标签中有 FIX 或 NOMOVE 时, 忽略这一操作.
   (d) 遍历体系中的全部原子, 直至更新完整个结构.
   (e) 视用户设置, 用唯象的对势 (pp3) 对结构进行简单弛豫.
3. 通过 buildcell.f90 模块输出上述满足要求的随机结构.

3.1 *.cell 文件结构概括

*.cell 文件是结构搜索的 “种子文件”, 您可以在此文件中设置搜索约束条件. 文件名称前的星号 (*) 代表是 Linux 系统下的通配符, 也即它代表任意字符. 在 AIRSS 文件系统中, 我们把后缀名前的部分称为 “seed name”. Seed name 可用于区分不同计算的同类文件. 文中后续见到所有通配符, 如果不特殊指明, 均代指 seed name.

***.cell 文件有以下特点:**

1. 由于 AIRSS 是由部分参与 CASTEP 研发的人员编写, 因此该程序对 CASTEP 极其友好, 这里的 *.cell 文件与 CASTEP 中的 *.cell 文件完全兼容.
2. *.cell 文件设置参数时, 需要使用一定的 keywords 标明所设参数的含义.
3. AIRSS 完全复用了 CASTEP 中进行结构声明的 keywords, 如 LATTICE_CART 等.
4. AIRSS 内置的 keywords 一般以 # 开头, 如 #RASH. (以使得 AIRSS 的 *.cell 文件与 CASTEP 完全兼容)
5. *.cell 文件主要由两部分组成: “结构数据” 和 “全局参数”. 相应的, 文件中的 keywords 也可以分为上述两类.
6. 任何 keyword 之间没有书写顺序上的限制.
7. *.cell 中设定的所有的 keywords 需要采用**大写字母**书写, 并注意在 keywords、等号和参数数值之间**不能添加空格**. 同时, 任何空行都将被自动忽略.
8. 该文件中使用双井号 (##) 标明注释内容.
9. 虽然程序允许单行中出现多个 keywords, 但出于对所写文件的规范性和易读性考虑, 最好一行只写一个 keyword.

接下来我们会介绍 *.cell 文件的具体书写方式, 能否独立高效使用 AIRSS 就在于能否掌握下面几页的内容. 比起那些 “关于 AIRSS” 之类的 “没人细看” 的前言章节, 此处可以说是该文章核心位置, 因此笔者想借用这个 “黄金段位” 讨论一些 AIRSS 程序设计合理性的问题, 这对于更好的使用该软件是相当有益的.

结构搜索想解决的问题其实是相当好定义的. 我们设想一个 N 维相空间, 空间的不同维度代表结构的不同可变参数 (如原子配比, 晶格基矢, 原子位置等), 空间中的每个点代表不同的原子构型, 每个构型对应一个能量值. 于是我们的系统中就会存在一个 $N+1$ 维的能量曲面 (比如我们的相空间是二维的, 那么就会存在一个关于能量的三维曲面). 结构搜索要做的事情, 就是找到这个能量曲面的部分/全部的能量极小值点, 进而确定对应能量全局最小值的结构 (也就是我们所说的最稳定构型).

解决的这个问题的一种思路, 是用一些十分巧妙的算法 (比如遗传算法或者机器学习), 使得体系基于不同的亚稳态构型 “进化”, 找到全局最稳定构型. 这种算法的实现步骤如下: 初始产生一系列随机构型; 而后用第一性原理计算软件弛豫这些构型, 从弛豫的结果中继承/学习低能结构的特点; 接着利用从上一代学习的信息产生下一代结构; 如此迭代, 最终期望找到一个全局的最稳态. 这种算法的优点是, 可以在一个很大的相空间中迅速定位能量极小值点, 这在不借助任何先验信息, 只是纯理论的预测某个结构时是十分有效的. 但同时, 由于产生的上一代和下一代结构之间有因果关系, 因此不同代的结构只能序列 (串行) 地计算. 另外使用这一类算法还会面临一个很大的挑战, 也就是所谓的 “相空间坍缩”(也即由于初始选取随机构型的局限性, 后续产生的新结构全部局域在某几个亚稳态附近). 为了解决这个问题, 我们不得不引入一个可调参数: “变异概率”(也即认为结构在继承父代结构基础上, 还有一定概率在某些位置发生异变). 但在某些情况下 (比如设置变异概率过高或过低), 这个参数并不如我们所想的有效.

AIRSS 则是使用了另一种结构搜索的策略. 它的 *.cell 文件中提供了大量的可调参数, 每一个参数对应着若干约束条件, 可以实现 **对相空间的精准限制**. 系

统在满足这些约束条件的相空间中**随机**设置初始构型, 而后经过第一性原理计算软件的弛豫找到初始构型附近的某个 (亚) 稳态, 这就算是完成了一次 “结构搜索”. 通过大规模 (比如几千次或上万次)“搜索”, 结合**结构查重算法**, 最终就能在约束的相空间中找到一个全局最稳态. 这样的稳态大概有两个特点: **能量较 (最) 低**, **重复出现次数较 (最) 多**.

“随机选取” 初始结构这种策略乍一听起来似乎是毫无道理且效率极低的, 但如果经过 “精准约束” 后得到的相空间不是非常大, 那么即使不使用十分复杂的遗传算法, 单单依靠简单的随机撒点, 也可以获得不错的结果. 同时, 随机设置初始结构在保证了**并行计算效率**的同时, 还能避免 “相空间坍缩” 的发生.

也许看到这, 你会说, 如果我们一开始在遗传算法中把第一代设置得和 AIRSS 中一样多, 比如 10000 个, 那最终的结果不是比 AIRSS 这种算法要好得多, 毕竟 AIRSS 只相当于遗传算法的第一步? 但事实是, 在一个小的相空间中, 我们用每代 10000 个构型迭代 100 代所得到的结果, 和直接用 1000000 个随机结构弛豫计算, 其实差别不大, 有时甚至是后者更精确一些. 而且光是说出 “每代 10000 个构型迭代 100 代” 这样的话就会感觉荒唐, 因为这种计算需要的算力是海量的. 一般来说, 我们会取每代构型数量在 100 这个量级. 所以我更倾向于认为这两种算法是互补的关系, 一个适用于串行地寻找一个超大相空间中的部分低能极值点, 另一个适用于并行地寻找某个被牢牢限制的小相空间内的全部极值点. 而且根据本人的经验, AIRSS 在较大的相空间中的表现, 并不像我们想象的那么差. 因为对实际满足物理约束的相空间, 其内的极值点是少量的 (不然这个结构在实验上可能会存在大量的亚稳态), 完全可以依赖第一性原理计算的弛豫过程, 由一个看起来随机但其实就在 DFT 弛豫能力范围内的初始点, 找到这些最终的极值点.

使用 **精准限制相空间大小**加 **随机搜索**的策略搜索结构, 可以说是一种非常清奇且有效的思路. 因为大部分时候, 我们并非需要完全理论的预测某个结构, 而是和实验合作 (比如实验上已经知道了元素配比和晶胞形状, 有时甚至可以通过电镜知道其大概的原子位置), 共同研究某种材料的原子结构. 此时原子可变的相空间可能并不是很大, 因此 “如何更精确地定义所研究的相空间” 就变成了比 “如何优化算法更快从而更快得到极小值点” 更重要的问题. 当然相空间的大小也是相对的, 比如我们选取 10 个随机构型显然不能覆盖任何所谓的 “小的相空间”, 只有当这个数字增大到比如 10000 的时候, 对某些体系, 我们才能勉强这么说. 但这里就引出了随机生成结构的另一个好处, 那就是天生适合并行计算. 我们知道, 结构搜索中几乎全部的时间都花在了结构弛豫上. 如果我们手上有 8000 个核, 使用 8 个核优化一个构型, 那么在同一时间我们就可以同时搜寻 1000 个结构, 如果有 80000 个核, 那么同一时间就可以计算 10000 个结构. 这种简单粗暴的高效并行方式, 是其他算法所不能比拟. AIRSS 同时集合了**精准限制相空间大小**和 **随机产生结构**两大特点, 在很多时候 (尤其在与实验合作共同搜寻原子结构时), 是十分高效的.

目前 *.cell 文件中可用参数的功能可分为以下几类:

1. **声明初始晶格结构.** 比如声明初始的晶胞基矢, 初始的原子位置等.
2. **对晶胞参数的约束.** 比如固定初始晶格基矢, 约束晶胞体系, 约束晶胞变化程度, 约束晶胞所在晶系, 为晶胞添加真空层, 是否扩胞等.
3. **对体系化学式的要求.** 比如体系是几元的, 电子是否能配平 (是否有悬挂键) 等.
4. **对原子位置的约束.** 如两原子之间的最短距离, 某个原子偏离初始位置的最大/最小距离, 某原子的配位数等.
5. **对体系对称性的约束.** 如体系有几个对称操作, 在什么空间群, 对称性寻找的精细程度, 是否引入破坏对称性扰动等.
6. **对程序本身搜索算法的调整.** 如, 接受搜索失败次数上限, 是否引入 PSUH, 是否引入 TPSD, 是否引入 RASH 等.
7. (不太常用的) 在晶体中加入一个覆盖整个晶胞的力场. 如, 加入一个球形力场, 椭球力场, 长条状力场, 平面力场等.

以下是预测金属铝结构所用到的 `Al.cell` 文件:

```
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$
```

上述 *.cell 文件的结构可做如下概括:

1. 前 9 行是由%BLOCK [keywords] 格式定义的数据读取区块, 这种格式在 CASTEP 中是十分常见的. 这些区块定义了晶体基本的**结构数据**.
2. 1 至 5 行数据的 [keywords] 是 LATTICE_CART, 声明使用笛卡尔绝对坐标系定义的单胞基矢.
3. 7 至 9 行使用的 [keywords] 是 POSITIONS_FRAC. 这在 CASTEP 中的意义是 “以分数坐标定义的原子位置”. 而在 AIRSS 中, 该数据模块不仅可以指明原子初始位置, 还可以定义搜寻过程中对单种原子的约束条件. 若初始不知道原子的具体位置, 可将其设为任意值, 如 (0, 0, 0), 程序将自行找到符合要求的原子位置. 原子位置设置的更具体细节将在之后给出.
4. 第 11 行的 #MINSEP 是 AIRSS 中的**全局参数**. #MINSEP=1.5 表明了任意两原子的间距不得低于 1.5 Å.

可以看到, *.cell 文件的结构大致分为两大部分: **结构数据**和**全局参数**.

3.2 *.cell 文件参数细节

3.2.1 结构数据

该部分沿用了 CASTEP 的结构文件中定义数据的模式. 结构数据由两部分构成: 晶格参数数据区块, 原子位置数据区块.

数据区块的具体模式如下所示:

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

可以在 AIRSS 中使用的数据区块关键字 ([keywords]) 已在 **Table 1** 中列出.

Table 1: AIRSS Cheat Sheet – Data BOLCK

参数名称	功能说明
LATTICE_CART	使用笛卡尔坐标系, 以一个 3 × 3 的矩阵定义**单胞**基矢. 通过在该区块中附加以下标签: `#FIX`, `#CFIX`, `#ABFIX`, 可分别实现: 固定整个晶格形状, 固定晶格常数 c, 固定晶格常数 ab. 另外, 需要强调的是, 如果要确保晶格参数不变, 除了在晶格参数数据区块中加入 `#FIX`, 以保证在猜测晶格结构时保持其为常值外, 还需要在 DFT 弛豫软件中做相应的设置, 以保证其在结构弛豫时也不变. **需要特别注意的是**, 如果晶格形状可以随机变换 (也即没有晶格标签 `#FIX`), 则该区块中定义的晶胞基矢所对应的体积 V_{targ} **不是**原胞的初始总体积. 体系最终的总体积算法如下: $V_{cell} = V_{targ}N_{symm}N_{all}/N_{lines}$ 或者 $V_{cell} = V_{targ}N_{symm}N_{atom}$, 其中 N_{all} 是考虑 `NUM=n` 定义后的单胞总原子数, N_{lines} 是 `BLOCK POSITIONS_*` 区块中定义的原子的总行数, N_{symm} 是体系对称操作群元个数, N_{atom} 是使用 `#NATOM=` 定义的原子个数.[5]

Continued on next page...

[5]之所对体系单胞体积做这样的设计, 是为了给变组分或变原子数目搜索提供便利.

Table 1 – *Continued from previous page*

参数名称	功能说明
LATTICE_ABC	以一个 2 × 3 的矩阵定义单胞的晶格参数. 矩阵第一行是参数 abc, 第二行是三个夹角. 在同一个 .cell 中, 与 LATTICE_CART 二选一即可. 通过在该区块中附加以下标签: #FIX, #CFIX, #ABFIX, 可分别实现: 固定整个晶格形状, 固定晶格常数 c, 固定晶格常数 ab. **需要特别注意的是**, 如果晶格形状可以随机变换 (也即没有晶格标签 #FIX), 则该区块中定义晶胞常数所对应的体积 V_{targ} **不是**原胞的初始总体积. 体系最终的总体积算法如下: $V_{cell} = V_{targ}N_{symm}N_{all}/N_{lines}$ 或者 $V_{cell} = V_{targ}N_{symm}N_{atom}$, 其中 N_{all} 是考虑 NUM=n 定义后的单胞总原子数, N_{lines} 是 BLOCK POSITIONS_* 区块中定义的原子的总行数, N_{symm} 是体系对称操作群元个数, N_{atom} 是使用 #NATOM= 定义的原子个数.
POSITIONS_FRAC	以晶格基矢坐标系下的分数坐标定义原子位置.
POSITIONS_ABS	以笛卡尔坐标系下的绝对数值坐标定义原子位置. 与 POSITIONS_FRAC 二者选一即可.
SYMMETRY_OPS	定义生成的单胞所必须满足的对称操作, 四行一组, 前三行定义旋转操作, 第四行行定义平移操作. 具体用法可参考 CASTEP:SYMMETRY OPS.

下面简单介绍, **晶格参数**数据区块和**原子位置**数据区块的详细书写方法.

晶格参数 首先介绍晶格参数的设定, 以 LATTICE_CART 为例.
例 *1*.

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

上述字段构建了一个 20 × 20 × 20 Å^3 的正方体作为晶体的单胞. 这里的 #FIX 被称为**晶格标签** (Lattice Tags).

它声明了晶格常数在搜寻 (生成初始随机结构) 过程中是不能改变的. 如果您想保证体系晶格常数在进行 DFT 弛豫时也不变, 则需要在相应计算软件的输入文件中设置.

原子位置　再介绍原子位置的设置方法, 以 POSITIONS_FRAC 为例.
例 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
```

通过这个例子可以看出, 原子位置结构数据的基本格式是:

```
[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. **需要特别注意的是**, 如果晶格形状可以随机变换 (也即没有晶格标签 #FIX), 则使用 BLOCK LATTICE_* 晶格数据区块中定义晶胞常数 (基矢) 所对应的使用全局参数 #TARGVOL=, #VARVOL= 定义的体积 V_{targ} **不是**原胞的初始总体积. 体系最终的总体积算法如下: $V_{cell} = V_{targ}(N_{all}/N_{lines})N_{symm}$, 其中 N_{all} 是考虑 NUM=n 定义后的单胞总原子数, N_{lines} 是 BLOCK POSITIONS_* 区块中定义的原子的总行数, N_{symm} 是体系对称操作群元个数.
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.

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
ANGAMP=θ	float [0, 180]	-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. 其优先级低于直接设置全局参数 #MINSEP=.
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. 同时向生成的 cell 文件中写入相关指令, 使该原子位置在使用 CASTEP 弛豫时也保持固定. 这一指令仅在晶格被 #FIX 时有效. [6]
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

Continued on next page...

[6] 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.

Table 2 – *Continued from previous page*

参数名称	输入类型	默认值	功能说明
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 #NFROM= 中的说明.
MULT=m	float	symm num	设置该原子点位的 multiplicity.[7] 格点的 multiplicity 告诉我们, 如果我们在该位置上放置一个原子, 那么对称性将产生多少个原子. 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, 也即控制生成原子位置的对称性高低. 该参数默认大小为群操作个数, 也即默认只寻找 general position. 若该行原子有标签 FIX 或 NOMOVE, 则 MULT 失效, 同时将 OCC 强制设为-1. 除了 MULT 外, 还可使用全局参数 #NFORM= 控制体系由对称性生成的原子个数, 具体细节请参考 #NFROM= 的说明. 关于格点 multiplicities 与对称性的更多信息可参考 PDF: Space Group and Multiplicities.

Continued on next page...

[7] 这一数值为正时, 程序会强制将 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. 也即在 AIRSS 中, multiplicities 是通过设置分数 OCC 实现的, 因此不设置 MULT 而直接设置分数的 OCC 也可达到一样的效果.

Table 2 – *Continued from previous page*

参数名称	输入类型	默认值	功能说明
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.

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
`#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.[8] 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.

Continued on next page...

[8] 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
#NFORM=n #NFORM=$n_{min} - n_{max}$	int	-1	声明单胞中实际存在的原子个数（即使在考虑对称性后也必须）是 %BLOCK POSITIONS_* 或 #SPECIES 中定义的 n 倍. 若为负值则此参数被关闭. 与原子标签 MULT 类似, 使用该参数可约束单胞中定义的原子因对称性而复制的份数, 但在单胞中有重复元素的原子时, #NFORM= 的效果会与 MULT 不同. 例如, 在有若干个对称操作的单胞中定义了 2 个 Si 原子. 将他们的 MULT 分别设置为 2, 则系统将产生 4 个原子, 每 2 个原子单独满足对称性. 而若设值 #NFROM=2, 则体系同样会产生 4 个原子, 但 4 个原子组合起来才会满足对称性. 该参数的出现会覆盖原子标签 MULT 的设置. 同时使用该参数, 也会将体系总体积计算公式[9]中的 N_{symm} 替换为 N_{fu}. 也即, $V_{cell} = V_{targ}N_{fu}N_{all}/N_{lines}$ 或者 $V_{cell} = V_{targ}N_{fu}N_{atom}$, 其中 V_{targ} 是由 BLOCK LATTICE_* 或 TARGVOl, VARVOL 定义的初始体积, N_{all} 是考虑 NUM=n 定义后的单胞总原子数, N_{lines} 是 BLOCK POSITIONS_* 区块中定义的原子的总行数, N_{fu} 是 #NFORM= 后跟的整数, N_{atom} 是使用 #NATOM= 定义的原子个数.[10] 同时, 在无对称性时, #NFORM= 不能（也没有必要）与 #NATOM= 联用.

Continued on next page...

[9] 请参考 BLOCK LATTICE_* 的说明.
[10] 使用原子标签 MULT 或 OCC 限制对称性时没有此效果.

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.[11]
#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.
#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.
#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.

Continued on next page...

[11] 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
#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 $[0, 180]$	-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 $(0, 180]$	0.0	设置搜索结构中, 原子键角的最小值.
#MAXBANGLE=θ	float $(0, 180]$	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.
#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

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
#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 POSITIONS_* 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. [12] 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. **需要特别注意的是**, 如果晶格形状可以随机变换 (也即没有晶格标签 #FIX), 则使用 BLOCK LATTICE_* 晶格数据区块中定义晶胞常数 (基矢) 所对应的、以及使用全局参数 #TARGVOL=, #VARVOL= 定义的体积 V_{targ} **不是**原胞的初始总体积. 体系最终的总体积算法如下: $V_{cell} = V_{targ} N_{symm} N_{atom}$, 其中 N_{symm} 是体系对称操作群元个数, N_{atom} 是使用 #NATOM= 定义的原子个数.
#FOCUS=n	int	0	与 #NATOM= 联用, 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.

Continued on next page...

[12] 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
#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, 结构搜索中该体积不再随机变化. 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. **需要特别注意的是**, 该处定义的体积 V_{targ} **不是**原胞的初始总体积. 体系最终的总体积算法如下: $V_{cell} = V_{targ}N_{symm}N_{all}/N_{lines}$ 或者 $V_{cell} = V_{targ}N_{symm}N_{atom}$, 其中 N_{all} 是考虑 NUM=n 定义后的单胞总原子数, N_{lines} 是 BLOCK POSITIONS_* 区块中定义的原子的总行数, N_{symm} 是体系对称操作群元个数, N_{atom} 是使用 #NATOM= 定义的原子个数.
#VARVOL=V	float	init. cell vol.	给定初始晶格体积, 实际的晶格体积会在给定体积的 0.95-1.05 倍之间随机选取. **需要特别注意的是**, 则该处定义的体积 V_{targ} **不是**原胞的初始总体积. 体系最终的总体积算法如下: $V_{cell} = V_{targ}N_{symm}N_{all}/N_{lines}$ 或者 $V_{cell} = V_{targ}N_{symm}N_{atom}$, 其中 N_{all} 是考虑 NUM=n 定义后的单胞总原子数, N_{lines} 是 BLOCK POSITIONS_* 区块中定义的原子的总行数, N_{symm} 是体系对称操作群元个数, N_{atom} 是使用 #NATOM= 定义的原子个数.
#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.
#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.

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
#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 (键长) 更加紧密. 该参数仅在原子标签 RAD 生效时有作用.
#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[13]更少的 special position 不予考虑. 若使用 MULT 逐个原子指定 multiplicity, 则波浪号 (~) 被忽略, 但若使用 #NFORM=n(n 大于 0), 则会与波浪号 (~) 冲突. 因此在含有波浪号 (~) 的模式下不能使用 MULT 或 #NFORM= 限制原子对称复制的个数. **需要补充说明的是,** 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, #NFORM= 等的使用, 可实现在更高对称的点位安放更少原子的操作.
#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, 在此模式下不能使用 MULT 或 NFORM 限制原子对称复制的个数.

Continued on next page...

[13] 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.

Table 3 – *Continued from previous page*

参数名称	输入类型	默认值	功能说明
#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, 在此模式下不能使用 MULT 或 NFORM 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 (stepsize) 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.[14] 在 buildcell 的输出有六种: ~, X, -, :, |, *. 波浪号 (~) 代表结构产生成功; 叉 (X) 代表产生失败, 失败原因可能是结构生成失败, 或者 PUSH 步失败, 而 PUSH 失败的判断标准就是经过 PUSHMAX 步的 PUSH 后, 体系是否满足要求; 短线, 冒号, 竖线, 星号 (-, :, |, *) 四个字符反映了单次 PUSH 步中体系所有原子被 PUSH 的总次数, 从左到右依次代表被 PUSH 了 0-9, 10-99, 100-999, 1000+ 次, 当前 PUSH 步刷新了之前单步 PUSH 内 PUSH 次数的记录时, 相应的符号才会输出.
#OVERLAP=cov	float	off	结束 PUSH 之后, 再附加采用 TPSD[15] 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 $(0, 180]$	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**.

Continued on next page...

[14] 请注意区分 'PUSH 步数' 和 '单步 PUSH 次数' 这两个概念. 前者是体系 PUSH 这个动作 (PUSH 函数) 循环执行的总次数, 后者则代表在每一次 PUSH 动作中 (在 PUSH 函数内部) 体系中原子被 PUSH 的次数.

[15] 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
#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)
#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. [16]
#MAXTIME=t	float	1.0	设置对一个猜测结构 PUSH 步使用时间的上限，超过该时间程序将放弃当前结构，并重新猜测新结构. 默认 $t = 1s$.

Continued on next page...

[16] 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
#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.
#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.

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
`#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=14
#SPECIES=A%NUM=2,B%NUM=2
#MINSEP=1.5
```

例 *4.2*

```
#VARVOL=7
#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:

```
1  n_spec m n range
2  specs
3  # Epsilon
4  eps_11 eps_12 ...
5  eps_22 ...
6  ...
7  # Sigma
8  sgm_11 sgm_11 ...
9  sgm_11 ...
10 ...
11 # Beta
12 beta_11 beta_12 ...
13 beta_22 ...
14 ...
```

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 $d^{(ij)}_{min} = \sigma_{ij} * d_{range}$.
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 自带了联合 VASP 弛豫计算的选项.

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

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.[17]

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. [18]

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=. 此外由于 VASP 本身对团簇 (cluster) 计算不是十分友好, 因此 a4v 中并未集成有关 cluster 的计算组件 (如对称性判断组件等).

[17] 如果把 C++ 比作文言文, 则 python 相当于现代白话, bash 等 shell 语言则是口语. 因此一般大规模脚本推荐使用 python 书写.

[18] 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:[19]

```
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$
```

[19] 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. 第二列是体系静水压力值 (GPa).
3. 第三列是每个化学式结构单元 (f.u.) 的体积.
4. 第四列第一行是对应结构的总能 (焓, eV/f.u.), 之后的几行是不同结构下相对于第一行的能量差.
5. 第五列是单胞中化学式结构单元 (f.u.) 的总数.
6. 第六列是化学式结构单元 (f.u.) 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.[20] 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. [21] 该值可根据需求调整, 建议在 0.1–0.01 之间选择.

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

[21] 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.[22]

```
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}"
```

[22]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.