计算材料学常用资源下载

本页面提供本站（我爱搜集网 http://www.52souji.net）所有可下载内容。持续更新！

要了解详细的使用方法，可在本站右侧搜索框中搜索相关关键词。

内容目录 [隐藏]

一. 软件下载

1. 可视化软件

- atomeye：Linux 下大规模原子可视化软件
- VESTA：功能强大的傻瓜式可视化软件
- VMD: a molecular visualization program for displaying, animating, and analyzing large biomolecular systems using 3-D graphics and built-in scripting.
- XCrySDen: a crystalline and molecular structure visualisation program aiming at display of isosurfaces and contours, which can be superimposed on crystalline structures and interactively rotated and manipulated. It can run on most UNIX platforms, without any special hardware requirements.

2. 计算程序

MD & MC

- LAMMPS: A powerful, efficient, parallelized, well documented, easy extendable and open source MD code.
- Gromacs: Another nice free MD code.
- DL-POLY: a general purpose classical molecular dynamics (MD) simulation software. open source.
- NAMD: a parallel molecular dynamics code designed for high-performance simulation of large biomolecular systems.

Ab initio (DFT)

- VASP: 功能强大、使用简单、广泛使用的第一性原理计算程序，非免费
- Quantum-Espresso: A nice, free DFT code
- Ab init: Another popular open source DFT code
- SIESTA: a linear scaling DFT code

3. 科学分析软件

- 1stOpt：即学即会的非线性拟合软件

二. 代码下载

1. 格式转换

- xyz 转 lammps
- vasp 转 lammps

2. 计算方法

- 粒子群优化算法 PSO
- 计算多边形面积
- 计算多面体体积
- 判断点在多面体内
- 判断两条之前平行
- verlet 积分算法
- verlet 速度算法

三. 标准与规范下载

- 晶格常数表
- 元素结合能表
- 体积热容量表
- 热膨胀系数表
- 体积模量表

四. 势函数下载

- Potentials: A collection of EAM potentials for some metal and alloys.
- Potfit: A code to fit atomic potentials from ab initio data.
- LAMMPS 常用势函数下载

LAMMPS 相关文件功能简介

由 www.52souji.net 发表于 2012 年 5 月 15 日 || 3,050 浏览

这篇文章是为初学 LAMMPS 的同学准备的，让你了解与 LAMMPS 相关的文件及其作用，具体的介绍会在其他博文中进行。

整体上来说，与 LAMMPS 相关的文件可以分为用于**输入到 LAMMPS 的文件**和**从 LAMMPS 输出的文件**，下面分别介绍。

内容目录 [隐藏]

输入到 LAMMPS 的文件

（这里没有直接叫“输入文件”，是因为有一个专门的文件叫做输入文件。）

主要包括三类，分别是输入脚本（input script），数据文件（data file）和重启动（restart）文件。

1. 输入脚本

input script，文本文件，必须。

不少人习惯直接叫它“输入文件”，我为了区别“输入的文件”，采用直译叫法“输入脚本”。又因为习惯使用 in. 作为这个文件的前缀，所以也常常被称作“in 文件”。

既然被称为 script，自然是因为里面包含了很多 LAMMPS 的命令。LAMMPS 运行的过程就是一行一行执行这个文件中的命令的过程，命令执行完了，LAMMPS 的运行也就结束了。因此，我们在使用 LAMMPS 进行计算模拟的时候，一个很重要的前提和核心就是提供可执行的正确的输入文件。

具体如何写 LAMMPS 输入文件，是一个很大的问题，我会在以后将我自己的经验（虽然不是很多）慢慢介绍给大家。

2. 数据文件

data file，文本文件，非必须。

我们知道要进行一个模拟计算，必须有一个初始构型，那么这个数据文件（ data file ）就是用来存放要模拟体系的初始构型的。简单一点的，里面就包含原子的坐标信息；复杂一点，里面会包含键长、键角等信息。

之所以它是一个非必须文件，是因为一些简单的初始构型，如 FCC 等，可以直接使用 LAMMPS 提供的命令进行创建。一般，只有当模型比较复杂时，才会使用这个文件。

3. 重启动文件

restart，二进制文件，非必须。

顾名思义这个文件是用于重启动一个计算，这对于减少重复计算会有很大的帮助。

刚开始接触可能有点难理解，我举个例子说明下。

比如要运行一个由 A-B-C 三段组成的计算任务，B 是以 A 最终状态为初态开始，C 是以 B 的最终状态为初态开始，那么我就可以在 A 计算结束后写一个 restart 文件，在运行 B 的时候读入；在 B 结束的时候再写一个 restart 文件，在运行 C 的时候读入。这样做，可以保证你在要重新进行 B 的计算时，不需要重复计算 A；重新计算 C 的时候，可以不用重复计算 A-B。

从 LAMMPS 输出的文件

主要包括日志（log）文件，结构文件，重启动（restart）文件和任意文本文件。

默认情况下，日志文件是一定会输出的，其他的都是有输入脚本文件中的命令决定是否输出的。

1. 日志文件

log：LAMMPS 在运行过程中，默认会产生日志文件，用于记录命令执行的情况。默认文件名为 log.lammps。

2. 结构文件

LAMMPS 可以输出多种不同类型的结构文件，用于记录某一时刻体系的构型（在 LAMMPS 里叫做 snapshot，即“快照”），但都是使用同一个 dump 命令输出。

主要的结构文件类型包括 cfg 格式，xyz 格式，lammps 格式，图像格式等。

3. 重启动文件

restart：与前面提到的输入文件一样。实际上，只有先使用命令产生 restart 文件，才有可能有前面的输入，也就是所谓的“重启动”。

4. 任意文本文件

LAMMPS 中有几个命令可以实现将模拟中定义的某些变量以一定的格式输出到文件。因为这类文件格式相对自由，所以我称这类文件为任意文本文件。

LAMMPS 常用建模方法总结

由 www.52souji.net 发表于 2012 年 5 月 16 日 || 5,327 浏览

建模是进行材料模拟的第一步，这里对 LAMMPS 常用的建模方法进行总结（我平时用到的，难免不全面）。

概况来说，建模方法有两种：内部建模和外部建模。

内容目录 [隐藏]

内部建模

内部建模，即使用 LAMMPS 提供的命令建立模型。这种方法主要用于构建比较简单和标准的体系。

相关命令主要有 4 个：

- lattice：定义晶格类型；
- region：定义模拟盒子的大小；
- create_box：创建模拟盒子；
- create_atoms：在模拟盒子中创建原子。

当然，创建模型不仅限于这些命令，还有一些其他的命令，比如 delete_atoms 等。事实上，我看到有些大牛仅仅使用 LAMMPS 的内置命令，就建立了很复杂的模型。

下面简单举一例：构建 6x6x6 的 FCC-Cu 的晶胞。

输入脚本：in.fcc-Cu

```
# model of FCC-Cu, 6x6x6

boundary        p p p
units           metal
atom_style      atomic

########################################
lattice         fcc 3.61
region          box block 0 6 0 6 0 6
create_box      1 box
create_atoms    1 box
########################################

pair_style      eam
pair_coeff      * * /home/xbduan/lammps/potentials/Cu_u3.eam

dump            1 all cfg 1 a*.cfg id type xs ys zs
dump_modify     1 element Cu
run             0
```

第 8~11 行即为建模部分。这里使用的命令参数很简单，不做过多介绍。

稍微复杂的模型构建，我会专门开博文介绍。

外部建模

外部建模主要是通过数据文件 data file 实现的，即只要你将你需要的构型以 data file 所要求的格式保存成文件，就能够被 LAMMPS 读入，而建立模型。

假如已经有了能够反映模型的 data file 后，就可以直接使用 read_data 命令完成建模了。

data file 基本格式

如下图所示，这给出的是 data file 最基本的格式，只包含原子坐标，而不包含键能键角等参数，一般的合金体系是这种类型。

```
Start File for LAMMPS   1298.88171          # 注释行

2000 atoms                                  # 总原子数

2 atom types                                # 原子种类

-0.5  35.5400009 xlo xhi                    # 模拟盒子x尺寸
-0.5  35.5400009 ylo yhi                    # 模拟盒子y尺寸
-0.5  35.5400009 zlo zhi                    # 模拟盒子z尺寸

Masses                                      # 关键字：定义原子质量

1  22.9899998                               # 第一种原子的原子质量
2  35.4500008                               # 第二中原子的原子质量

Atoms                                       # 关键字：定义原子坐标

1 1  10.8120003  32.4360008  7.20800018     # 原子坐标：编号 种类 x坐标 y坐标 z坐标
2 2  30.6340008  23.4260006  5.40600014
3 2  27.0300007  23.4260006  16.2180004
4 2  30.6340008  30.6340008  9.01000023
```

上面的注释已经比较明确了，不多解释。更多可以参考 LAMMPS 文档。

data file 获取方式

实际上，只要你最后得到的 data file 的格式类似于上面（不包括注释），就可以被 LAMMPS 读入，而不管你是采用什么途径。

目前来说，比较遗憾的是还没有任何一种软件支持直接导出 data file 的文件格式，但是因为 data file 文件格式还比较简单，所以还是比较容易获得的。

1. 软件导出+手动编辑

很多软件，如 Material Studio，具有强大的建模功能，而且支持很多种文件格式的导出。遗憾的是，目前还不支持直接导出 data file 格式。

（1）使用建模软件完成建模后，导出 xyz 文件格式（因为 xyz 文件格式与 data file 格式比较相近，比较容易修改）；

（2）使用文本编辑器，参考上图所示的 data file 的文件格式编辑 xyz 文件，主要是增加模拟盒子的参数。

这种方法适用于构型相对简单、原子数不多的情况下，否则编辑的工作量会很大。

我写过一个 matlab 脚本，可以完成这种格式转换：【xyz2lmp: 将 xyz 格式转成 lammps 的 data file】

2. 程序产生

使用程序建模是最自由的方式，因为它几乎不受任何限制。不论你的模型多么复杂，总能够使用一定的算法完成建模。

但，这要求你对你要建立的模型由深入的理解和有一定的编程能力来实现你的算法。

对于一些复杂的模型，如位错、多晶等缺陷，几乎只能用这种方法实现（如果你能够在网上找到现成的代码，You are lucky!）。

xyz2lmp: 将 xyz 格式转成 lammps 的 data file

目前，大部分的建模软件都支持导出 xyz 文件格式，而并不支持直接导出 lammps 所需要的 data file 文件格式。因此将 xyz 文件格式转换成 lammps 的格式是很有必要的。

原子数比较少的，可以自己手动编辑完成，对于原子数比较多的，实在比较麻烦。所以就写了下面的脚本可以帮助完成这一过程。

目前，程序只适用于合金体系，即不考虑力场参数（我的体系都属于这方面）。

因为 xyz 文件格式不包括晶胞信息，而 lammps 的 data file 文件是需要晶胞信息的，所以这个转换过程实际上是缺少信息的，为了弥补，请务必将晶胞信息按着类似以下格式放在 xyz 文件的注释行，也就是第二行，才可以完成转换，不然程序会出错。

```
01      system_name xlo xhi ylo yhi zlo zhi
```

输入 xyz 文件

test.xyz (只是为了测试程序，结构没有实际意义)

```
01
        11
02
        Cu 0 100 0 100 0 100
03
        Cu  0.000000     0.000000    0.000000
04
        Cu  3.800000     0.000000    0.000000
05
        Fe  7.600000     0.000000    0.000000
06
        Cu  11.400000    0.000000    0.000000
07
        Fe  1.900000     1.900000    0.000000
08
        Cu  5.700000     1.900000    0.000000
09
        Cu  9.500000     1.900000    0.000000
10
        Au  0.000000     3.800000    0.000000
11
        Cu  3.800000     3.800000    0.000000
12
        Au  7.600000     3.800000    0.000000
13
        Cu  11.400000    3.800000    0.000000
```

转换命令

```
01      &gt;&gt; xyz2lmp('test.xyz')
```

转换后 lmp 文件

test.lmp

```
Converted from .xyz to .lmp @ 04-May-2012 10:42:30

11   atoms
3    atom types
0.000000  100.000000    xlo xhi
0.000000  100.000000    ylo yhi
0.000000  100.000000    zlo zhi

Atoms

1    1   0.000000    0.000000    0.000000
2    1   3.800000    0.000000    0.000000
3    2   7.600000    0.000000    0.000000
4    1   11.400000   0.000000    0.000000
5    2   1.900000    1.900000    0.000000
6    1   5.700000    1.900000    0.000000
7    1   9.500000    1.900000    0.000000
8    3   0.000000    3.800000    0.000000
9    1   3.800000    3.800000    0.000000
10   3   7.600000    3.800000    0.000000
```

```
20      11   1   11.400000   3.800000    0.000000
21
```

xyz2lmp 源代码

xyz2lmp.m

```
01      function xyz2lmp(f_xyz)
02      % This script converts .xyz file to lammps data file.
03      % Input:
04      %   f_xyz: name of the input .xyz file.
05      % Example:
06      %   xyz2lmp('PdAu.xyz')
07      % NOTE: The second line must be in specified format as:
08      %   PdAu xlo xhi ylo yhi zlo zhi
09      % Poweed by Xianbao Duan
10      % Email: xianbao.d@gmail.com
11      % Website: http://www.52souji.net/
12
13      % open the .xyz file
14      fidin = fopen(f_xyz,'r');
15      if fidin == -1
            error('Failed to open the file. Please check!');
```

```
16      end
17
18      % number of all the atoms
19      atom_num_a = fscanf(fidin,'%d');
20
21      % comment
22      comment = textscan(fidin,'%s %f %f %f %f %f %f',1);
23      xlo = comment{2};
24      xhi= comment{3};
25      ylo = comment{4};
26      yhi= comment{5};
27      zlo = comment{6};
28      zhi= comment{7};
29
30      % coordinates of the atoms
31      atoms = textscan(fidin,'%s %f %f %f',atom_num_a);
32      fclose(fidin);
33
34      % initialize
        type_name = atoms{1}(1);    % names of types
```

```
35      atom_num(1) = 1;            % atom number of each type
36
37      % sort the atoms according to their types
38      for i = 2: length(atoms{1})
39          flag = 0;
40          for j = 1:length(type_name)
41              if strcmp(atoms{1}(i),type_name(j)) == 1
42                  atom_num(j) = atom_num(j) + 1;
43                  flag = 1;
44                  break;
45              end
46          end
47          if flag == 0
48              type_name(end+1) = atoms{1}(i);
49              atom_num(end+1) = 1;
50          end
51      end
52      type_num = length(type_name);
53
54      % write the lammps file
```

```
54      outfilename = strrep(f_xyz,'.xyz','.lmp');
55      fidout = fopen(outfilename, 'w');
56      new_comment =  ['Converted from .xyz to .lmp @ ',datestr(now) ];
57      fprintf(fidout,'%s\n\n',new_comment);
58      fprintf(fidout,'%d \t %s\n',atom_num_a,'atoms');
59      fprintf(fidout,'%d \t %s\n',type_num,'atom types');
60      fprintf(fidout,'%f  %f \t%s\n',xlo,xhi,'xlo xhi');
61      fprintf(fidout,'%f  %f \t%s\n',ylo,yhi,'ylo yhi');
62      fprintf(fidout,'%f  %f \t%s\n\n',zlo,zhi,'zlo zhi');
63      fprintf(fidout,'%s\n\n','Atoms');
64      % the data
        for i = 1:length(atoms{1})
65          for j = 1:type_num
66              if strcmp(atoms{1}(i),type_name(j)) == 1
67                  fprintf(fidout,'%d \t %d \t %f \t %f \t %f \n',...
68                      i,j,atoms{2}(i),atoms{3}(i),atoms{4}(i));
69                  break;
70              end
71          end
72      end
        fclose(fidout);
```

```
73

74

75

76

77
```

lammps 中构建合金模型

合金体系的创建，主要有两种方法，如下。

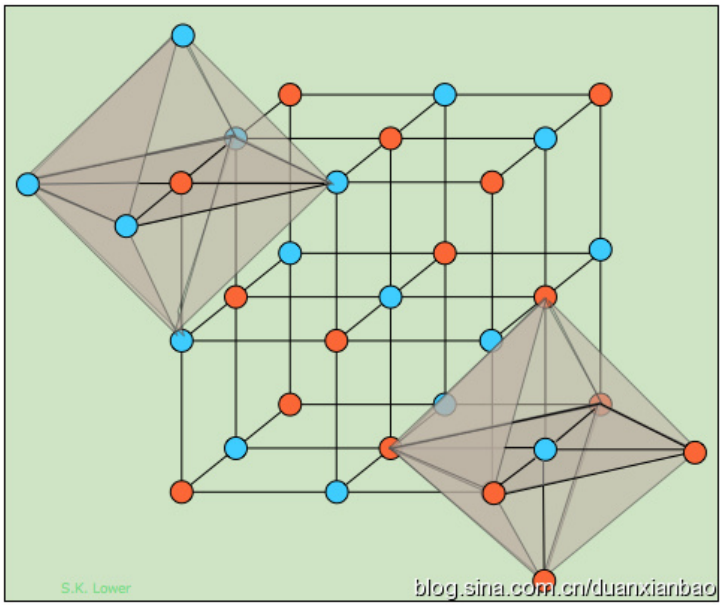
1. 程序创建 data file 文件

根据合金体系的结构，按着 data file 的文件格式编程产生相关 data file 文件，由 read_data 命令读入即可。

可以说，这是一种万能的方法，具体不多说，主要是要了解 data file 的格式，并且有一定的编程基础。

2. lammps 命令创建

lammps 提供的命令也能很方便的创建合金结构，当然可能仅限于一些比较规则的晶体结构。这里以 B1 结构(rocksalt)的 NaCl 为例进行说明（NaCl 显然不是合金，但合金类似）。



创建结构的命令不外乎 lattice, create_box, create_atoms，这里只是强调在合金体系创建的应用。

先直接给出命令，如下：

```
lattice         custom $x a1 1.0 0.0 0.0 a2 0.0 1.0 0.0 a3 0.0 0.0 1.0 &
                basis       0.0 0.0 0.0 basis 0.5 0.5 0.0 basis 0.5 0.0 0.5
                basis       0.5 0.5 0.5 basis 0.0 0.0 0.5 basis 0.0 0.5 0.0
region          box block 0 3 0 3 0 3
create_box      2 box
create_atoms    2 box basis 1 1 basis 2 1 basis 3 1 basis 4 1 &
                basis       5 2 basis 6 2 basis 7 2 basis 8 2
```

具体说明：

(1) lattice 第一行为晶格矢量，其中$x 为晶格矢量；第二行和第三行每一个 basis 对应原胞中的一个原子。对于 B1 结构，是包含 8 个原子，即 4 个 Na，4 个 Cl。
(2) region 是定义盒子的区域

(3) create_box 定义盒子，指定原子类型的数量。这里的 2 是因为有两种类型。
(4) create_atoms 在盒子中，以 lattice 晶格创建原子。这里的 basis 有两个参数，第一个指定是 lattice 中的哪一个原子，这里从 1 到 8；第二个参数指定原子类型，因为有两种类型，所以为 1 或者 2.

建议可以先看 manual，熟悉上面涉及到的几个命令。

- lattice: http://lammps.sandia.gov/doc/lattice.html
- create_box: http://lammps.sandia.gov/doc/create_box.html
- create_atoms: http://lammps.sandia.gov/doc/create_atoms.html

这里，我们看到只是指定了两种原子类型，但并没有告诉程序是哪两种，它怎么会知道呢？
所以这需要在后面的势参数中指定。

lammps 输出的模型如何导入 MS 中建模

由 www.52souji.net 发表于 2013 年 4 月 28 日 || 2,645 浏览

前面我已经介绍了如何将 MS（materials studio）建立的模型导入到 lammps 中。

参考：Materials Studio 构建的模型如何导入 lammps？

这里提出一个相反的问题：如何将 lammps 输出的模型再反过来导入到 MS 中呢？

一般 lammps 的 dump 命令输出的数据文件格式有三种：dump 格式、xyz 格式和 cfg 格式。

- dump 格式是 lammps 自己定义的，并不具有通用性，是很难转换成其他通用格式的。
- xyz 格式虽然很通用，但包含的信息却不完整，丢失了晶胞的信息。
- cfg 格式也不是一种通用格式，但因为有一些现成的程序可以帮助转换，所以使用起来还比较方便。

所以，这里面实际上是借助 cfg 格式作为桥梁，进行格式转换的。

总的思路是：lammps 输出 cfg 格式 →pdb 格式 → 导入 ms 建模

内容目录 [隐藏]

一. lammps 输出 cfg 格式

按着 lammps 中 dump 命令的格式要求输出即可，如下：

```
dump 1 all cfg 100 dump.snap.*.cfg id type xs ys zs
```

有时需要辅助 dump_modify 命令，用来设置其中的元素类型。

二. 转换成 pdb 格式

cfg 格式是李巨提出来的，他也提供了 cfg 转换成 pdb 的脚本（非开源）。

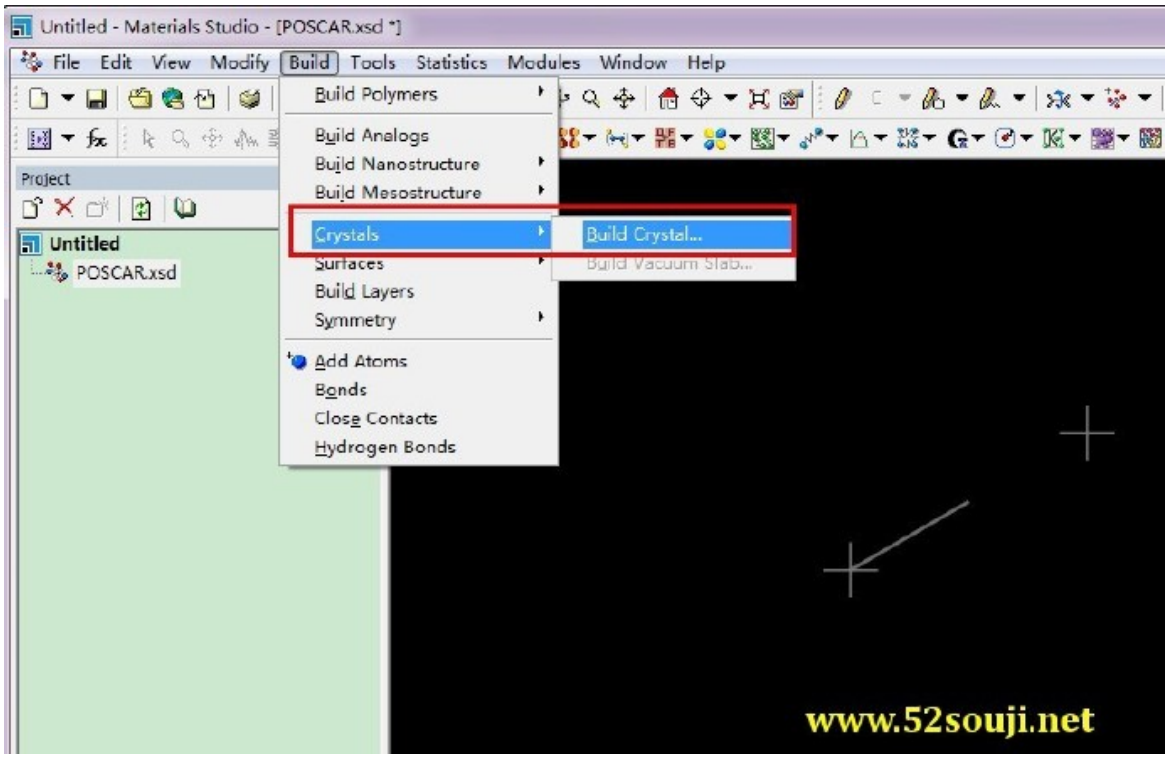
linux 版下载地址：cfg2pdb | 更多下载

使用很简单，如下即可完成转换：

```
$ cfg2pdb input.cfg output.pdb
```

三. 导入 MS 完成建模

将转换完成的 pdb 文件用 MS 打开即可完成建模。如果看到模型有点不对，可能需要先进行 build crystal 操作，如下图所示。



这样就实现了 lammps 输出文件到 MS 的无缝连接。

Materials Studio 构建的模型如何导入 lammps？

由 www.52souji.net 发表于 2013 年 4 月 23 日 || 4,031 浏览

经常碰到这类问题，诸如“MS 建的模型如何自动转换成 lammps 的数据文件？”，“MS 建的模型如何导入 lammps？”等等。以前也没有很好的办法，分子模拟论坛里说有一个叫做 msi2lmp 的小程序可以实现，但我自己没有搞成功。不过我自己摸索了下面的方法，如果你不觉得麻烦，不妨试试。

- 总的思路是：模型 → cif 格式 → vasp 格式 → lammps 格式
- 需要用到的软件：Materials Studio，VESTA，poscar2lammps 脚本

具体步骤如下所示。

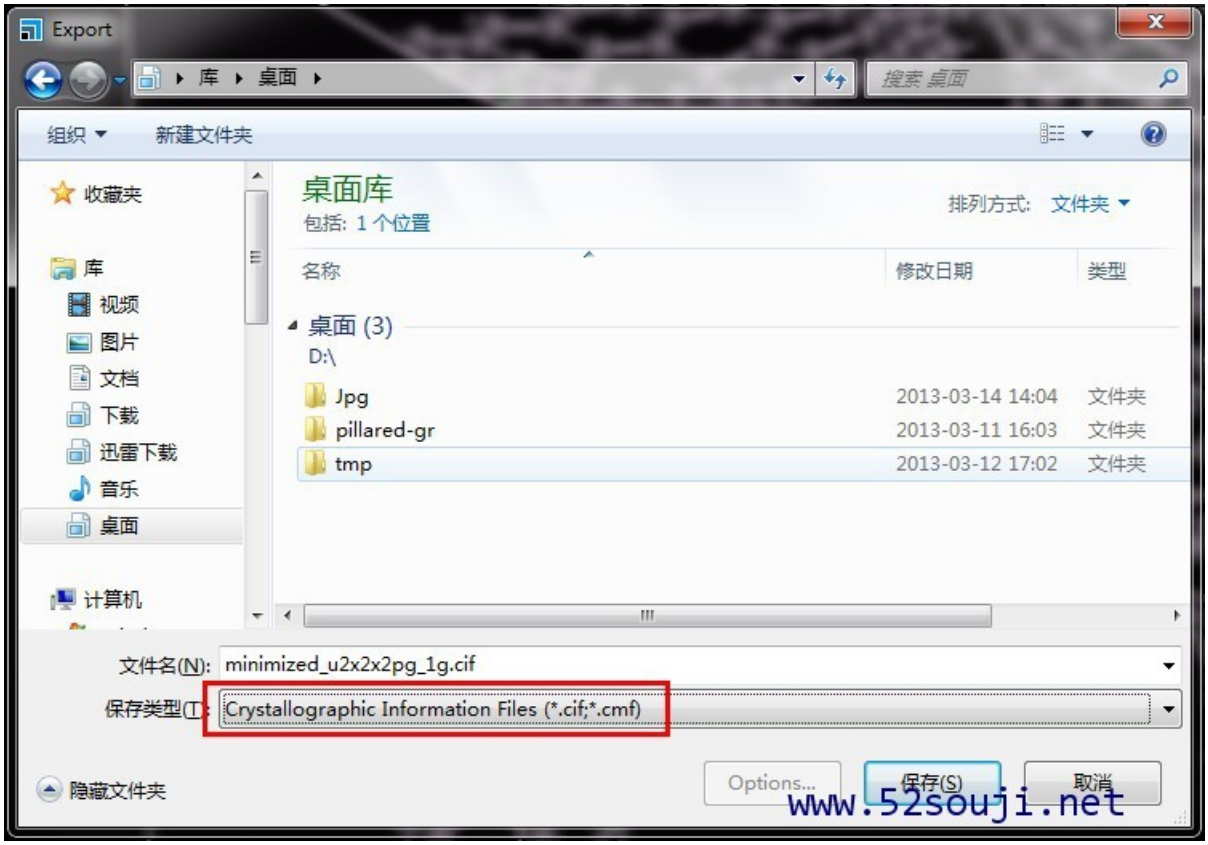
内容目录 [隐藏]

1. 建模

使用 MS（Materials Studio）完成模型的建立。需要特别注意，模型中必须包括晶胞信息，不然在下一步就会提示不能导出 cif 格式。可以使用 build–crsytals–build crystals 命令为没有晶胞的模型创建晶胞。

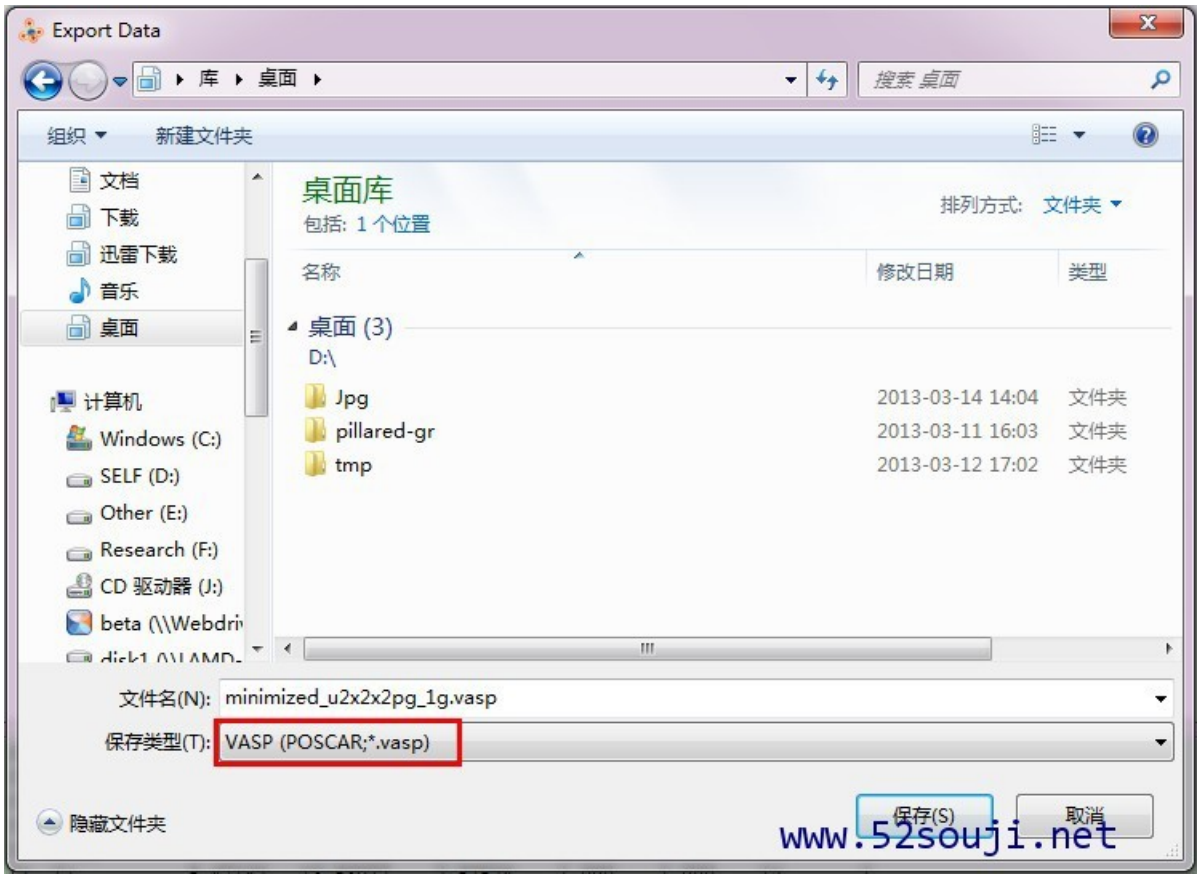
2. 导出 cif 格式

选择菜单【File】–【Export】，弹出的对话框中选择 cif 格式，导出。如下图所示。



3. 转换为 vasp 格式

使用 VESTA 打开第二步导出的 cif 格式的文件，选择菜单【File】–【Export Data】，弹出的对话框中选择 vasp 格式，导出。如下图所示。



VESTA 程序介绍：结构模型可视化软件 VESTA

4. 转换为 lammps 格式

将第三步中 vasp 格式的文件转移到 linux 操作系统中，使用 poscar2lammps 脚本将其转换成 lammps 格式。

poscar2lammps 脚本下载：vasp poscar 结构转 lammps 结构文件

完成。

一点说明

仅限于没有力场作用的体系，对于有力场的情形，我猜测这种方案是行不通的。

LAMMPS 如何定义六角密堆结构 HCP

由 www.52souji.net 发表于 2012 年 8 月 16 日 || 2,757 浏览

因为六角密堆结构的截面是六边形的，所以当我们在构建模拟盒子的时候，有可能因为觉得边界周期性不好处理，而觉得不知道如何构建。

而实际上，我们的考虑是多余的，因为 LAMMPS 为我们提供的 HCP 的原胞已经将这个问题很好的解决了。因此我们在使用时，只需要按着如下定义：

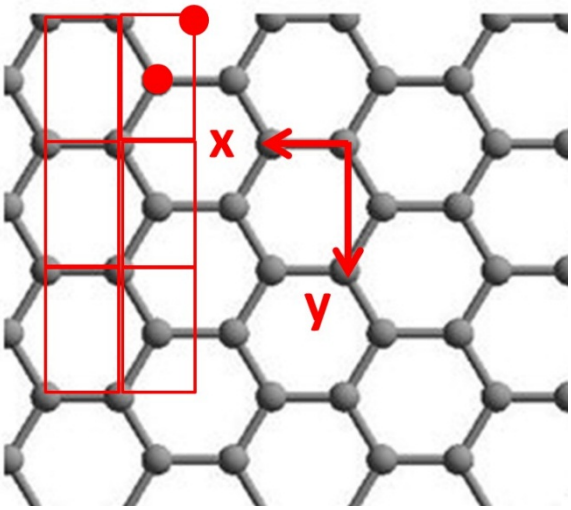
```
01      lattice hcp alength
```

其中的 alength 就是晶格常数 a 的长度。

下面是 LAMMPS-lattice 命令中对 HCP 的说明：

Style hcp has a1 = 1 0 0, a2 = 0 sqrt(3) 0, and a3 = 0 0 sqrt(8/3)...A hcp lattice has 4 basis atoms, two in the z = 0 plane and 2 in the z = 0.5 plane.

这里可以看到，LAMMPS 在选取的是正交的坐标系，而不是通常用来定义 HCP 的四轴坐标系中的三轴。下图展示了 x-y 平面上的坐标轴。



红色的长方形盒子即为原胞在该平面的投影，该盒子在 x-y 平面包含 2 个原子，整个盒子包含 4 个原子。这样的盒子的周期性是很容易理解的。

另外，有些人可能会有疑问：HCP 的晶格常数有两个：a 和 c，为什么这里提到了 a ？

确实如此，LAMMPS 已经将 c/a 给固定了（不然它没办法确定 z 轴）。对于 c/a 不为这个固定比值的情况，只需要对结构进行弛豫即可。

lammps 的 restart 重启动相关命令

由 www.52souji.net 发表于 2012 年 5 月 3 日 || 6,045 浏览

这是我以前写过的一篇博文，发表在新浪博客上，转载过来，并做适当修正。

内容目录 [隐藏]

restart 作用

restart 命令是 lammps 提供的重启动命令，用于重启 lammps 计算。这样可以省去一些共有的计算，从而节省计算时间，特别是对于比较大的体系。

比如原子弛豫一般是各种计算都必须要做的，那么就可以先单独弛豫一下，生成 restart 文件，供其他计算直接调用，从而避免每一个计算都进行弛豫。

restart 原理

重启动的原理很简单，就是写一个中间 restart 文件，在这个文件中记录某个状态的绝大部分信息，在重启动时，只需要读入这个文件，接着完成剩下的计算即可。restart 文件是二进制文件。

restart 命令

restart 相关的命令有三个：restart, write_restart, read_restart。还有一个相关的脚本在 lammps 的 tools 目录下 restart2data，可以用来将二进制的 restart 文件转换成可识别的文文文件，即 data file 文件，可以由 read_data 读入。

restart 命令和 write_restart 命令是用来写重启动文件的，区别是前者用来周期性地将系统状态写入不同的 restart 文件，而后者则是写出该命令使用前的系统状态。

read_restart 命名是用来读入 restart 文件，开始新的计算。

tools 目录下的 restart2data 工具提供的是 cpp 源文件，需要编译后才可使用。

下面以 lammps/bench 目录下的输入文件为例进行简要介绍。

写 restart 文件

```
# FENE beadspring benchmark

units lj
atom_style bond
special_bonds fene

read_data data.chain

neighbor 0.4 bin
neigh_modify every 1 delay 1

bond_style fene
bond_coeff 1 30.0 1.5 1.0 1.0

pair_style lj/cut 1.12
pair_modify shift yes
pair_coeff 1 1 1.0 1.0 1.12

```

```
fix 1 all nve
fix 2 all langevin 1.0 1.0 10.0 904297

thermo 100
timestep 0.012

restart 50 tmp.restart # write restart file periodiclly
run 100
#write_restart tmp*.restart # write restart file of the current state
```

注意 restart 和 write_restart 命令的使用位置是不同的。这里使用 restart 命令，会产生两个文件 tmp.restart.50, tmp.restart.100.

转换 restart 文件

首先编译 restar2data.cpp，如下：

```
$ g++ restart2data.cpp -o restart2data
```

这时就会在当前目录下产生可执行的 restart2data 文件，cp 到 bench 目录下，转换上面例子产生的 tmp50.restart 文件。

```
restart2data tmp50.restart data.tmp50
```

这时就会在 bench 目录下产生 data.tmp50 文件，它是可直接辨识的文本文件。

读入 restart 文件

把 in.chain 文件拷贝成 in.chain.restart 文件，用来测试 restart。由于 restart 文件中包含了很多的命令，所以需要重新设置的命令不多，这里如下：

```
read_restart    tmp.restart.50

neighbor        0.4 bin
neigh_modify    every 1 delay 1

fix             1 all nve
fix             2 all langevin 1.0 1.0 10.0 904297

timestep        0.012

run             50
```

这里实际就是接着 timestep 等于 50 的那个状态重启计算的。

参考：restart, write_restart, read_restart, restart2data, Restarting a simulation

【格式转换】vasp poscar 结构转 lammps 结构文件

由 www.52souji.net 发表于 2013 年 2 月 22 日 || 2,638 浏览

在计算材料学中，结构数据文件格式转换是很常见的事情，这里给大家推荐一个脚本可以轻松实现 VASP POSCAR/CONTCAR 到 lammps 的 data file 文件的转换。

内容目录 [隐藏]

功能简介

- 实现 VASP POSCAR/CONTCAR 到 lammps 的 data file 文件的转换
- 支持 VASP4.6 和 5.2（包括原子类型）的格式
- 支持分数坐标(Direct)和绝对坐标(Cartesian)
- 支持非正交原胞

使用方法

1. 把脚本 VASP-poscar2lammps.awk 下载下来，拷贝到你的当前目录（或者添加到系统路径中），修改权限为可执行（chmod +x）
2. 使用如下命令完成转换

```
VASP-poscar2lammps.awk f.POSCAR > f.lammps
```

其中，VASP-poscar2lammps.awk 为转换脚本名称，f.POSCAR 为具有 VASP POSCAR 结构的文件的文件名，f.lammps 为转换的 LAMMPS 结构文件的文件名。

下载

地址：百度网盘

一点说明

该脚本是使用 awk，所以 awk 必须先在目录/bin 下，如果在其他目录下，需要修改脚本的第一行。

另外，该脚本版权归原作者。如果侵权，请给我爱搜集网留言（http://www.52souji.net）删除。

参考来源：https://sites.google.com/site/pmitev/academic/vasp-poscar2lammps-awk

如何使用 atomeye 将 lammps 的模拟结果做成视频动画

由 www.52souji.net 发表于 2013 年 9 月 23 日 || 1,937 浏览

有些时候，会希望将 lammps 分子动力学模拟的过程制作成动画，这样展示起来会比较形象直接。那么如何做到呢？

方法并不唯一，这里介绍如何使用 atomeye 软件将 lammps 的模拟结果做成视频动画。

至于 atomeye 软件，这里就不做过多介绍，不了解的可以翻看我之前写过的博文：

- 可视化软件 atomeye 简介、使用方法及常用快捷键

内容目录 [隐藏]

1. 输出 cfg 文件

在 lammps 中使用 dump cfg 命令，输出一些列的 cfg 格式的文件。

这里的每一个文件就代表要做成的动画中的一帧，所以如果你希望动画比较连续，可以把输出的频率设大一些。

命令的使用方式如下所示：

```
01      dump 2 inner cfg 10 dump.snap.*.cfg id type xs ys zs
```

2. cfg 文件转 jpg 图像文件

使用 atomeye 软件打开上面输出的这一些列 cfg 文件中的第一个，调整构型到一个合适的姿态，然后按'y'，atomeye 就会以相同的姿态，依次输出各个构型的 jpg 图片。

在 atomeye 输出的过程中，看上去就已经是在播放动画了。

3. 将 jpg 图像合成视频动画

图像合成视频的方法也很多，这里推荐大家使用 movgear 软件合成。

该软件使用起来非常傻瓜，可以很容易将一些列图片合成 gif 动态图片。网上很容易下到，这里暂时不提供下载。

这样就完成了视频动画的制作啦！

【LAMMPS 如何系列】计算平衡晶格常数

由 www.52souji.net 发表于 2012 年 5 月 28 日 || 3,552 浏览

平衡晶格常数（equilibrium lattice constant）对应的体系能量是最低的，因此只需要计算一些列不同晶格常数下体系的能量，那么体系能量最小时对应的晶格常数就是平衡晶格常数。

在初次计算时，建议将晶格常数增加的步长取得大一点，比如 0.1A，这样通过结果数据就可以大致确定晶格常数的范围。

然后再将晶格常数的范围缩小，减小步长，比如 0.01A，这样做就可以获得精度比较高的结果。

下面以金刚石为例介绍具体如何计算平衡晶格常数。

输入文件： in.diamond

```
01  # This input script is used to calculate
02  # the lattice constant of diamond
03  # Powered by Xianbao Duan
04  # Email: xianbao.d@gmail.com
05  # Website: http://www.52souji.net/
06
07  units       metal
08  boundary        p p p
09  atom_style      atomic
10
11  variable        i loop 20
12  #variable       x equal 2.5+0.1*$i
```

```
variable        x equal 3.5+0.01*$i

# build the model
lattice         diamond $x
region          box block 0 10 0 10 0 10
create_box      1 box
create_atoms    1 box

# specify the potential
pair_style      tersoff
pair_coeff      * * SiC.tersoff C
mass            1 12

variable        n equal count(all)
variable        P equal pe/$n
#variable       v equal vol

timestep        0.005
thermo          10

# minimize the total energy
```

```
min_style       cg
minimize        1.0e-12 1.0e-12 1000 1000

print           "@ $x $P"

# loop
clear
next            i
jump            in.diamond


```

脚本很简单，只要一个简单的能量最小化就可以了，然后输出晶格常数和结合能 log 文件中。

然后从 log 文件中将晶格常数和结合能的数据提取出来。

```
$ grep ^@ log.lammps > lat.vs.Ecoh.step1
```

grep 是一个 linux 命令，上面的意思就是将 log.lammps 文件中以@开始的行输出到 lat.vs.Ecoh.step1 文件中。

得到的文件中，每一行开头会有一个@，通过列编辑器一起删掉，就可以获得有如下内容的文件。

```
3.51 -7.35079243
```

```
02      3.52 -7.356562846
03      3.53 -7.361141466
04      3.54 -7.364556691
05      3.55 -7.366836396
06      3.56 -7.368007936
07      3.57 -7.368098161
08      3.58 -7.367133418
09      3.59 -7.365139565
10      3.6 -7.362141974
11      3.61 -7.358165545
12      3.62 -7.353234707
13      3.63 -7.347373432
14      3.64 -7.340605242
15      3.65 -7.332953213
16      3.66 -7.324439984
17      3.67 -7.315087768
18      3.68 -7.304918353
19      3.69 -7.293953116
20      3.7 -7.282213024
```

上面的数据实际上对应的就是结合能曲线，其能量最低点对应的就是平衡晶格常数。当然，能量最低点的能量就是晶体的结合能。

具体如何从上面的数据中获得最终结果，可以参考我之前写的一篇文章：
MATLAB 计算平衡晶格常数。
晶格常数表可以参考：所有元素的晶格常数查询表。

【LAMMPS 如何系列】计算体积模量

由 www.52souji.net 发表于 2012 年 5 月 28 日 || 4,029 浏览

（在阅读本文前，建议你先阅读：【LAMMPS 如何系列】计算平衡晶格常数）

体积模量（Bulk Modulus）是材料很常用的一个属性，下面是维基百科里的解释。

The **bulk modulus** (*K*) of a substance measures the substance's resistance to uniform compression. It is defined as the pressure increase needed to decrease the volume by a factor of 1/e.　(From wikipedia: Bulk Modulus)

$$B \equiv -\frac{dP}{dV/V}$$

即使一种材料体积减小一定程度时需要的增加的压强。

推导体积模量另一种形式的表达式

对于立方原胞，压强可以有如下定义：

$$P = -\frac{d\varepsilon}{dV} = -\frac{M}{3a^2}\frac{dE}{da}$$

上式中，E 是晶胞总能量，M 是晶胞中的原子数，a 是晶格常数。

将上式代入体积模量的定义式，可以得到体积模量的另外一种表达：

$$B = \frac{M}{9a_0}\frac{d^2E}{da^2}\bigg|_{a_0}$$

其中 a0 是平衡晶格常数。

分析以上表达式，晶胞中的原子数 M 很容易得到，平衡晶格常数在前面的文章已经介绍如何计算，而 d2E/da2|a0 只是在计算平衡晶格常数时进一步求二次导数就可以获得。因此，在计算了平衡晶格常数后，并不需要进行新的计算就可以获得体积模量。

计算体积模量

下面接着计算平衡晶格常数的计算，进一步计算体积模量，仍然以金刚石为例。金刚石为 diamond 类型，每个原胞中有 8 个原子，即 M=8。仍然使用计算平衡晶格常数中的晶格常数-结合能数据。

下面的 MATLAB 计算程序可以直接给出体积模量。

cal_bulk_modulus.m

```
% calcuate the bulk modulus according to "lat_const cohesive energy"
%
% Paramters:
% N: input. int. order of the polynomial fitting.
% M: number of atoms in the unit cell
%       sc: M=1;    bcc: M=2;   fcc: M=4;   dc: M=8
% inFileNmae: input. string. name of the file storing "lat_const cohesive energy"
%
% Example:
% cal_bulk_modulus(6,8,'Au')
%
% Create: 2012-1-9     Complete: 2012-1-9
% Poweed by Xianbao Duan @ Lab. of Advanced Material Design
% Email: Xianbao.d at gmail.com
% ********************************************************************
```

```
 function cal_bulk_modulus(N,M,inFileName)


cvt_factor = 160.22;        % convert the modulus from eV/A^2 to GPa


data = load(inFileName,'-ascii');       % read in data from the file
x = data(:,1);  y = data(:,2);
% [x y] =  textread(inFileName,'%f %f');
bindEnergy = polyfit(x,y,N);    % polynomial fitting
dbindEnergy = polyder(bindEnergy);  % derivation of the polynomial equation
zero_points = roots(dbindEnergy);    % solve the zero points
for i = 1: length(zero_points)
   if isreal(zero_points(i))
      if zero_points(i) > x(1)
         if zero_points(i) < x(end)
            lat_const = zero_points(i)
            coh_energy = spline(x,y,lat_const)
         end
      end
   end
end
d2 = polyder(dbindEnergy);
d2_da = polyval(d2,lat_const);
bulk_modulus = M*d2_da*cvt_factor/(9*lat_const)
```

已经注释的比较清楚了，不做更多解释了。

如果使用 4 阶拟合（即 N=4），得到金刚石的体积模量为：425.7265GPa，实验值为 442 GPa。

【LAMMPS 如何系列】计算体积热容量

由 www.52souji.net 发表于 2012 年 5 月 30 日 || 3,582 浏览

体积热容简介

热容(Heat Capacity, or thermal capacity)，单位质量或体积的物质温度升高或降低 1K 时吸收或放出的热量。

Wikipedia 里热容的定义：Heat capacity (usually denoted by a capital C, often with subscripts), or thermal capacity, is the measurable physical quantity that characterizes the amount of heat required to change a substance's temperature by a given amount. (link:Heat capacity)

从上面的定义也可以看出，热容有质量热容 (Specific Heat) 和体积热容 (Volumetric Heat Capacity)，分别是以单位质量和单位体积的物质定义的。这里具体介绍体积热容如何计算。

Wikipedia 里体积热容的定义：Volumetric heat capacity (VHC), also termed volume-specific heat capacity, describes the ability of a given volume of a substance to store internal energy while undergoing a given temperature change, but without undergoing a phase change. (link: Volumetric heat capacity)

可以总结出如下公式：

$$C_V = \frac{\Delta E}{\Delta T \cdot V}$$

简单分析一下，既然是单位体积的物质，那么体积在模拟的过程中就不能变；然后考虑体系能量随着温度的变化，自然就会想到选择 NVT 系综。

更具体的描述是：在保持体系体积不变的前提下，计算不同温度下对应的体系总能。E-T 曲线的斜率应该就与体积热容有一个直接的对应关系了。

体积热容计算举例

下面以 Cu 为例，介绍如何使用 LAMMPS 计算其体积热容 Cv。

输入脚本 in.specific.heat

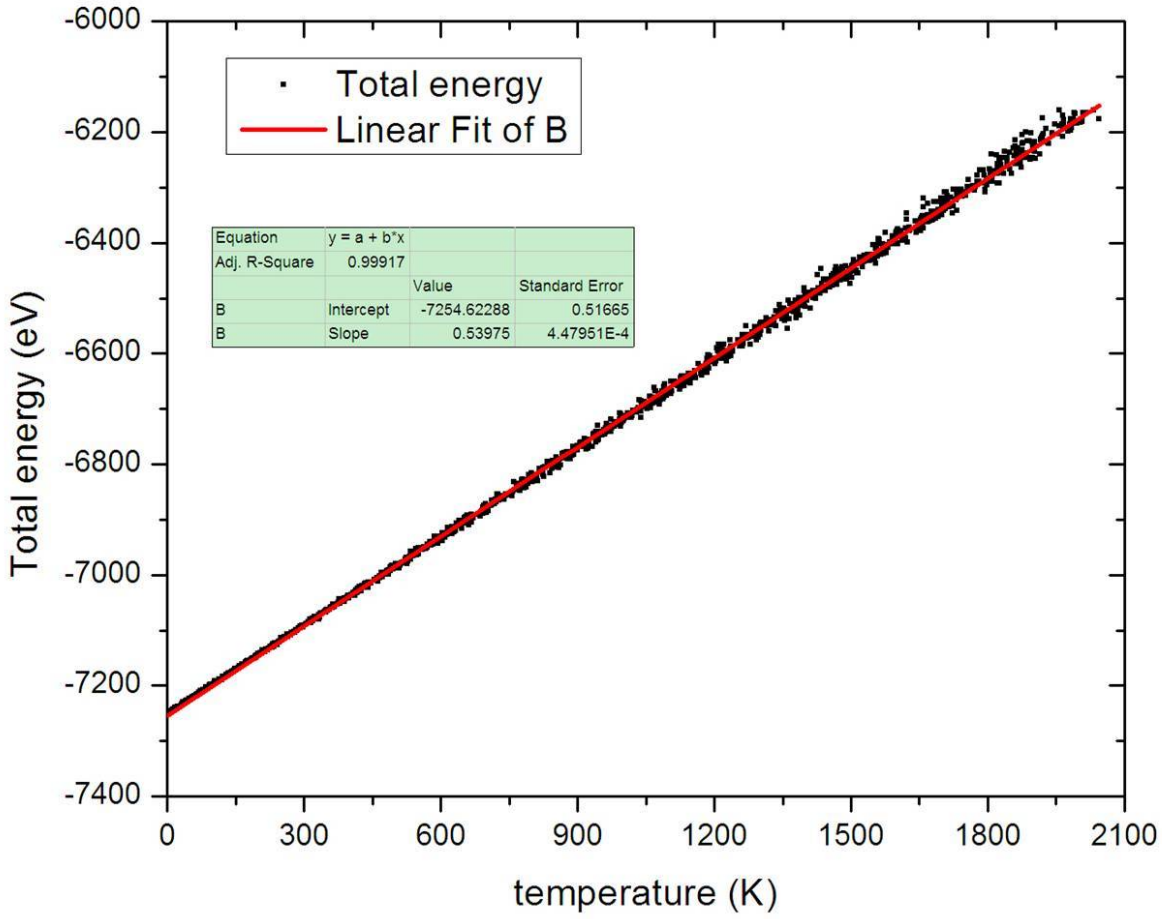
```
# This input script is used to calculate
# the specific heat of copper.
# Powered by Xianbao Duan
# Email: xianbao.d@gmail.com
# Website: http://www.52souji.net/

units     metal
boundary        p p p
atom_style      atomic

variable        x equal 2.5

lattice     fcc 3.62
region      box block 0 8 0 8 0 8
create_box  1 box
create_atoms    1 box

```

```
17      pair_style      eam
18      pair_coeff      1 1 Cu_u3.eam
19
20      variable        N equal step
21      variable        Etotal equal etotal
22      variable        T equal temp
23      variable        V equal vol
24
25      velocity        all create $x 825577 dist gaussian
26      fix             extra all print 100 "${N} ${V}  ${T}  ${Etotal} " file data
27
28      timestep    0.002
29      thermo      1000
30
31      fix             1 all nvt temp $x 2000 0.2
32      run             120000
33
```

很简单的输入脚本，在 34~35 行定义了 NVT 系综，温度从初始温度 2.5K 一直升高到 2000K，整个过程运行 120000 步。

输出的文件 data 中包含了所有的数据，分别是计算步，体积，温度和总能量。

将数据导入到 origin 中，进行简单的线性拟合分析，即可得到下图。



图中拟合曲线的斜率为 0.53975，即 $\Delta E/\Delta T=0.53975$ eV/K。计算过程中保持恒定的体积 V=24288.21914 A^3 。

从而，可以计算得到 Cv=3.56 J/(cm^3K) 。实验值为：3.45 J/(cm^3K)。

(其中用到了单位换算：1 eV = $1.60217646 \times 10^{-19}$ J, 1 A = 10^{-8} cm)

体积热容表可以参考：体积热容量查询表

【LAMMPS 如何系列】计算热膨胀系数

由 www.52souji.net 发表于 2012 年 6 月 8 日 || 5,136 浏览

热膨胀系数（thermal expansion coefficent）是物质因温度改变时，体积发生变化的趋势。

维基百科的定义：Thermal expansion is the tendency of matter to change in volume in response to a change in temperature. (Link: thermal expansion)

热膨胀系数有体膨胀系数 $\beta=\Delta V/(V*\Delta T)$ 和 线膨胀系数 $\alpha=\Delta L/(L*\Delta T)$ 。

从这个定义大致就可以获得计算热膨胀系数的思路：计算不同温度下物质的体积或长度，那么体积或长度随着温度的变化曲线的斜率就是 $\Delta L/\Delta T$ 了。所以，在模拟的过程中，体积是变化的，那么自然就会想到使用 NPT 系综。在使用 NPT 系综之前和之后，建议使用 NVT 系综进行平衡一定的步数。

举例计算热膨胀系数

下面仍然以铜为例，具体介绍如何使用 LAMMPS 计算线膨胀系数 α。

输入脚本：in.thermal.expansion

```
# This input script is used to calculate
# the thermal expansion of copper.
# Powered by Xianbao Duan
# Email: xianbao.d@gmail.com
# Website: http://www.52souji.net/

units          metal
boundary       p p p
atom_style     atomic

variable   i loop 10
variable       x equal 200+100*$i

lattice                fcc 3.62
region                 box block 0 8 0 8 0 8
create_box             1 box
```

```
create_atoms          1 box

pair_style      eam
pair_coeff      1 1 Cu_u3.eam

variable        N equal step
variable        pote equal pe
variable        Etotal equal etotal
variable        T equal temp
variable        Press equal press
variable        V equal vol

velocity        all create 2.5 825577 dist gaussian

timestep   0.002
thermo                  1000

fix                     1 all nvt temp 2.5 2.5 0.2
run                     2000

unfix                   1

fix             2 all npt temp 2.5 $x 0.2 iso 0 0 10
run             120000

unfix                   2
```

```
fix         extra all print 100 "${N} ${T} ${V} ${pote} ${Etotal} ${Press}" append data

fix             3 all nvt temp $x $x 0.2
run             2000

unfix           3
unfix           extra

clear
next            i
jump            in.thermal.expansion
```

对以上脚本进行简单解释：

1. 第 12 行定义的变量 x 为温度，即计算这些温度下体系的晶格常数。

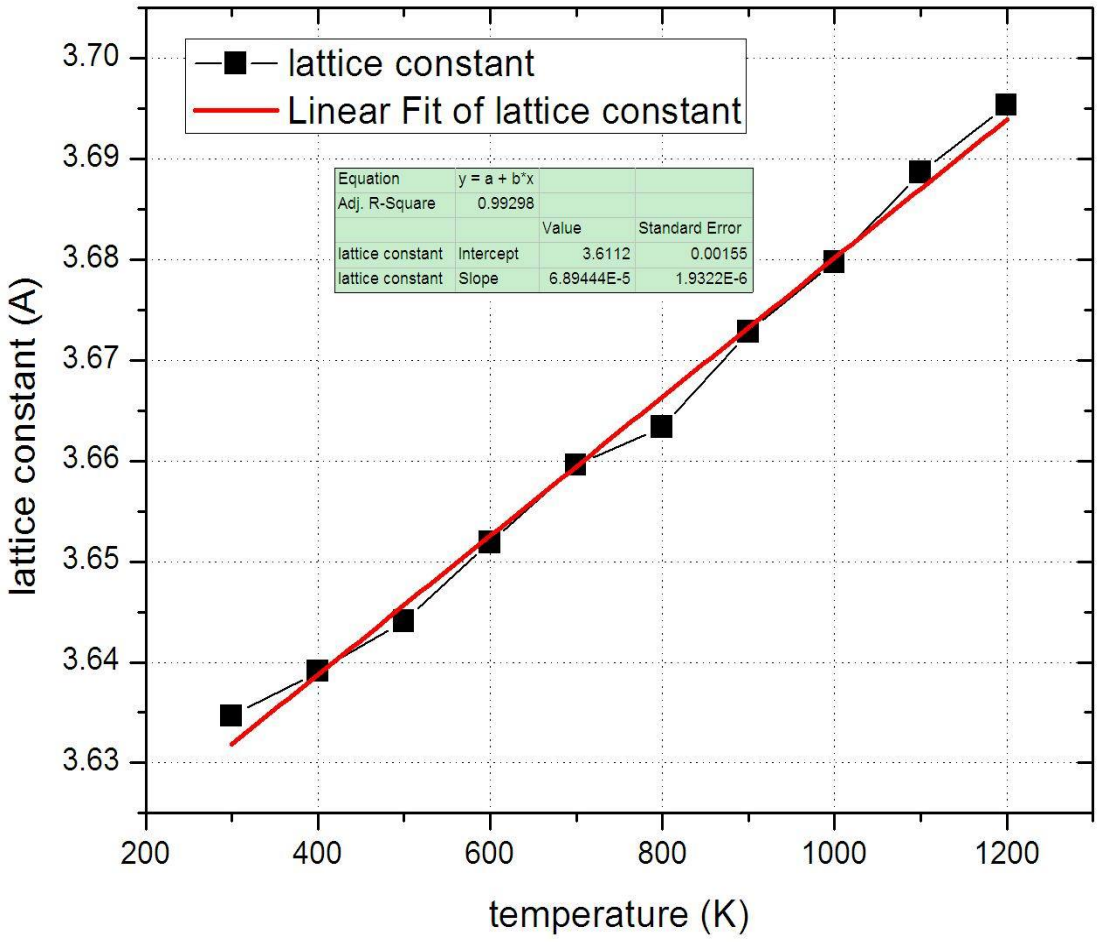
2. 第 34-35 行将体系在 2.5K 下进行 NVT 平衡，为 NPT 升温过程做准备。

3. 第 39-40 行将体系在 NPT 系综下进行升温。

4. 第 44 行定义输出有关变量。注意这里使用的是 append 关键字，而不是 file，是为了让脚本在循环执行时不重写这个文件。

5. 第 46-47 行将升温后的体系在 NVT 下进行适当平衡，在这个过程中输出了有关变量到文件中。

得到的 data 文件即为结果数据。将其中的温度和体积复制出来列表，绘图，如下图。



图中曲线的斜率是 6.89444e-5，即 $\Delta L/\Delta T$= 6.89444e-5 A/K。又初始的晶格常数大致为 3.6346A，所以可以算出线膨胀系数 $\alpha=\Delta L/(L*\Delta T)$ =18.97 10^{-6} K^{-1}。
实验结果为 17.5 10^{-6} K^{-1}，两者基本吻合。
材料的热膨胀系数可以参考：热膨胀系数查询表

LAMMPS 常用命令中文翻译索引

- 没有列出的命令目前不在博主计划翻译的范围内。
- 没有链接的命令是博主要翻译但还没有完成的，会在翻译之后加上链接。

初始化

atom_style, boundary, dimension, newton, processors, units

建模

create_atoms, create_box, lattice, read_data, read_dump, read_restart, region, replicate

力场

pair_coeff, pair_modify, pair_style, pair_write

设置

communicate, group, mass, min_modify, min_style, neigh_modify, neighbor,reset_timestep, run_style, set, timestep, velocity

约束

fix, fix_modify, unfix

计算

compute, compute_modify, uncompute

输出

dump, dump image, dump_modify, restart, thermo, thermo_modify, thermo_style, undump, write_data, write_restart

操作

delete_atoms, delete_bonds, displace_atoms, change_box, minimize, neb, prd, rerun, run, temper

其他混杂

clear, echo, if, include, jump, label, log, next, print, shell, variable

元素热膨胀系数查询表

由 www.52souji.net 发表于 2012 年 5 月 27 日 || 3,233 浏览

以下是元素的热膨胀系数，来自维基百科：thermal expansion coefficients 。

其中 CRC, LNG, WEL 等代表不同的数据来源，一般没有区别，或区别很小。（发布出来，发现格式有点乱，不过应该不会不影响你找到你需要的数据，我就不花时间去调整了）

$\times 10^{-6}\ m \cdot m^{-1} \cdot K^{-1}$

3 Li lithium

use	46
CRC	46
LNG	46
WEL	46

4 Be beryllium

use	11.3
CRC	11.3
LNG	11.3
WEL	11.3

5 B boron

use	5–7
LNG	5–7
WEL	6

6 C carbon

use	
WEL	7.1

11 Na sodium

use	71
CRC	71
LNG	71
WEL	71

12 Mg magnesium

use	24.8
CRC	24.8
LNG	24.8
WEL	8.2

13 Al aluminium

use	23.1
CRC	23.1
LNG	23.1
WEL	23.1

14 Si silicon	
use	2.6
WEL	2.6
19 K potassium	
use	83.3
CRC	83.3
20 Ca calcium	
use	22.3
CRC	22.3
LNG	22.3
WEL	22.3
21 Sc scandium	
use	(room temperature) (alpha, polycrystalline) 10.2
CRC	10.2
CR2	(room temperature) (alpha, amorphous) 7.6
CR2	(room temperature) (alpha, crystalline) 15.3
CR2	(room temperature) (alpha, polycrystalline) 10.2
LNG	10.2
WEL	10.2
22 Ti titanium	
use	8.6
CRC	8.6
LNG	8.6
WEL	8.6
23 V vanadium	
use	8.4

CRC	8.4
LNG	8.4
WEL	8.4

24 Cr chromium

use	4.9
CRC	4.9
LNG	4.9
WEL	4.9

25 Mn manganese

use	21.7
CRC	21.7
LNG	21.7
WEL	21.7

26 Fe iron

use	11.8
CRC	11.8
LNG	11.8
WEL	11.8

27 Co cobalt

use	13.0
CRC	13.0
LNG	13.0
WEL	13.0

28 Ni nickel

use	13.4
CRC	13.4

LNG	13.4
WEL	13.4
	29 Cu copper
use	16.5
CRC	16.5
LNG	16.5
WEL	16.5
	30 Zn zinc
use	30.2
CRC	30.2
LNG	30.2
WEL	30.2
	31 Ga gallium
use	
CRC	18[1]
LNG	(>30 C) (liquid) 120
WEL	(>30 C) (liquid) 120
	32 Ge germanium
use	6.0
LNG	6.0
WEL	6
	34 Se selenium
use	(amorphous) 37
LNG	(amorphous) 37
	38 Sr strontium
use	22.5

CRC	22.5
LNG	22.5
WEL	22.5

39 Y yttrium

use	(room temperature) (alpha, polycrystalline) 10.6
CRC	10.6
CR2	(room temperature) (alpha, amorphous) 6.0
CR2	(room temperature) (alpha, crystalline) 19.7
CR2	(room temperature) (alpha, polycrystalline) 10.6
LNG	10.6
WEL	10.6

40 Zr zirconium

use	5.7
CRC	5.7
LNG	5.7
WEL	5.7

41 Nb niobium

use	7.3
CRC	7.3
LNG	7.3
WEL	7.3

42 Mo molybdenum

use	4.8
CRC	4.8
LNG	4.8
WEL	4.8

44 Ru ruthenium	
use	6.4
CRC	6.4
LNG	6.4
WEL	6.4
45 Rh rhodium	
use	8.2
CRC	8.2
LNG	8.2
WEL	8.2
46 Pd palladium	
use	11.8
CRC	11.8
LNG	11.8
WEL	11.8
47 Ag silver	
use	18.9
CRC	18.9
LNG	18.9
WEL	18.9
48 Cd cadmium	
use	30.8
CRC	30.8
LNG	30.8
WEL	30.8
49 In indium	

use	32.1
CRC	32.1
LNG	32.1
WEL	32.1

50 Sn tin

use	22.0
CRC	22.0
LNG	22.0
WEL	22

51 Sb antimony

use	11.0
CRC	11.0
LNG	11.0
WEL	11

55 Cs caesium

use	97
CRC	97

56 Ba barium

use	20.6
CRC	20.6
LNG	20.6
WEL	20.6

57 La lanthanum

use	(room temperature) (alpha, polycrystalline) 12.1
CRC	12.1
CR2	(room temperature) (alpha, amorphous) 4.5

CR2	(room temperature) (alpha, crystalline) 27.2
CR2	(room temperature) (alpha, polycrystalline) 12.1
LNG	12.1
WEL	12.1

58 Ce cerium

use	(room temperature) (gamma, polycrystalline) 6.3
CRC	6.3
CR2	(room temperature) (gamma, amorphous) 6.3
CR2	(room temperature) (gamma, polycrystalline) 6.3
LNG	6.3
WEL	6.3

59 Pr praseodymium

use	(room temperature) (alpha, polycrystalline) 6.7
CRC	6.7
CR2	(room temperature) (alpha, amorphous) 4.5
CR2	(room temperature) (alpha, crystalline) 11.2
CR2	(room temperature) (alpha, polycrystalline) 6.7
LNG	6.7
WEL	6.7

60 Nd neodymium

use	(room temperature) (alpha, polycrystalline) 9.6
CRC	9.6
CR2	(room temperature) (alpha, amorphous) 7.6
CR2	(room temperature) (alpha, crystalline) 13.5
CR2	(room temperature) (alpha, polycrystalline) 9.6
LNG	9.6
WEL	9.6

61 Pm promethium	
use	(room temperature) (alpha, polycrystalline) est. 11
CRC	est. 11
CR2	(room temperature) (alpha, amorphous) est. 9
CR2	(room temperature) (alpha, crystalline) est. 16
CR2	(room temperature) (alpha, polycrystalline) est. 11
LNG	est. 11
WEL	11
62 Sm samarium	
use	(room temperature) (alpha, polycrystalline) 12.7
CRC	12.7
CR2	(room temperature) (alpha, amorphous) 9.6
CR2	(room temperature) (alpha, crystalline) 19.0
CR2	(room temperature) (alpha, polycrystalline) 12.7
LNG	12.7
WEL	12.7
63 Eu europium	
use	(room temperature) (polycrystalline) 35.0
CRC	35.0
CR2	(room temperature) (amorphous) 35.0
CR2	(room temperature) (polycrystalline) 35.0
LNG	35.0
WEL	35
64 Gd gadolinium	
use	(100 °C) (alpha, polycrystalline) 9.4
CRC	(100 °C) 9.4

CR2	(100 °C) (alpha, amorphous) 9.1
CR2	(100 °C) (alpha, crystalline) 10.0
CR2	(100 °C) (alpha, polycrystalline) 9.4
LNG	(100 °C) 9.4
WEL	9.4

65 Tb terbium

use	(room temperature) (alpha, polycrystalline) 10.3
CRC	10.3
CR2	(room temperature) (alpha, amorphous) 9.3
CR2	(room temperature) (alpha, crystalline) 12.4
CR2	(room temperature) (alpha, polycrystalline) 10.3
LNG	10.3
WEL	10.3

66 Dy dysprosium

use	(room temperature) (alpha, polycrystalline) 9.9
CRC	9.9
CR2	(room temperature) (alpha, amorphous) 7.1
CR2	(room temperature) (alpha, crystalline) 15.6
CR2	(room temperature) (alpha, polycrystalline) 9.9
LNG	9.9
WEL	9.9

67 Ho holmium

use	(room temperature) (polycrystalline) 11.2
CRC	11.2
CR2	(room temperature) (amorphous) 7.0
CR2	(room temperature) (crystalline) 19.5
CR2	(room temperature) (polycrystalline) 11.2

LNG	11.2
WEL	11.2

68 Er erbium

use	(room temperature) (polycrystalline) 12.2
CRC	12.2
CR2	(room temperature) (amorphous) 7.9
CR2	(room temperature) (crystalline) 20.9
CR2	(room temperature) (polycrystalline) 12.2
LNG	12.2
WEL	12.2

69 Tm thulium

use	(room temperature) (polycrystalline) 13.3
CRC	13.3
CR2	(room temperature) (amorphous) 8.8
CR2	(room temperature) (crystalline) 22.2
CR2	(room temperature) (polycrystalline) 13.3
LNG	13.3
WEL	13.3

70 Yb ytterbium

use	(room temperature) (beta, polycrystalline) 26.3
CRC	26.3
CR2	(room temperature) (beta, amorphous) 26.3
CR2	(room temperature) (beta, polycrystalline) 26.3
LNG	26.3
WEL	26.3

71 Lu lutetium

use	(room temperature) (polycrystalline) 9.9
CRC	9.9
CR2	(room temperature) (amorphous) 4.8
CR2	(room temperature) (crystalline) 20.0
CR2	(room temperature) (polycrystalline) 9.9
LNG	9.9
WEL	9.9

72 Hf hafnium

use	5.9
CRC	5.9
LNG	5.9
WEL	5.9

73 Ta tantalum

use	6.3
CRC	6.3
LNG	6.3
WEL	6.3

74 W tungsten

use	4.5
CRC	4.5
LNG	4.5
WEL	4.5

75 Re rhenium

use	6.2
CRC	6.2
LNG	6.2

WEL	6.2
	76 Os osmium
use	5.1
CRC	5.1
LNG	5.1
WEL	5.1
	77 Ir iridium
use	6.4
CRC	6.4
LNG	6.4
WEL	6.4
	78 Pt platinum
use	8.8
CRC	8.8
LNG	8.8
WEL	8.8
	79 Au gold
use	14.2
CRC	14.2
LNG	14.2
WEL	14.2
	80 Hg mercury
use	60.4
CRC	60.4
	81 Tl thallium
use	29.9

CRC	29.9
LNG	29.9
WEL	29.9
	82 Pb lead
use	28.9
CRC	28.9
LNG	28.9
WEL	28.9
	83 Bi bismuth
use	13.4
CRC	13.4
LNG	13.4
WEL	13.4
	84 Po polonium
use	23.5
CRC	23.5
	90 Th thorium
use	11.0
CRC	11.0
LNG	11.1
WEL	11.0
	92 U uranium
use	13.9
CRC	13.9
LNG	13.9
WEL	13.9

94 Pu **plutonium**

use	46.7
CRC	46.7
LNG	46.7

体积热容量查询表

由 www.52souji.net 发表于 2012 年 5 月 26 日 || 5,449 浏览

下表是不同元素的热容量数据，其中 Volumetric heat capacity 一列为体积热容量。数据来自维基百科：heat capacity 。

Table of specific heat capacities at 25 °C (298 K) unless otherwise noted Substance	Phase	(mass) specific heat capacity c_p or c_m J·g⁻¹·K⁻¹	Constant pressure molar heat capacity $C_{p,m}$ J·mol⁻¹·K⁻¹	Constant volume molar heat capacity $C_{v,m}$ J·mol⁻¹·K⁻¹	Volumetric heat capacity C_v J·cm⁻³·K⁻¹	Constant vol. atom-molar heat capacity in units of R $C_{v,m(atom)}$ atom-mol⁻¹
Air (Sea level, dry, 0 °C (273.15 K))	gas	1.0035	29.07	20.7643	0.001297	~ 1.25 R
Air (typical room conditions[A])	gas	1.012	29.19	20.85	0.00121	~ 1.25 R
Aluminium	solid	0.897	24.2		2.422	2.91 R
Ammonia	liquid	4.700	**80.08**		3.263	3.21 R
Animal tissue (incl. human)[19]	mixed	3.5			3.7*	
Antimony	solid	0.207	25.2		1.386	3.03 R
Argon	gas	0.5203	20.7862	12.4717		1.50 R
Arsenic	solid	0.328	24.6		1.878	2.96 R
Beryllium	solid	1.82	16.4		3.367	1.97 R

Table of specific heat capacities at 25 °C (298 K) unless otherwise noted Substance	Phase	(mass) specific heat capacity c_p or c_m J·g⁻¹·K⁻¹	Constant pressure molar heat capacity $C_{p,m}$ J·mol⁻¹·K⁻¹	Constant volume molar heat capacity $C_{v,m}$ J·mol⁻¹·K⁻¹	Volumetric heat capacity C_v J·cm⁻³·K⁻¹	Constant vol. atom-molar heat capacity in units of R $C_{v,m(atom)}$ atom-mol⁻¹
Bismuth[20]	solid	0.123	25.7		1.20	3.09 R
Cadmium	solid	0.231	26.02			3.13 R
Carbon dioxideCO_2[16]	gas	0.839*	36.94	28.46		1.14 R
Chromium	solid	0.449	23.35			2.81 R
Copper	solid	0.385	24.47		3.45	2.94 R
Diamond	solid	0.5091	**6.115**		1.782	0.74 R
Ethanol	liquid	2.44	112		1.925	1.50 R
Gasoline(octane)	liquid	2.22	228		1.64	1.05 R
Glass[20]	solid	0.84				
Gold	solid	0.129	25.42		2.492	3.05 R
Granite[20]	solid	0.790			2.17	

Table of specific heat capacities at 25 °C (298 K) unless otherwise noted Substance	Phase	(mass) specific heat capacity c_p or c_m J·g⁻¹·K⁻¹	Constant pressure molar heat capacity $C_{p,m}$ J·mol⁻¹·K⁻¹	Constant volume molar heat capacity $C_{v,m}$ J·mol⁻¹·K⁻¹	Volumetric heat capacity C_v J·cm⁻³·K⁻¹	Constant vol. atom-molar heat capacity in units of R $C_{v,m(atom)}$ atom-mol⁻¹
Graphite	solid	0.710	8.53		1.534	1.03 R
Helium	gas	5.1932	20.7862	12.4717		1.50 R
Hydrogen	gas	**14.30**	28.82			1.23 R
Hydrogen sulfideH_2S[16]	gas	1.015*	34.60			1.05 R
Iron	solid	0.450	25.1[*citation needed*]		3.537	3.02 R
Lead	solid	0.129	26.4		1.44	3.18 R
Lithium	solid	3.58	24.8		1.912	2.98 R
Lithium at 181 °C[21]	liquid	**4.379**	30.33		2.242	3.65 R
Magnesium	solid	1.02	24.9		1.773	2.99 R
Mercury	liquid	0.1395	27.98		1.888	3.36 R

Table of specific heat capacities at 25 °C (298 K) unless otherwise noted Substance	Phase	(mass) specific heat capacity c_p or c_m J·g⁻¹·K⁻¹	Constant pressure molar heat capacity $C_{p,m}$ J·mol⁻¹·K⁻¹	Constant volume molar heat capacity $C_{v,m}$ J·mol⁻¹·K⁻¹	Volumetric heat capacity C_v J·cm⁻³·K⁻¹	Constant vol. atom-molar heat capacity in units of R $C_{v,m(atom)}$ atom-mol⁻¹
Methane at 2 °C	gas	2.191	35.69			0.66 R
Methanol(298 K)[22]	liquid	2.14	68.62			1.38 R
Nitrogen	gas	1.040	29.12	20.8		1.25 R
Neon	gas	1.0301	20.7862	12.4717		1.50 R
Oxygen	gas	0.918	29.38	21.0		1.26 R
Paraffin wax $C_{25}H_{52}$	solid	2.5 (ave)	**900**		2.325	1.41 R
Polyethylene (rotomolding grade)[23]	solid	2.3027				
Polyethylene (rotomolding grade)[23]	liquid	2.9308				
Silica (fused)	solid	0.703	42.2		1.547	1.69 R

Table of specific heat capacities at 25 °C (298 K) unless otherwise noted Substance	Phase	(mass) specific heat capacity c_p or c_m J·g⁻¹·K⁻¹	Constant pressure molar heat capacity $C_{p,m}$ J·mol⁻¹·K⁻¹	Constant volume molar heat capacity $C_{v,m}$ J·mol⁻¹·K⁻¹	Volumetric heat capacity C_v J·cm⁻³·K⁻¹	Constant vol. atom-molar heat capacity in units of R $C_{v,m(atom)}$ atom-mol⁻¹
Silver[20]	solid	0.233	24.9		2.44	2.99 R
Sodium	solid	1.230	28.23			3.39 R
Tin	solid	0.227	27.112			3.26 R
Titanium	solid	0.523	26.060			3.13 R
Tungsten[20]	solid	0.134	24.8		2.58	2.98 R
Uranium	solid	**0.116**	27.7		2.216	3.33 R
Water at 100 °C (steam)	gas	**2.080**	**37.47**	**28.03**		1.12 R
Water at 25 °C	liquid	**4.1813**	**75.327**	**74.53**	**4.1796**	3.02 R
Water at 100 °C	liquid	**4.1813**	**75.327**	**74.53**	**4.2160**	3.02 R
Water at −10 °C (ice)[20]	solid	**2.11**	**38.09**		**1.938**	1.53 R

Table of specific heat capacities at 25 °C (298 K) unless otherwise noted

Substance	Phase	(mass) specific heat capacity c_p or c_m J·g⁻¹·K⁻¹	Constant pressure molar heat capacity $C_{p,m}$ J·mol⁻¹·K⁻¹	Constant volume molar heat capacity $C_{v,m}$ J·mol⁻¹·K⁻¹	Volumetric heat capacity C_v J·cm⁻³·K⁻¹	Constant vol. atom-molar heat capacity in units of R $C_{v,m(atom)}$ atom-mol⁻¹
Zinc[20]	solid	0.387	25.2		2.76	3.03 R

Notable minima and maxima are shown in maroon

所有元素的结合能查询表

由 www.52souji.net 发表于 2012 年 5 月 14 日 || 3,223 浏览

结合能也是计算材料中经常用的，这里提供两种方法。

需要的自己拿走，如果你能够回复一下，让我知道它对你有帮助，我将会更加开心。

结合能在线查询

http://www.knowledgedoor.com/2/elements_handbook/cohesive_energy.html

我也专门截了这个图，方便你保存到本地。

Element	Cohesive Energy (Click ▼ to see citations)	
	Per Mole	Per Atom
Actinium	410 kJ/mol ▼	4.25 eV/atom ▼
Aluminum	327 kJ/mol ▼	3.39 eV/atom ▼
Americium	264 kJ/mol ▼	2.73 eV/atom ▼
Antimony	265 kJ/mol ▼	2.75 eV/atom ▼
Argon	7.74 kJ/mol ▼	0.080 eV/atom ▼
Arsenic	285.3 kJ/mol ▼	2.96 eV/atom ▼
Barium	183 kJ/mol ▼	1.90 eV/atom ▼
Beryllium	320 kJ/mol ▼	3.32 eV/atom ▼
Bismuth	210 kJ/mol ▼	2.18 eV/atom ▼
Boron	561 kJ/mol ▼	5.81 eV/atom ▼
Bromine	118 kJ/mol ▼	1.22 eV/atom ▼
Cadmium	112 kJ/mol ▼	1.16 eV/atom ▼
Calcium	178 kJ/mol ▼	1.84 eV/atom ▼
Carbon	711 kJ/mol ▼	7.37 eV/atom ▼
Cerium	417 kJ/mol ▼	4.32 eV/atom ▼
Cesium	77.6 kJ/mol ▼	0.804 eV/atom ▼
Chlorine	135 kJ/mol ▼	1.40 eV/atom ▼
Chromium	395 kJ/mol ▼	4.10 eV/atom ▼
Cobalt	424 kJ/mol ▼	4.39 eV/atom ▼
Copper	336 kJ/mol ▼	3.49 eV/atom ▼
Curium	385 kJ/mol ▼	3.99 eV/atom ▼
Dysprosium	294 kJ/mol ▼	3.04 eV/atom ▼
Erbium	317 kJ/mol ▼	3.29 eV/atom ▼
Europium	179 kJ/mol ▼	1.86 eV/atom ▼
Fluorine	81.0 kJ/mol ▼	0.84 eV/atom ▼
Gadolinium	400 kJ/mol ▼	4.14 eV/atom ▼
Gallium	271 kJ/mol ▼	2.81 eV/atom ▼
Germanium	372 kJ/mol ▼	3.85 eV/atom ▼
Gold	368 kJ/mol ▼	3.81 eV/atom ▼
Hafnium	621 kJ/mol ▼	6.44 eV/atom ▼
Holmium	302 kJ/mol ▼	3.14 eV/atom ▼
Indium	243 kJ/mol ▼	2.52 eV/atom ▼
Iodine	107 kJ/mol ▼	1.11 eV/atom ▼
Iridium	670 kJ/mol ▼	6.94 eV/atom ▼
Iron	413 kJ/mol ▼	4.28 eV/atom ▼

另外一种格式的表格（点击看大图）

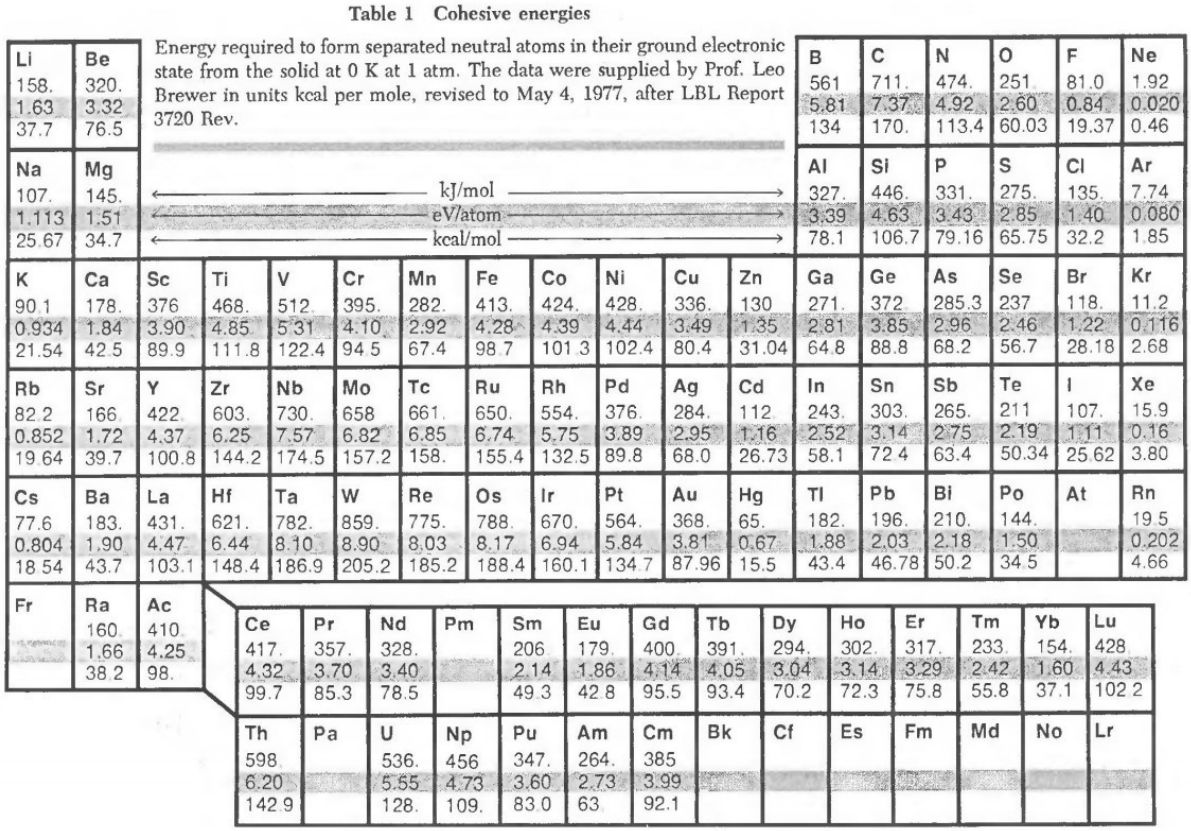


Table 1 Cohesive energies

Energy required to form separated neutral atoms in their ground electronic state from the solid at 0 K at 1 atm. The data were supplied by Prof. Leo Brewer in units kcal per mole, revised to May 4, 1977, after LBL Report 3720 Rev.

kJ/mol
eV/atom
kcal/mol

Li	Be											B	C	N	O	F	Ne
158.	320.											561	711.	474.	251	81.0	1.92
1.63	3.32											5.81	7.37	4.92	2.60	0.84	0.020
37.7	76.5											134	170.	113.4	60.03	19.37	0.46
Na	Mg											Al	Si	P	S	Cl	Ar
107.	145.											327.	446.	331.	275.	135.	7.74
1.113	1.51											3.39	4.63	3.43	2.85	1.40	0.080
25.67	34.7											78.1	106.7	79.16	65.75	32.2	1.85
K	Ca	Sc	Ti	V	Cr	Mn	Fe	Co	Ni	Cu	Zn	Ga	Ge	As	Se	Br	Kr
90.1	178.	376	468.	512.	395.	282.	413.	424.	428.	336.	130	271.	372.	285.3	237	118.	11.2
0.934	1.84	3.90	4.85	5.31	4.10	2.92	4.28	4.39	4.44	3.49	1.35	2.81	3.85	2.96	2.46	1.22	0.116
21.54	42.5	89.9	111.8	122.4	94.5	67.4	98.7	101.3	102.4	80.4	31.04	64.8	88.8	68.2	56.7	28.18	2.68
Rb	Sr	Y	Zr	Nb	Mo	Tc	Ru	Rh	Pd	Ag	Cd	In	Sn	Sb	Te	I	Xe
82.2	166	422.	603.	730.	658	661.	650.	554.	376.	284.	112	243.	303.	265.	211	107.	15.9
0.852	1.72	4.37	6.25	7.57	6.82	6.85	6.74	5.75	3.89	2.95	1.16	2.52	3.14	2.75	2.19	1.11	0.16
19.64	39.7	100.8	144.2	174.5	157.2	158.	155.4	132.5	89.8	68.0	26.73	58.1	72.4	63.4	50.34	25.62	3.80
Cs	Ba	La	Hf	Ta	W	Re	Os	Ir	Pt	Au	Hg	Tl	Pb	Bi	Po	At	Rn
77.6	183.	431.	621.	782.	859.	775.	788.	670.	564.	368.	65.	182.	196.	210.	144.		19.5
0.804	1.90	4.47	6.44	8.10	8.90	8.03	8.17	6.94	5.84	3.81	0.67	1.88	2.03	2.18	1.50		0.202
18.54	43.7	103.1	148.4	186.9	205.2	185.2	188.4	160.1	134.7	87.96	15.5	43.4	46.78	50.2	34.5		4.66
Fr	Ra	Ac															
	160.	410.															
	1.66	4.25															
	38.2	98.															

Ce	Pr	Nd	Pm	Sm	Eu	Gd	Tb	Dy	Ho	Er	Tm	Yb	Lu
417.	357.	328.		206	179.	400	391.	294.	302.	317.	233.	154.	428
4.32	3.70	3.40		2.14	1.86	4.14	4.05	3.04	3.14	3.29	2.42	1.60	4.43
99.7	85.3	78.5		49.3	42.8	95.5	93.4	70.2	72.3	75.8	55.8	37.1	102.2
Th	Pa	U	Np	Pu	Am	Cm	Bk	Cf	Es	Fm	Md	No	Lr
598		536.	456	347.	264.	385							
6.20		5.55	4.73	3.60	2.73	3.99							
142.9		128.	109.	83.0	63	92.1							

所有元素的晶格常数查询表

由 www.52souji.net 发表于 2012 年 5 月 14 日 || 11,462 浏览

晶格常数是计算中比较常用的参数，有时需要用到时，却发现找不到。我这里搜集了几个功能比较强大的网站提供给大家，希望能够方便大家查找。实际上，他们提供的信息远远不限于晶格常数。

WebElements: http://www.webelements.com/copper/crystal_structure.html
PeriodicTable: http://www.periodictable.com/Properties/A/LatticeConstants.html

另外，我从网上找到了一个晶格常数的表，可以直接查询，也一并提供给大家（点击放大）

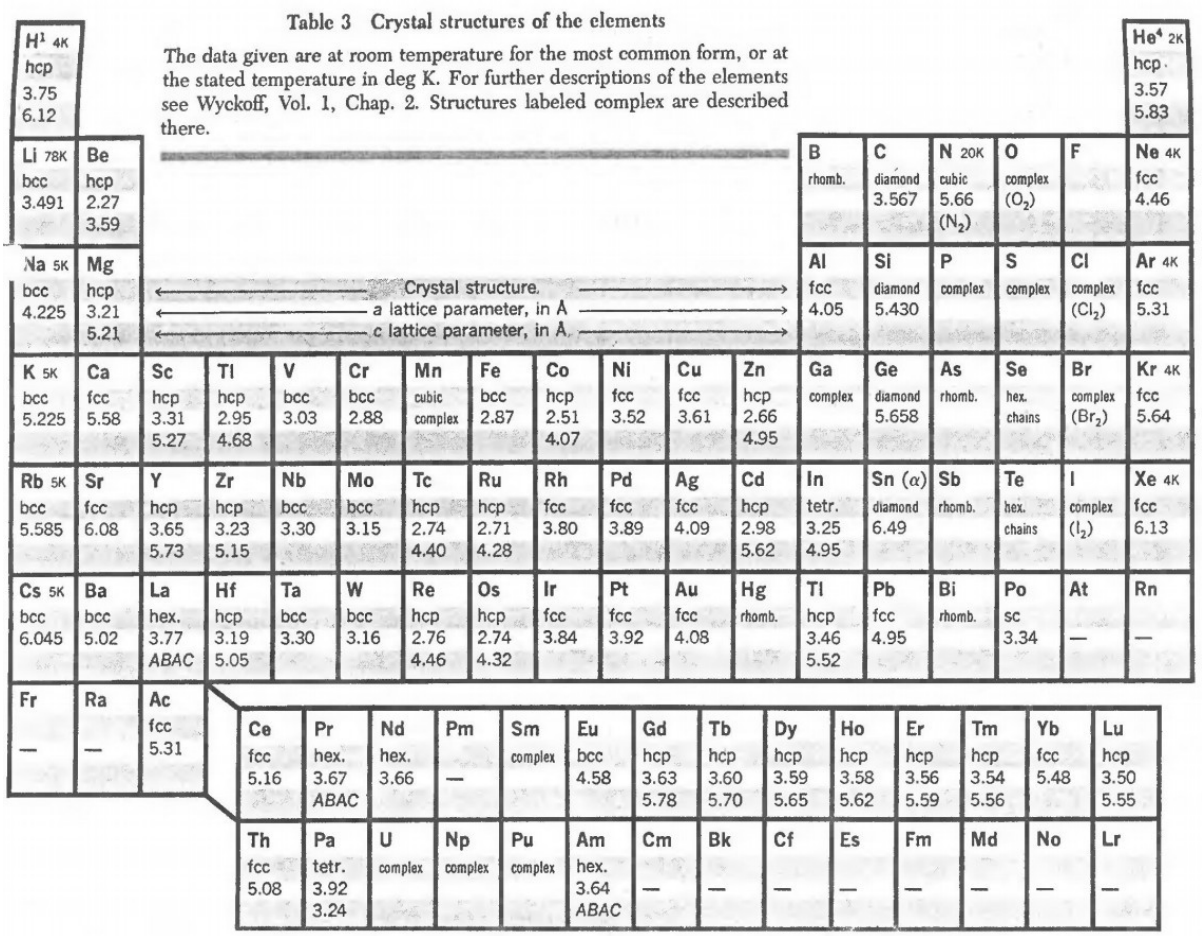


Table 3 Crystal structures of the elements

The data given are at room temperature for the most common form, or at the stated temperature in deg K. For further descriptions of the elements see Wyckoff, Vol. 1, Chap. 2. Structures labeled complex are described there.

Crystal structure.
a lattice parameter, in A
c lattice parameter, in A

H^1 4K hcp 3.75 6.12																	He^4 2K hcp 3.57 5.83
Li 78K bcc 3.491	Be hcp 2.27 3.59											B rhomb.	C diamond 3.567	N 20K cubic 5.66 (N_2)	O complex (O_2)	F	Ne 4K fcc 4.46
Na 5K bcc 4.225	Mg hcp 3.21 5.21											Al fcc 4.05	Si diamond 5.430	P complex	S complex	Cl complex (Cl_2)	Ar 4K fcc 5.31
K 5K bcc 5.225	Ca fcc 5.58	Sc hcp 3.31 5.27	Ti hcp 2.95 4.68	V bcc 3.03	Cr bcc 2.88	Mn cubic complex	Fe bcc 2.87	Co hcp 2.51 4.07	Ni fcc 3.52	Cu fcc 3.61	Zn hcp 2.66 4.95	Ga complex	Ge diamond 5.658	As rhomb.	Se hex. chains	Br complex (Br_2)	Kr 4K fcc 5.64
Rb 5K bcc 5.585	Sr fcc 6.08	Y hcp 3.65 5.73	Zr hcp 3.23 5.15	Nb bcc 3.30	Mo bcc 3.15	Tc hcp 2.74 4.40	Ru hcp 2.71 4.28	Rh fcc 3.80	Pd fcc 3.89	Ag fcc 4.09	Cd hcp 2.98 5.62	In tetr. 3.25 4.95	Sn (α) diamond 6.49	Sb rhomb.	Te hex. chains	I complex (I_2)	Xe 4K fcc 6.13
Cs 5K bcc 6.045	Ba bcc 5.02	La hex. 3.77 *ABAC*	Hf hcp 3.19 5.05	Ta bcc 3.30	W bcc 3.16	Re hcp 2.76 4.46	Os hcp 2.74 4.32	Ir fcc 3.84	Pt fcc 3.92	Au fcc 4.08	Hg rhomb.	Tl hcp 3.46 5.52	Pb fcc 4.95	Bi rhomb.	Po sc 3.34	At —	Rn —
Fr —	Ra —	Ac fcc 5.31															

Ce fcc 5.16	Pr hex. 3.67 *ABAC*	Nd hex. 3.66	Pm —	Sm complex	Eu bcc 4.58	Gd hcp 3.63 5.78	Tb hcp 3.60 5.70	Dy hcp 3.59 5.65	Ho hcp 3.58 5.62	Er hcp 3.56 5.59	Tm hcp 3.54 5.56	Yb fcc 5.48	Lu hcp 3.50 5.55
Th fcc 5.08	Pa tetr. 3.92 3.24	U complex	Np complex	Pu complex	Am hex. 3.64 *ABAC*	Cm —	Bk —	Cf —	Es —	Fm —	Md —	No —	Lr —

所有元素体积模量查询表

由 www.52souji.net 发表于 2012 年 8 月 25 日 || 1,591 浏览

体积模量，bulk modulus

The bulk modulus (K) of a substance measures the substance's resistance to uniform compression. It is defined as the pressure increase needed to decrease the volume by a factor of 1/e. (From wikipedia: Bulk Modulus)

下表是所有元素的体积模量：

Hydrogen	N/A	Niobium	170 GPa	Thallium	43 GPa
Helium	N/A	Molybdenum	230 GPa	Lead	46 GPa
Lithium	11 GPa	Technetium	N/A	Bismuth	31 GPa

Beryllium	130 GPa	Ruthenium	220 GPa	Polonium	N/A
Boron	320 GPa	Rhodium	380 GPa	Astatine	N/A
Carbon	33 GPa	Palladium	180 GPa	Radon	N/A
Nitrogen	N/A	Silver	100 GPa	Francium	N/A
Oxygen	N/A	Cadmium	42 GPa	Radium	N/A
Fluorine	N/A	Indium	N/A	Actinium	N/A
Neon	N/A	Tin	58 GPa	Thorium	54 GPa
Sodium	6.3 GPa	Antimony	42 GPa	Protactinium	N/A
Magnesium	45 GPa	Tellurium	65 GPa	Uranium	100 GPa
Aluminum	76 GPa	Iodine	7.7 GPa	Neptunium	N/A
Silicon	100 GPa	Xenon	N/A	Plutonium	N/A
Phosphorus	11 GPa	Cesium	1.6 GPa	Americium	N/A
Sulfur	7.7 GPa	Barium	9.6 GPa	Curium	N/A
Chlorine	1.1 GPa	Lanthanum	28 GPa	Berkelium	N/A
Argon	N/A	Cerium	22 GPa	Californium	N/A
Potassium	3.1 GPa	Praseodymium	29 GPa	Einsteinium	N/A
Calcium	17 GPa	Neodymium	32 GPa	Fermium	N/A
Scandium	57 GPa	Promethium	33 GPa	Mendelevium	N/A
Titanium	110 GPa	Samarium	38 GPa	Nobelium	N/A
Vanadium	160 GPa	Europium	8.3 GPa	Lawrencium	N/A
Chromium	160 GPa	Gadolinium	38 GPa	Rutherfordium	N/A

Manganese	120 GPa	Terbium	38.7 GPa	Dubnium	N/A
Iron	170 GPa	Dysprosium	41 GPa	Seaborgium	N/A
Cobalt	180 GPa	Holmium	40 GPa	Bohrium	N/A
Nickel	180 GPa	Erbium	44 GPa	Hassium	N/A
Copper	140 GPa	Thulium	45 GPa	Meitnerium	N/A
Zinc	70 GPa	Ytterbium	31 GPa	Darmstadtium	N/A
Gallium	N/A	Lutetium	48 GPa	Roentgenium	N/A
Germanium	N/A	Hafnium	110 GPa	Ununbium	N/A
Arsenic	22 GPa	Tantalum	200 GPa	Ununtrium	N/A
Selenium	8.3 GPa	Tungsten	310 GPa	Ununquadium	N/A
Bromine	1.9 GPa	Rhenium	370 GPa	Ununpentium	N/A
Krypton	N/A	Osmium	N/A	Ununhexium	N/A
Rubidium	2.5 GPa	Iridium	320 GPa	Ununseptium	N/A
Strontium	N/A	Platinum	230 GPa	Ununoctium	N/A
Yttrium	41 GPa	Gold	220 GPa		
Zirconium	N/A	Mercury	25 GPa		

来源：http://periodictable.com/Properties/A/BulkModulus.an.html

LAMMPS 常用原子间势函数下载

本页面提供 LAMMPS 下常用体系的原子间势函数的下载。持续更新。

对于各种不同势的介绍可以先参考 LAMMPS 文档。右键另存为下载。

内容目录 [隐藏]

一. 纯相的势函数

- Ag：eam/alloy | eam
- Al：eam/fs-1 | eam/fs-2 | eam | eam/alloy
- Au：eam/alloy | eam
- Co：eam/alloy
- Cu：eam/fs-1 | eam/fs-2 | eam/fs-3 | eam/alloy-1 | eam/alloy-mishin1 | eam/alloy-zhou | eam-smf7 | eam-u3 | eam-u6
- Fe：eam/fs-1 | eam/fs-2 | eam/fs-3 | eam/fs-mm
- Mg：eam/fs | eam/fs-mm
- Mo：eam/fs
- Nb：eam/alloy
- Ni：eam/fs-1 | eam/fs-2 | eam/alloy | eam-smf7 | eam-u3
- Pd：eam-u3
- Pt：eam-u3
- Ru：eam/fs
- Si：sw | tersoff
- Ta：eam/alloy
- Ti：eam/fs
- U：eam/fs
- W：eam/fs | eam/alloy
- Zr：eam/fs-1 | eam/fs-2 | eam/fs-3 | eam/fs-mm

二. 二元合金相的势函数

- Ag-Cu：eam/alloy-1 | eam/alloy-2
- Al-Co：eam/alloy
- Al-Cu：eam/alloy-1 | eam/alloy-2
- Al-Fe：eam/fs-1 | eam/fs-2
- Al-H：eam/alloy
- Al-Mg：eam/fs | eam/alloy
- Al-Ni：eam/alloy-1 | eam/alloy-2 | eam/alloy-3 | eam/alloy | eam/fs-mishin
- Al-Pb：eam/alloy

- Al-Ti：eam/alloy
- C-Fe：eam/fs
- C-H：airebo
- C-Si：tersoff | tersoff-zbl | meam
- Cd-Te：sw
- Cu-Fe：eam/alloy
- Cu-Ni：eam/alloy
- Cu-Pd：eam/alloy
- Cu-Zr：eam/fs-1 | eam/fs-2 | eam/fs-3
- Fe-Ni：eam/alloy
- Fe-P：eam/fs | eam/fs
- Fe-V：eam/fs | eam/fs-mm
- Ga-N：sw | tersoff
- H-Ni：eam/alloy
- Ni-Zr：eam/fs
- Pd-H：eam/alloy

三. 三元合金相的势函数

- Al-Mn-Pd：eam/alloy (可能有问题)
- Al-Ni-H：eam/alloy-1 | eam/alloy-2 | eam/fs-1
- C-Ge-Si：tersoff
- Cu-Fe-Ni：eam/alloy

四. 几点说明

1. 调换元素顺序，不影响势函数的选用。比如 Ag-Cu 与 Cu-Ag 势函数相同。
2. 一般而言，高元合金的势函数适用于低元合金或纯相，只需在设置 pair_coeff 时选择相应元素即可。比如 Cu-Fe-Ni 的势函数适用于 Cu-Fe 或 Fe-Ni 等。
3. 因为来源问题，可能有些势函数原本是一样的，请自行鉴别。

五. 势函数来源

1. Interatomic Potentials Repository Project
2. 链接: http://redmine.scorec.rpi.edu/anonsvn/lammps-cuda/potentials/
3. 链接：https://cmse.postech.ac.kr/html/research/2nnmeam.htm

【软件推荐】结构模型可视化软件 VESTA

由 www.52souji.net 发表于 2012 年 9 月 15 日 || 5,376 浏览

内容目录 [隐藏]

1. VESTA 基本介绍

VESTA 是 Visualization for Electronic and STructural Analysis 的简称，直接翻译过来就是电子和结果分析的可视化。

VESTA 是免费软件，提供各种操作系统下的版本[windows, linux, mac]。

VESTA 官网：http://jp-minerals.org/vesta/en/

[不知道为什么这个网站竟然需要翻墙才能访问！]

2. VESTA 软件功能

这里我就不翻译了，用处不大。

- Deal with multiple structural models, volumetric data, and crystal morphologies in the same window.
- Support multiple tabs corresponding to files.
- Support multiple windows with more than two tabs in the same process.
- Deal with virtually unlimited number of objects such as atoms, bonds polyhedra, and polygons on isosurfaces (theoretical limit on 32bit operating system is 1,073,741,823)

- Support lattice transformation from conventional to non-conventional lattice by using matrix. The transformation matrix is also used to create superlattice and sublattice.
- Visualize interatomic distances and bond angles that are restrained in Rietveld analysis with RIETAN-FP.
- Transparent isosurfaces can be overlap with structural models.
- Isosurface can be colored on the basis of another physical quantity.
- Arithmetic operations among multiple volumetric data files.
- High quality smooth rendering of isosurfaces and sections.
- Export high-resolution graphic images exceeding Video card limitation.

3. VESTA 优势

之所以推荐这款软件，是因为在某些方面，它具有无可比拟的优势。

首先，这是一款免费软件，可以在各个操作系统下运行。

其次，VESTA 可以可视化原子数目比较多的体系，几万个原子体系是很顺畅的。

另外，它可以直接可视化 VASP 里面的一些文件，比如 POSCAR，电荷密度等，这绝对是 VASP 使用者的福音。

还有，它提供的测量功能也是非常好用的，比如键长、键角等。其他的功能，比如构建超晶胞等，也是非常不错的。

还有更多功能，等待你自己去挖掘。

4. VESTA 使用方法

VESTA 提供了一个很详细的文档，具体介绍如何使用，但是在我看来完全没有必要，因为软件是傻瓜式的，很容易上手。这里也不多做介绍。

5. VESTA 下载

官方下载地址：http://jp-minerals.org/vesta/en/download.html

我这里也提供备份的下载：

- windows 32 位 | 64 位
- linux 32 位 | 64 位

可视化软件 atomeye 简介、使用方法及常用快捷键

由 www.52souji.net 发表于 2012 年 5 月 3 日 || 9,788 浏览

atomeye 是一款小巧的、功能强大的可视化软件，由 李巨 开发（据说是其博士期间苦于没有很好的可视化软件而开发的）。

内容目录 [隐藏]

1. atomeye 的优势

- 可以轻松可视化原子数在百万级的体系（目前，我还没有发现其他的可视化软件能够做到如此）；
- 程序小巧，仅仅 4M；
- 无需编译，直接运行（这个可能未必算优点）；
- 可以远程在终端下打开；
- 功能强大，渲染效果很不错；
- 免费。

2. atomeye 支持的文件格式

atomeye 目前只支持两种文件格式：CFG 格式（包括标准 CFG 和扩展 CFG 格式）和 PDB 格式。

对 CFG 格式的支持更加完美，所以建议使用 CFG 格式。

3. atomeye 下载

可以前往右边页面下载：http://li.mit.edu/Archive/Graphics/A/#download

- i686 Linux
- Alpha Linux GLIBC2.1
- Sgi Irix
- Sgi Irix64
- Sun Solaris
- HP UX
- Windows with Cygwin / X (README.txt)

- Alpha Tru64 UNIX
- Mac OS X (v10.4 and before, v10.5 “Leopard”) with Darwin (see A, B, C for button issues)

4. atomeye 使用

下载到的 atomeye 文件名为 A.i686，我一般会把文件名改为 atomeye，更直观：然后赋予 atomeye 可执行属性。

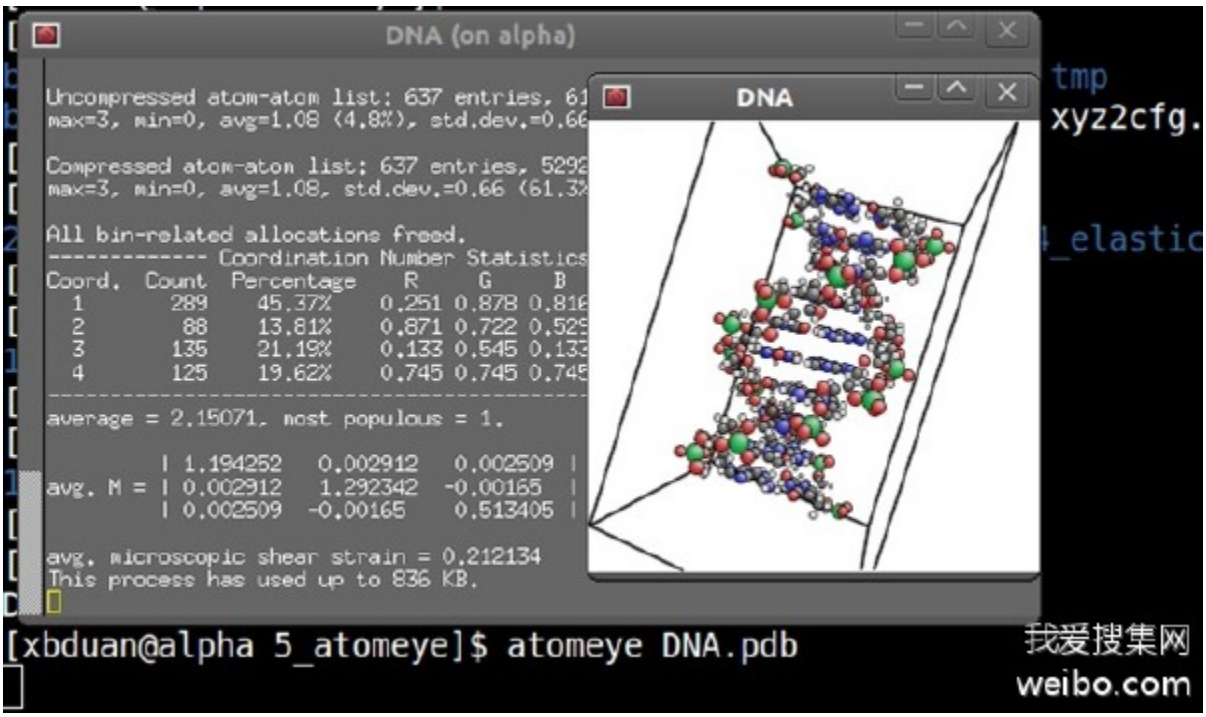
```
$ mv A.i686 atomeye

$ chmod +x atomeye
```

另外，建议将 atomeye 添加到系统路径，这样就可以在任意目录下均可以直接使用 atomeye 命令调用。

假如你当前的系统是 linux 系统（而不单单是一个 linux 终端），你的 atomeye 装在远程的主机上，那么你只需要使用 -Y 参数连接 ssh，那么你可以实现在终端上运行 atomeye。

```
$ ssh -Y servername

$ atomeye DNA.cfg
```

atomeye 会打开如下图所示：



5. atomeye 常用命令

atomeye 的功能很强大，这里只列举部分常用功能。

快捷键	功能描述
F10	重新加载当前文件
B	切换是否显示键
D	改变背景颜色
F	根据原子 ID 查找并显示原子信息
I	调整模拟盒子边框显示模式
J	jpg 格式截图
K	根据原子坐标关系对原子上色
L	切换到正常的原子颜色

O	切换到最初的原子颜色（对个别原子改变颜色后）
P	png 格式截图
Q	退出窗口
U	将视图放正
Y	自动连续产生图片
,	显示过去单击的两个原子间的距离
TAB	一般视图与透视图切换
Insert/Delete	显示下一个文件
Home/End	增加/减少键半径
Pageup/PageDown	增加/减少原子半径
Right/Left/Up/Down	绕轴旋转
Ctrl+Insert/Delete	显示第一个/最后一个文件
Ctrl+Left-click	改变选中一个原子/键的颜色/半径
Ctrl+Shift+Left-click	改变选中一类原子/键的颜色/半径
Ctrl+Right/Left/Up/Down	平面内移动视图
Ctrl+Shift+Up/Down	放大/缩小视图
Shift+Up/Down	平面内顺时针/逆时针旋转视图
Shift+Left/Right	向前/向后推进截平面
Shift+[0-9,a-f]	创建截平面

Shift+CapsLockOn+[0-9,a-f]	删除截平面
CapsLockon+Right/Left/Up/Down	在周期性边界条件下移动对象

更多关于 atomeye 的信息，可以访问：http://li.mit.edu/Archive/Graphics/A/

建模：MS 为 lammps 建模(msi2lmp 转换)

MS 为 lammps 建模： 不得不承认，lammps 以及很多其他开源的模拟软件在易用性上做得远不如商业模拟软件。其中最为重要的部分就是 model builder 和 visualizer。即使可以联用一些辅助软件，例如 VMD，由于软件间的接口设计或者模拟软件输出格式等问题，也是极为不方便（相比用过 VMD 显示 lammps 输出的朋友都有体会吧）。这只是牢骚话，下面转入主题。

MS 的 builder 没得说，lammps 也有附加工具“msi2lmp”支持把 MS 支持的文件格式转换为 lammps 可以使用的格式。其实名字叫 msi2lmp，事实上你不能使用*.msi 文件最为输入，而应该使用 car/mdf 组合。*.car 文件记录了原子坐标，mdf 文件记录了键接方式。使用 MS 为 lammps 建模的工作流程是：

1. MS 中建模

2. Typing，也就是为每个原子分配原子类型。

这是最重要的一步。因为 msi2lmp 工具只支持 cvff 和 cff91，所以在 MS 中一定要为原子预先分配好这两个力场所支持的原子类型！具体做法是，在 discover 模块中打开 setup 对话框，energy 选项卡中选中 cvff 或者 cff91，然后去 typing 选项卡，type 即可。如果你不希望力场自动分配电荷，可以取消之。

对于某些力场不支持的原子类型（有的时候支持的也会分析错，哈哈），typying 过程中会报错，你可以在选中那些原子，手工分配原子类型。

3. 输出。只要选择 car 格式 export 就好了。

4. 转换。把 msi2lmp.exe 和 cvff.frc 或 cff91.frc 复制到你存放 car/mdf 文件的目录中，执行 msi2lmp ×××就可以得到 lammps 可以使用的数据文件了（×××就是你的×××.car 文件没有后缀的名称）。msi2lmp 默认使用 cvff，要使用 cff91 可以看 msi2lmp 的帮助信息。

最后说明一下，msi2lmp 对有些模型会报告不能为某些键角或者两面角找到相关的力场参数，这不一定表示有什么错误。例如 sp2 杂化的中心原子和其他 3 个原子所形成的平面结构中，不是所有的键角和 improper 都需要显示表达的。如果遇到这些警告，你会发现生成的 lammps 数据文件中有些力场参数是 0.000。你要注意检查并确认这些项的确是无关紧要的。最后你运行你的模拟时，你还有可能遇到“Incorrect sign arg for dihedral coefficients”错误，这也和 msi2lmp 为上述零参数相互作用项生成的数据有关。你只要在 lammps 数据文件中把那些零参数项的符号参数改为 1 即可，反正能量参数都是 0，符号项是什么又有什么关系呢。

Material Studio 操作

1. 一般性说明

Materilas Studio 的精华不是 Discover 板块，而是其主打 CASTEP 模块和 DMOL3 模块。

2. 建模

MS 能够修正由于误差造成的 3D 结构，但是绝对不会纠正严重的结构错误。看结构画得好不好，可以通过几何优化过程中能量演化曲线的下降速率看出，正确的结构能量曲线会迅速下降的。这是一个很容易掌握的经验。

ICSD 的一点说明

搞无机晶体的虫虫一定要有完全版的 ICSD。我用的 ICSD 是在下面地址下载的：

ftp://upload:mse@166.111.38.38

以上面为例：

Space Group F d -3 m S 是空间群。在 MS 中按这样输入是可行的，应为这种编号在 MS 中是统一的；

Unit Cell 8.45 8.45 8.45 90. 90. 90.是你要在 BUILD Crystal 中的 Lattice 第二页要填写的对应晶胞大小；

上面两个填好了，晶胞格子就 OK 了。

Atom # OX SITE x y z SOF H

Ni 1 +2 8 a 0 0 0 1. 0

Mn 1 +3 16 d 0.625 0.625 0.625 1. 0

O 1 -2 32 e 0.385 0.385 0.385 1. 0

以上 Atom 数据在 MS 中用在 Add Atom 操作中为建好的格子里面加原子的，这些参数依次填完后，你的污迹晶体就初步建好了。

Vol 603.35 是晶胞体积，可以选中格子，在 properties 浏览窗中看到的。

最后别忘了选力场 Typing 一下，不然所有的电荷都是 0。

未掺杂体系---掺杂

问：现在想计算一个掺杂体系，知道了未掺杂体系的结构（晶胞参数，空间群，原子坐标），现在想把其中的一种元素进行部分取代，进行计算，不知道哪位高人能指点一下，在建立晶胞模型的时候怎么操作?

答：用鼠标点上将要被取代得原子（点上后原子颜色将变成黄色），在窗口的右边属性栏中，将会显示这个原子的相关属性，并告诉你这个原子的元素种类（比方是 Al 吧），然后点这个元素种类 Al，将出现一个元素周期表，选择你要掺杂得原子，确定就可以了！

我是在 MS-CASTEP 模块下做模拟计算的,这个问题最后是这样解决的:建立完没有掺杂的晶胞后,选 supercell,然后再选择要替换的原子,进行掺杂.如果不把晶胞的对称性改为 supercell 就不能改变其中的一个原子,而是由于对称性把同一元素的所以原子改变了。（应该是把晶胞的对称性改成 P1 吧，supercell 不是晶胞的对称性。）

在 MS 下，在反应中加溶液，怎么画呢？加个盒子吗？

你提的问题要分开来看。如果考察反应，应该用 Dmol3 做反应历程探寻，如果是为了考察溶液的物理性质，可以用 amorphous cell。后者要加一个盒子，那个盒子叫周期性边界条件。由于你的问题描述的太笼统，且 MS 不是虚拟化学反应实验模拟，所以我只能给你讲述到这里。MS 的主要功用还是服务于计算化学的。（或改用 Gaussian 软件）

关于画苯环，查看电荷

很多初学者经常会把环已三烯和苯环相混淆。表面上看，凯库勒式是一样的。实际上在 MS 中二者是截然不同的。画苯环最简单的方法是：Alt+六元环画笔。第二种方法就是先画一个六元环，再按住 Shift 键选中六个环单键，将其改为 partial double bond。

当我们考察各种种画法是否相同，可以考察电荷。刚画好的环已三烯和苯环所有原子的电荷都为 0（可以从 properties 浏览器中看到）。这是因为画 3D 模型只是第一步，我们并没有用力场定义它们。很多朋友往往认为画好就行了，马上就可以用模块算，这种想法是导致许多严重错误的根源。我向大家建议一个好的习惯：画好模型后，用 Discover--〉Setup--〉Energy 页中选中 compass 力场（最常用的定义最全力场），再在 Typing 页中按下 calculate 键，马上分子里的所有原子都具有了电荷。

细心的朋友可能发现菜单 Modify 下面有一个 charge 项，里面也有一个 calculate 按钮。这个按钮和 discover-->setup-->typing 中的 calculate 有着本质的不同。请大家不要随便按。好奇的可以看看相关 help。

3. 基础优化(结构优化)

规则化按钮，

minimizer 之外，还有更好的单分子和周期晶胞的优化方法；

Forsite Tools 优化，特别是 Geometry Optimization 方法：

（比 Minimizer 好多了，特别适合新手。（注意力场的选择）。可以看看 Forcite 的帮助文件，篇幅很短，而且很简单！）；

什么画好了还要计算一下电荷？直接进行构型优化之类的不可以吗?

完全可以！这取决于你在 Discover 中 SETUP 的参数设置。很多动力学相关模块都仰仗 Discover 中 SETUP 的参数设置，然后自动就可以 TYPING！我这里只是告诉大家一个纠错的方法。很多时候我们犯的错误都来源于一些很难想到的小错误。

小补贴：使用 fragment 进行 build polymer 的时候一定要先将自己定义的 frament 优化一下，再 build polymer。否则一旦用没优化的 fragment 建了长链，能量有可能高的离谱，那时你再做优化都没用。用这样的模型即使运算成功了，那本来就是个错误！这就是所谓的软错误,软错误计算机是不报警的！

4. 力场

力场 COMPASS, PCFF, CVFF, DREIDING

COMPASS 力场：

我们用得最多的还是 COMPASS 力场。其他的力场用在纯有机物研究，或生命药物研究方面。力场主要强调对元素周期表的覆盖度和各价态元素的定义准确性。这两个任务属 COMPASS 力场结合的最好（最广的覆盖度和较为准确的价态元素定义）。其它的力场在有机分子或 DNA 方面有着很高的准确性，但是元素覆盖较少。这里我就不具体介绍其他的力场了。大家如果感兴趣可以看看徐攸杰的计算机药物分子模拟一书，那里面讲的很全面。

另外 universal 力场是 COMPASS 的候选。大家在使用的时候一定要到 DISCOVER 里看看有没有对所研究的分子各形式的定义，否则会报错。

5. 加电场

materials studio 软件， 如何在碳纳米管两端加电场？

答：build- charges-calculate 就可以啦，或者如果用 discover 做 md 时间足够长电荷自动就加上去了。（答案不理想）

6. 周期边界条件

中如何进行设置，使 XY 方向上建立周期边界条件，而在 Z 方向上自由?

建包，加真空层

1.Build->Surfaces->Cleave surface，切割晶体，得到一个二维（u,v）的平面——只有二维周期性

2.Build->Crystals->Build Vacuum Slab，在表面上方建立一个真空层。

如何得到三维的,真空层呢?

LAMMPS 翻译系列】atom_style 命令

由 www.52souji.net 发表于 2013 年 10 月 2 日 || 2,305 浏览

atom_style 命令用来定义模拟过程中原子的类型，它会决定原子包括哪些属性。

内容目录 [隐藏]

使用语法

```
atom_style style args
```

其中：

- style 可取：

  angle or *atomic* or *body* or *bond* or *charge* or *dipole* or *electron* or *ellipsoid* or *full* or *line* or *meso* or *molecular* or *peri* or *sphere* or *tri* or *hybrid*
- 只有 body 和 hybrid 需要设置 args（具体参考原文档）

使用举例

```
atom_style atomic

atom_style bond

atom_style full

atom_style body nparticle 2 10

atom_style hybrid charge bond

atom_style hybrid charge body nparticle 2 5
```

使用介绍

定义模拟过程中原子的类型，它会决定原子包括哪些属性。该命令必须在建立模拟盒子（使用命令 read_data 或 read_restart 或 create_box）之前使用。

一定原子的类型被设置了，在模拟过程中就不能在改变了，因此尽量更加通用的类型，以免某些属性没有被包括却被用到了。举例来说，对于 bond 类型，原子没有

角度项，也不能在模拟的过程中为模型添加角度属性。尽量使用更加通用的类型，虽然这可能会因为某些属性没有用到，同时会稍微降低效率，但总比出错好。

类型的选择会影响每个原子所能够存储的性质、在计算力的时候处理器间能够交换的性质以及在数据文件中所能够显示并被命令 read_data 读入的性质。
下表中列出了每种类型所包括的属性以及会用到这种类型的典型物理体系。所有的类型都包括坐标、速度、原子 ID 和原子类型。参考命令 read_data、create_atoms 和 set，了解关于如何设置这些性质。

angle	bonds and angles	bead-spring polymers with stiffness
atomic	only the default values	coarse-grain liquids, solids, metals
body	mass, inertia moments, quaternion, angular momentum	arbitrary bodies
bond	bonds	bead-spring polymers
charge	charge	atomic system with charges
dipole	charge and dipole moment	system with dipolar particles
electron	charge and spin and eradius	electronic force field
ellipsoid	shape, quaternion, angular momentum	aspherical particles
full	molecular + charge	bio-molecules
line	end points, angular velocity	rigid bodies
meso	rho, e, cv	SPH particles
molecular	bonds, angles, dihedrals, impropers	uncharged molecules
peri	mass, volume	mesocopic Peridynamic models

sphere	diameter, mass, angular velocity	granular models
tri	corner points, angular momentum	rigid bodies
wavepacket	charge, spin, eradius, etag, cs_re, cs_im	AWPMD

注意：通过命令 fix property/atom 对原子添加它原先没有的属性是可以的，比如分子 ID。该命令也允许为原子添加新的由整数或浮点数组成的自定义属性。参考命令 fix property/atom，了解更多细节和例子。
上面的类型中，除了 *sphere*, *ellipsoid*, *electron*, *peri*, *wavepacket*, *line*, *tri*, and *body* 定义的是有限尺寸粒子，其他的类型定义的都是点粒子。

对于定义为有限尺寸粒子的类型，在设置质量时是对每一个粒子进行的；而对于点粒子，这是对类型进行设置的。

- sphere：粒子是球形的，每个粒子存有其直径和质量。如果其直径大于 0，粒子是有限尺寸的球；如果直径等于 0，它是一个点粒子（point particle）。
- ellipsoid：粒子是椭球体，每个粒子存有一个标记，用来区分该粒子是有限尺寸的椭球还是点粒子。如果是椭球，那么每个粒子就存有形状矢量，这个矢量包括椭球体的三个直径和一个代表其方位的四维矢量。
- electron：the particles representing electrons are 3d Gaussians with a specified position and bandwidth or uncertainty in position, which is represented by the eradius = electron size.
- peri：粒子是球形的，每个粒子存有其质量和体积。
- meso：粒子用于光滑粒子流体动力学 SPH 计算，存有密度、能量和热容。
- wavepacket：与类型 electron 类似，but the electrons may consist of several Gaussian wave packets, summed up with coefficients cs= (cs_re,cs_im). Each of the wave packets is treated as a separate particle in LAMMPS, wave packets belonging to the same electron must have identical *etag* values.
- line：the particles are idealized line segments and each stores a per-particle mass and length and orientation (i.e. the end points of the line segment).
- tri：the particles are planar triangles and each stores a per-particle mass and size and orientation (i.e. the corner points of the triangle).
- body：the particles are arbitrary bodies with internal attributes defined by the “style” of the bodies, which is specified by the *bstyle* argument. Body particles can represent complex entities, such as surface meshes of discrete points, collections of sub-particles, deformable objects, etc.

一般来说，模拟时需要单独一种的原子类型（non-hybrid）。但如果模拟中的有些原子不需要某些性质，但其他的原子需要，那么就要使用包含所有这些性质的类型。比如，如果有些原子需要电荷，而另外一些不需要，要使用类型 charge；如果有些原子有键，而另外一些没有，要使用类型 bond。

唯一可能用到类型 hybrid 的情况是没有任何一种类型包括了所有需要的性质。比如，如果你要使用极性粒子，并且要这些粒子在扭矩下旋转，那么你就需要用到 atom_style hybrid sphere dipole。如果使用类型 hybrid，那么原子就会储存由这两种类型定义的所有性质。

使用限制

该命令不能在模拟盒子被定义（使用命令 read_data 或 create_box）之后使用。

类型 angle/bond/full/molecular 是 MOLECULAR 包的一部分；类型 line/tri 是 ASPHERE 包的一部分；类型 body 是 BODY 包的一部分；类型 dipole 是 DIPOLE 包的一部分；类型 peri 是 PERI 包的一部分；类型 electron 是 USER-EFF 包的一部分；类型 meso 是 USER-SPH 包的一部分；wavepacket 是 USER-AWPMD 包的一部分。只有在 LAMMPS 编译的时候把相应的包编译进去了，这些类型才可以使用。

相关命令

read_data，pair_style

默认设置

```
atom_style atomic
```

【LAMMPS 翻译系列】boundary 命令

由 www.52souji.net 发表于 2013 年 10 月 28 日 || 3,480 浏览

boundary 命令用来设置模拟盒子的边界条件。

内容目录 [隐藏]

- 相关命令
- 默认设置

使用语法

```
boundary x y z
```

x,y,z 可取 p/s/f/m 中的一个字母或两个字母的组合。

- p：周期性边界条件 periodic
- f：非周期性固定边界条件 fixed
- s：非周期性包覆边界条件 shrink-wrapped
- m：非周期性包覆最小值边界条件 minimum value

使用举例

```
boundary p p f

boundary p fs p

boundary s f fm
```

使用介绍

设置模拟盒子沿着各个方向的边界条件。单独的一个字母会将模拟盒子沿着某个方向的两个面设置为一样的边界条件。两个字母会将这两个面分别设置为不同的边界条件。模拟盒子的初始尺寸是由命令 read_data，read_restart 或 create_box 命令设置的。
p 代表周期性边界条件，就是说原子在跨越模拟盒子的边界时，会从盒子的另外一边再进入盒子里。设置为周期性的方向的盒子尺寸是可以通过常压边界条件或盒子变形（参考命令 fix npt 和 fix deform）而发生改变的。p 必须同时用在某个方向的两个面上。

f/s/m 都将模拟盒子定义为非周期边界条件，就是说原子在穿越边界时不会再从盒子的另一面再进入盒子里。

- f 是将所对应的面设置为固定的。如果原子从这个面移动出去了，那么这个原子就丢失了。
- s 是将所对应的面设置为浮动的，不论原子在那个方向上移动到哪里，都会通过调整所对应的面的位置而将原子包围在盒子里。

- m 是在 s 的基础上定义的，虽然包覆原子的行为会发生，但却被限制在一定的范围内。这个限制值是通过数据文件或重启动文件或命令 create_box 设置的。举例说明：如果在数据文件中设置了 z 的正方向为 50.0，那么即便模拟盒子在 z 的正方向上的最大值变的比 50.0 小，z 正方向上的这个面也只会在 50.0 或者其上的位置上。

对于非正交模拟盒子，如果倾斜因子的第二个维度（比如 xy 的 y 方向）是周期性的，那么在执行周期性的时候会强制倾斜因子带来的偏移。如果第一个维度是包覆型的（即 s 或 m），那么包覆是针对倾斜面而言的。举例来说，对于一个正的 xy 倾斜因子，模拟盒子的 xlo 面和 xhi 面就是沿着+y 方向进行倾斜的面。这些倾斜面决定了模拟盒子在 x 方向的范围，原子也是被包覆在这些倾斜面内。[译注：本段建议参考原文]

使用限制

使用该命令之前，需要先定义模拟盒子（使用命令 read_data 或 create_box 或 read_restart）。
参考命令 change_box 介绍如何改变模拟盒子的边界条件。

对于二维模拟来说，z 必须设置为周期性。

相关命令

命令 thermo_modify 有关于丢失原子的介绍

默认设置

```
boundary p p p
```

LAMMPS 翻译系列】dimension 命令

由 www.52souji.net 发表于 2013 年 10 月 3 日 || 790 浏览

dimension 命令用来定义模拟的维度。

内容目录 [隐藏]

使用语法

```
dimension N
```

- N： *2 or 3*

使用举例

```
dimension 2
```

使用介绍

该命令用来定义模拟的维度。默认情况下，LAMMPS 运行三维模拟。如果要运行二维的模拟，需要在建立模拟盒子（使用命令 create_box 或 read_data）之前，先使用该命令进行设置。重启动文件也会保存该设置。

参考 how to 部分，了解如何进行二维模拟。

注意：LAMMPS 中的有些模型将粒子作为有限尺寸的球体或椭球体对待，而不是点粒子。在二维模拟时，这些粒子仍然是球体或椭球体，而不是圆或椭圆，这也就意味着它们的惯性矩与三维情形下是相同的。

使用限制

该命令必须在模拟盒子被建立（使用命令 create_box 或 read_data）之前使用。

相关命令

fix enforce2d

默认设置

```
dimension 3
```

LAMMPS 翻译系列】newton 命令

newton 命令用来开启或关闭对势或键相互作用中的第三运动定律。

内容目录 [隐藏]

使用语法

```
newton flag

newton flag1 flag2
```

- flag： 开关对势和键相互作用[on/off]
- flag1：开关对势相互作用[on/off]
- flag2：开关键相互作用[on/off]

使用举例

```
newton off

newton on off
```

使用介绍

该命令用来开启或关闭对势或键相互作用中的第三运动定律。对于大多数问题而言，将牛顿第三定律设置为开启是一种可以节省 2 倍及以上计算量的做法。具体到是否会更快，则取决于问题的规模、力截断长度、机器的计算/交换比、以及所使用的处理器数量。

将对势相互作用的 flag 设置为 off，那么如果相互作用的两个原子在不同的处理器上，两个处理器都会计算它们的相互作用，所得的关于力的结果信息也不会进行通信。类似的，对于键相互作用，将 flag 设置为 off，那么如果键、角、二面角或不合适的相互作用在 2 个或更多的处理器上，这些相互作用会被每个处理器分别计算。

但不论将 newton 命令的 flag 设置成什么，LAMMPS 计算出来的结果都是一样的，除了一些舍入误差之外。

对于 run_style respa，如果最内层时间步只计算键相互作用，那么将键相互作用的 newton 设置为 off 可以避免内层循环的外部外部通信，从而可能会变得更快。[译注：建议参考原文]

使用限制

模拟盒子定义（命令 read_data 或 create_box）以后，newton 键设置就不能再改变了。

相关命令

run_style respa

默认设置

newton on

【LAMMPS 翻译系列】units 命令

由 www.52souji.net 发表于 2013 年 10 月 1 日 || 2,007 浏览

units 命令用来定义模拟过程中使用的单位类型，它决定了所有输入脚本、数据文件和所有输出到屏幕、日志文件以及 dump 文件中物理量的单位。

内容目录 [隐藏]

使用语法

units style

- style 可取：*lj/real/metal/si/cgs/electron*

使用举例

```
units metal

units lj
```

使用介绍

该命令用来设置模拟中使用的单位类型。它决定了所有输入脚本、数据文件和所有输出到屏幕、日志文件以及 dump 文件中物理量的单位。一般来说，该命令用在输入脚本最开始的位置。

除了 lj 类型之外，其他所有的单位类型所用到的物理常量都来自网站：www.physics.nist.gov。real 类型中 Kcal 的定义，LAMMPS 使用热力学卡 =4.184J。

LJ 类型

对于 lj 类型，所有的物理量都是没有单位的。不失一般性，LAMMPS 将基本量 mass、sigma、epsilon 和波尔兹曼常数设为 1。你所指定的质量、距离、能量就是这些基本量的倍数。下面给出了无单位量（用*标出）与有单位量之间的换算关系。基于此，你可以将 lj 模拟无单位量转换为正常的物理量。

- mass = mass or m
- distance = sigma, where x* = x / sigma
- time = tau, where tau = t* = t (epsilon / m / sigma^2)^1/2
- energy = epsilon, where E* = E / epsilon
- velocity = sigma/tau, where v* = v tau / sigma
- force = epsilon/sigma, where f* = f sigma / epsilon
- torque = epsilon, where t* = t / epsilon
- temperature = reduced LJ temperature, where T* = T Kb / epsilon
- pressure = reduced LJ pressure, where P* = P sigma^3 / epsilon
- dynamic viscosity = reduced LJ viscosity, where eta* = eta sigma^3 / epsilon / tau
- charge = reduced LJ charge, where q* = q / (4 pi perm0 sigma epsilon)^1/2
- dipole = reduced LJ dipole, moment where *mu = mu / (4 pi perm0 sigma^3 epsilon)^1/2

- electric field = force/charge, where E* = E (4 pi perm0 sigma epsilon)^1/2 sigma / epsilon
- density = mass/volume, where rho* = rho sigma^dim

需要注意的是，lj 单位类型下，通过命令 thermo_style 设置的热力学信息的输出是将能量对原子数量进行了归一化，即能量/原子。这可以通过命令 thermo_modify norm 进行修改。

real 类型

- mass = grams/mole
- distance = Angstroms
- time = femtoseconds
- energy = Kcal/mole
- velocity = Angstroms/femtosecond
- force = Kcal/mole-Angstrom
- torque = Kcal/mole
- temperature = Kelvin
- pressure = atmospheres
- dynamic viscosity = Poise
- charge = multiple of electron charge (+1.0 is a proton)
- dipole = charge*Angstroms
- electric field = volts/Angstrom
- density = gram/cm^dim

metal 类型

- mass = grams/mole
- distance = Angstroms
- time = picoseconds
- energy = eV
- velocity = Angstroms/picosecond
- force = eV/Angstrom
- torque = eV
- temperature = Kelvin
- pressure = bars
- dynamic viscosity = Poise
- charge = multiple of electron charge (+1.0 is a proton)
- dipole = charge*Angstroms
- electric field = volts/Angstrom

- density = gram/cm^dim

si 类型

- mass = kilograms
- distance = meters
- time = seconds
- energy = Joules
- velocity = meters/second
- force = Newtons
- torque = Newton-meters
- temperature = Kelvin
- pressure = Pascals
- dynamic viscosity = Pascal*second
- charge = Coulombs
- dipole = Coulombs*meters
- electric field = volts/meter
- density = kilograms/meter^dim

cgs 类型

- mass = grams
- distance = centimeters
- time = seconds
- energy = ergs
- velocity = centimeters/second
- force = dynes
- torque = dyne-centimeters
- temperature = Kelvin
- pressure = dyne/cm^2 or barye = 1.0e-6 bars
- dynamic viscosity = Poise
- charge = statcoulombs or esu
- dipole = statcoul-cm = 10^18 debye
- electric field = statvolt/cm or dyne/esu
- density = grams/cm^dim

electron 类型

- mass = atomic mass units
- distance = Bohr
- time = femtoseconds

- energy = Hartrees
- velocity = Bohr/atomic time units [1.03275e-15 seconds]
- force = Hartrees/Bohr
- temperature = Kelvin
- pressure = Pascals
- charge = multiple of electron charge (+1.0 is a proton)
- dipole moment = Debye
- electric field = volts/cm

不同的单位类型，也设置了默认的时间步和领域表层距离，如下：

- For style *lj* these are dt = 0.005 tau and skin = 0.3 sigma.
- For style *real* these are dt = 1.0 fmsec and skin = 2.0 Angstroms.
- For style *metal* these are dt = 0.001 psec and skin = 2.0 Angstroms.
- For style *si* these are dt = 1.0e-8 sec and skin = 0.001 meters.
- For style *cgs* these are dt = 1.0e-8 sec and skin = 0.1 cm.
- For style *electron* these are dt = 0.001 fmsec and skin = 2.0 Bohr.

使用限制

该命令不能在模拟盒子被定义（使用命令 read_data 或 create_box）之后使用。

默认设置

```
units lj
```

【LAMMPS 翻译系列】create_atoms 命令

由 www.52souji.net 发表于 2014 年 4 月 8 日 || 2,124 浏览

create_atoms 命令用来在晶格阵点上创建原子，或创建一个单独的原子，或创建一些列随机原子。

内容目录 [隐藏]

- 默认设置

使用语法

```
create_atoms type style args keyword values ...
```

- type：要创建的原子类型（用 1 到 N 的数字代替）
- style：*box* or *region* or *single* or *random*
  - *box: args = none*
  - region: args = region-ID
    - 只有在 region 内的原子才会被创建
  - single: args = x y z
    - x,y,z 为要创建原子的坐标（以原胞为单位）
  - random: args = N seed region-ID
    - N：要创建的原子数
    - seed：随机数，用作种子（正整数）
    - region-ID：在 region 内创建原子，如果设置为 NULL，则在整个模拟盒子内创建。
- keyword：*basis* or *remap* or *units*
  - basis values = M itype
    - M：哪个基本原子
    - itype：该基本原子的类型（从 1 到 N）
  - remap value = yes or no
  - units value = lattice or box
    - lattice：以晶格距离作为单位
    - box：以模拟盒子作为单位

使用举例

```
create_atoms 1 box

create_atoms 3 region regsphere basis 2 3

create_atoms 3 single 0 0 5
```

使用介绍

该命令用来在晶格阵点上创建原子，或创建一个单独的原子，或创建一些列随机原子。也可以用命令 read_data 或 read_restart 通过直接给出原子坐标的方式创建

原子。在使用该命令之前，模拟盒子必须是存在的（使用 create_box 命令创建），同时晶格也必须已经被定义（使用 lattice 命令）。但对于创建 single 类型且以 box 为单位的原子，或创建 random 类型的原子时，不需要先定义晶格。

box 类型：该命令在整个模拟盒子中所有的晶格阵点上创建原子。如果你的模拟盒子是周期性的，你应该确保其尺寸是晶格距离的整数倍，从而避免在盒子边界处可能存在的原子重叠。如果你的盒子是周期性的，并且在某个方向上盒子的尺寸是晶格距离的整数倍，那么 LAMMPS 会在边界上只放置一个原子（译注：边界上的原子不会丢失或在两个面同时出现）。

region 类型：该命令会在 region-ID 所指定的区域与模拟盒子相交的公共区域内创建原子。参考命令 region，可以了解更多细节。需要注意的是，这里定义的区域可以在模拟盒子内，也可以在模拟盒子外。还需要注意，即便你在这里定义的区域与周期性模拟盒子的尺寸是一样的，LAMMPS 所执行的逻辑跟 box 类型也不一样，就是说并不能像 box 类型那样确保在边界上只有一个原子。所以如果你希望达成像 box 类型那样的效果，你最好使用 box 类型，或者就是非常精确的调整 region 的尺寸来获得你想要的原子。

single 类型：将指定坐标的原子添加到系统中。这对于调试或者创建一些列手动添加的原子会比较有用。

random 类型：在系统中按着随机坐标产生 N 各原子，这对于产生无定形系统会比较有用。根据指定的随机种子数 seed，程序会依次创建 N 个随机原子。这 N 个随机原子的坐标与所使用处理器的个数没有关系。另外，如果 region-ID 设置为 NULL，那么创建的原子会随机填充模拟盒子的任何位置；如果指定了特定的 region-ID，那么原子就只会随机填充在模拟盒子与 region-ID 所共同指定的区域。

注意：使用 random 类型产生的原子很有可能是相互重叠的（译注：或者是距离较近，不太符合实际物理），从而导致计算出较大的力或能量。因此，在你开始进行正常的动力学计算时，最好先使用 minimize 对体系进行能量最小化，或先使用 fix nve/limit 进行动力学计算。

该命令是向已经存在的体系中继续添加原子。换句话说，该命令可以多次使用，从而可以在模拟盒子中创建多组原子。举例来说，通过交错地使用 create_atoms 命令和 lattice 命令（配置为不同晶向 orientations），就可以创建晶界；联合使用 create_atoms 命令和 delete_atoms 命令，可以创建非常复杂的体系。

create_atoms 命令也可以在已经读入的体系中继续创建原子。在所有列举的这些情况中，都需要注意不要让新创建的原子与已经存在的原子重叠。可以使用 delete_atoms 命令来处理重叠原子的问题。

---

basis 关键字：用来为要创建的特定的基本原子指定原子类型。参考 lattice 命令，了解如何为原胞定义基本原子。默认情况下，所有创建的原子都被指定为参数 type 说指定的原子类型。
remap 关键字：仅仅在 single 类型下使用。如果将其设置为 yes，那么当指定的位置在模拟盒子外面时，它会将其挪到盒子里面，就好像在相应的方向上设置了周期性边界条件一样。如果将其设置为 no，就不会进行类似上面的调整，就是说如果创建原子的位置在模拟盒子外面，那么就不会创建原子。
units 关键字：用来设定要创建的 single 类型的那个原子的坐标的单位。若取值 box，坐标的单位就与命令 units 说指定的距离单位相同，比如 units 设为 real 或 metal 时，其单位为 Angstroms。若取值 lattice，坐标是以晶格距离为单位的。

所创建的原子的 ID 是遵从如下规律：所有待创建原子的 ID 是连续的，其中第一个被创建原子的 ID 是紧接着体系中已经存在原子的最大的 ID。使用不同数目的处理器进行计算时，不能保证某个特定的原子会分配到同样的 ID。

除了原子的 ID 之外，原子类型、坐标以及其他性质都设为默认值。具体到有哪些性质，是由 atom style 决定的。参考 set 和 velocity 命令，了解如何改变这些默认值。

- charge = 0.0
- dipole moment magnitude = 0.0
- diameter = 1.0
- shape = 0.0 0.0 0.0
- density = 1.0
- volume = 1.0
- velocity = 0.0 0.0 0.0
- angular velocity = 0.0 0.0 0.0
- angular momentum = 0.0 0.0 0.0
- quaternion = (1,0,0,0)
- bonds, angles, dihedrals, impropers = none

sphere 类型会将默认的粒子直径和密度设置为 1.0，那么粒子对应的质量就不是 1.0，而是 PI/6 * diameter^3 = 0.5236.
ellipsoid 类型会将默认的粒子形状设置为(0.0 0.0 0.0)，密度设置为 1.0，也就是一个点粒子，而不是真正的 ellipsoid，并且质量为 1.0。
peri 类型会将默认的体积和密度设置为 1.0，因此粒子的质量也就是 1.0。
可以使用 set 命令来重写这些默认设置。

使用限制

在使用该命令之前，必须已经定了 atom_style。

相关命令

lattice, region, create_box, read_data, read_restart

默认设置

- basis 的默认值是所有创建的原子都分配参数 type 所指定的原子类型。
- remap 的默认值是 no。
- units 的默认值是 lattice。
- **LAMMPS 命令官方手册**
- **LAMMPS 中文翻译全部命令索引**

【LAMMPS 翻译系列】read_restart 命令

read_restart 命令用来读入之前的模拟过程保存下的重启动文件。

内容目录 [隐藏]

使用语法

```
read_restart file
```

- file：要读入的二进制重启动文件的文件名。

使用举例

```
read_restart save.10000

read_restart restart.*

read_restart poly.*.%
```

使用介绍

该命令用来读入之前的模拟过程保存下的重启动文件。该命令可以帮助你实现接着之前的模拟过程继续进行。下面会告诉你哪些信息被存储在了重启动文件中。

重启动文件是二进制格式的，它是用来精确重启动之前模拟过程，就是说重启动之后的模拟会精确的接着之前已经进行的模拟继续进行。

但也有一些因素会因为舍入误差而影响到这种重启动的精确性，这种情况下这两个模拟间就会有一些偏差。这些影响因素包括使用数量不同的处理器进行计算，或者改变了某些设置，比如那些与命令 newton 或 processors 相关的设置。LAMMPS 会在这种情况下给出警告。

有些 fix 命令不能被重启动，尽管它们会提供相近的统计结果。这种命令包括 fix shake 和 fix langevin。

有些势类型不能被重启动，尽管它们会提供相近的统计结果。这是因为力的计算是依赖于原子速度的，而在每个时间步计算力的时候，使用的是半个时间步的速度值。当重启动一个计算时，力的计算却被初始化为全步的速度值，这就与写入重启动文件时的计算直接进行下去的结果不同。这些势类型有 granular pair styles, pair dpd 和 pair lubricate。

如果你在重启动一个计算时，从一开始就与产生重启动文件的计算有了不一样，这应该是 LAMMPS 的 bug，你可以报告给 LAMMPS 软件。

因为重启动文件是二进制的，所以这些文件一般来说是不能拷贝到其他的机器使用的，但你可以使用工具 restart2data 将其转换成文本文件。
与写重启动文件（参考命令 write_restart 和 restart）类似，读入的重启动文件的文件名也可以包括两种通配符。

- 如果是符号“*”，那么会在当前目录下匹配所有文件名中将“*”替换成时间步的文件，但只有最大时间步的那个文件会被读入。也就是说，最后写入的那个重启动文件会被读入用于重启动模拟。如果你想让你的计算从之前停下来的地方开始，那么这种方式非常实用。参考命令 run 和它的 upto 选项，了解怎样设置 run 命令可以不需要在新的计算中更改命令 run 的设置。
- 如果是符号“%”，LAMMPS 会要求有一系列的重启动文件存在。在命令 restart 和 write_restart 命令中，已经介绍了这一些列重启动文件是如何创建的。read_restart 命令首先会读入将“%”替换为 base 的那个文件，该文件告诉 LAMMPS 之前的模拟设置了多少个处理器。然后 read_restart 命令会读入剩下的那些重启动文件。举例来说，如果在写入的时候，设置重启动文件的文件名为 save.%，那么在使用

read_restart 读入的时候，就会读入这些文件 save.base, save.0, save.1, ..., save.P-1（这里面 P 是创建重启动文件时处理器的数量）。在当前的 LAMMPS 模拟中，所有的处理器同时读入这些文件，每个处理器会读入文件数量大致相等。创建重启动文件的处理器数量可以跟读入重启动文件的处理器数量不一样。对于支持并行 I/O 的并行机来说，这是一种快速的输入方式。

对一个模拟而言，下列信息会被存储在了重启动文件中：单位样式 units、原子样式 atom style、模拟盒子的尺寸和形状、边界条件、定义的组、与原子类型相关的设置（如质量）、与每个原子相关性质（包括它们被分配到的组和分子拓扑性质）、势函数类型及参数、特殊键 specail_bonds 的设置。这就意味着与这些量相关的命令不需要在读入重启动文件中的输入脚本中再重新进行指定。当然，你也可以重新指定这些设置。

有一个例外是有些势类型的信息没有存储在重启动文件中。具体哪种类型，会在介绍那种势类型的时候专门指出来。命令 bond_style hybrid (and angle_style, dihedral_style, improper_style hybrid) 也是这样的。

所有由命令 pair_modify 所指定的设置，比如 shift 和 tail，会与势类型一起存储在重启动文件中。但命令 pair_modify compute 是唯一的例外。

命令 kspace_style 的相关信息没有存储在重启动文件中，所以如果你希望使用 Ewald or PPPM 求解器，那么这些命令还需要在重启动文件被读入后重新指定。

所有与 fix 相关的命令都没有存储在重启动文件中。这就意味着在新的输入脚本中需要重新指定所有用到的 fix 命令。需要注意的是，有一些 fix 命令会将它们内在的状态写入到重启动文件中，这就使得在开始重启动模拟之后，这些 fix 命令可以继续起作用，但这些却需要被激活。你只需要在新的输入脚本中使用与写入重启动文件的那个脚本相同的 fix-ID 和 group-ID 来定义这些 fix 命令，就可以将它们激活。如果 fix 命令能够匹配上，那么 LAMMPS 会输出一个信息，指示这个 fix 命令被重新激活了。如果在执行新输入脚本中计算或能量最小化之前没有发现可以匹配的 fix 命令，那么从重启动文件保存的 fix 命令的内在状态信息就会被丢弃掉。参考 fix 命令，了解具体是哪个具有这样的重启动方式。

使用命令 fix_shake 或 delete_bonds 关掉的键作用，在被写入到重启动文件时，会被打开。所以，在读入重启动文件后，需要重新将这些键作用关掉。

断掉的键也被写入到了重启动文件中（断键用类型 0 表示），因此当读入重启动文件的时候，这些键仍然是断掉的。

注意：除了上面列举的信息之外，其他任何信息都没有被存储在重启动文件中。这就是说，对于读入重启动文件的输入脚本来说，需要指定的量有：时间步长 timestep、热力学信息 thermo_style、领域列表 neighbor、领域列表设置 neigh_modify、dump 文件输出、几何区域 region 等。

相关命令

read_data, read_dump, write_restart, restart

默认设置

```
restart 0
```

【LAMMPS 翻译系列】region 命令

由 www.52souji.net 发表于 2014 年 4 月 10 日 || 3,428 浏览

region 命令用于定义一个空间几何区域。

内容目录 [隐藏]

使用语法

```
region ID style args keyword arg ...
```

- ID = 待定义区域的 ID
- style = *delete* / *block* / *cone* / *cylinder* / *plane* / *prism* / *sphere* / *union* / *intersect*

- *delete* = no args
- *block* args = xlo xhi ylo yhi zlo zhi
- xlo,xhi,ylo,yhi,zlo,zhi = 各个维度上的范围值
- *cone* args = dim c1 c2 radlo radhi lo hi
- dim = *x* or *y* or *z* = 圆锥台轴线方向
- c1,c2 = 圆锥台轴线在另外两个维度上的坐标值

- radlo,radhi = 圆锥台上下底面的半径
- lo,hi = 圆锥台轴线方向的范围值
- *cylinder* args = dim c1 c2 radius lo hi
- dim = *x* or *y* or *z* = 圆柱体轴线方向
- c1,c2 = 圆柱体轴线在两位两个维度上的坐标值
- radius = 圆柱体半径，可以使用变量
- lo,hi = 圆柱体轴线方向的范围值
- *plane* args = px py pz nx ny nz
- px,py,pz = 平面上一个点的坐标
- nx,ny,nz = 平面法矢坐标
- *prism* args = xlo xhi ylo yhi zlo zhi xy xz yz
- xlo,xhi,ylo,yhi,zlo,zhi = 平行六面体在各个维度上的范围值
- xy = y 方向在 x 方向的倾斜值
- xz = z 方向在 x 方向的倾斜值
- yz = z 方向在 y 方向的倾斜值
- *sphere* args = x y z radius
- x,y,z = 球心坐标
- radius = 球体半径，可以使用变量
- *union* args = N reg-ID1 reg-ID2 ...
- N = 要进行并集操作的区域的数目，必须大于等于 2
- reg-ID1,reg-ID2, ... = 要进行并集操作的区域的 ID
- *intersect* args = N reg-ID1 reg-ID2 ...
- N = 要进行交集操作的区域的数目，必须大于等于 2

reg-ID1,reg-ID2, ... = 要进行交集操作的区域的 ID

- keyword = *side* or *units* or *move* or *rotate*

- *side* value = *in* or *out*
- *in* = 指定几何体内侧作为区域
- *out* = 指定几何体外侧作为区域
- *units* value = *lattice* or *box*
- *lattice* = 以晶格距离作为几何距离单位
- *box* = 以模拟盒子作为几何距离单位
- *move* args = v_x v_y v_z
- v_x,v_y,v_z = x, y, z 方向的位移量，equal 类型的变量
- *rotate* args = v_theta Px Py Pz Rx Ry Rz
- v_theta = 旋转角，equal 类型的变量，弧度单位
- Px,Py,Pz = 旋转操作的中心点
- Rx,Ry,Rz = 旋转轴矢量

使用举例

```
region 1 block -3.0 5.0 INF 10.0 INF INF

region 2 sphere 0.0 0.0 0.0 5 side out

region void cylinder y 2 3 5 -5.0 EDGE units box

region 1 prism 0 10 0 10 0 10 2 0 0

region outside union 4 side1 side2 side3 side4

region 2 sphere 0.0 0.0 0.0 5 side out move v_left v_up NULL
```

使用介绍

该命令用于定义一个空间几何区域。很多其他命令都会使用该命令定义的区域。举例来说，create_atoms 可以在定义区域中创建原子，create_box 命令可以根据定义区域定义模拟盒子，group 命令可以将定义区域中包括的原子定义为一个组，delete_atoms 命令可以将定义区域中的原子删掉，fix wall/region 可以将定义区域的表面定义为壁面（boundary wall）。

大多数使用 region 命令所定义区域的命令都需要检测某个原子的位置是否在区域内。刚好在区域边界上原子被认为是属于所定义区域的。举个例子，对于一个球形区域而言，如果在定义的时候使用了关键字 side in，那么球形区域表面上的原子就是该定义区域的一部分；如果使用了关键字 side out，那么球形区域表面上的原子就不属于该区域的一部分。详细可以参考下面的 side 关键字。

一般来说，LAMMPS 所定义的区域都是静态的，也就说所定义的几何空间范围并不会随着时间而改变。但如果使用了关键字 move 或 rotate，如下文所要介绍的，区域就会变成动态的，也就是说所定义区域的位置和朝向会随着时间发生改变。在有些时候，这种动态区域是比较有用的，比如在使用命令 compute temp/region 对一个区域进行恒温，或使用 fix wall/region 命令将包含运动粒子的区域表面定义为壁面时。

delete 类型：删除指定的区域。因为定义冗余的区域并不会占用太大的开销，所以一般不需要这样做，除非在你的输入脚本中定义了大量了区域。

***block* / *cone* / *cylinder* / *prism* 类型**中参数 lo/hi 的值可以指定为 EDGE 或 INF。

- EDGE 是指所定义方向一直延伸到所在方向模拟盒子的边界。需要注意的是，这里所说模拟盒子的边界是指当前盒子的边界。换句话说，如果模拟盒子在模拟的过程中发生了改变，按着这种方式定义的区域并不会随着改变。
- INF 是指一个非常大的负数或正数(1.0e20)，所以即便模拟盒子在模拟过程中发生改变了，使用该定义仍然能将其包括到区域中。

如果模拟盒子还没有创建，那么在使用 region 定义区域的时候就不能使用 EDGE 或 INF（译注：这是显然的，因为 EDGE 和 INF 都是相对于模拟盒子来说的）。

对于 **prism 类型**的区域，如果某个倾斜因子不为 0，那么它所对应的两个维度上 lo/hi 都不能设为 INF。举例来说，如果 xy 倾斜因子设为非零，那么 xlo/xhi 和 ylo/yhi 都不能设为 INF。

注意 1：Regions in LAMMPS do not get wrapped across periodic boundaries, as specified by the boundarycommand. For example, a spherical region that is defined so that it overlaps a periodic boundary is not treated as 2 half-spheres, one on either side of the simulation box.

注意 2：不论是否用 dimension 命令将模拟设为 2d 或 3d，LAMMPS 中所定义的区域总是三维的。所以在进行 2d 模拟时，你需要仔细定义区域，以使其与 2d 的 x-y 平面相交，并产生正确的交集区域。

cone 类型（圆锥台）：定义一个轴对称的圆锥台。除了要定义两个不同的底面半径，它的定义过程基本上与圆柱 cylinder 是一样的。两个底面半径都可以设置为 0，但不能同时设为 0.

cone 和 cylinder 类型：参数 c1、c2 是轴线在另外两个维度上的坐标参数。如果 dim=x，那么 c1/c2 分别对应 y/z；如果 dim=y，那么 c1/c2 分别对应 x/z；如果 dim=z，那么分别对应 x/y。因此，上面第三个例子定义了一个圆柱体，其轴线沿着 y 方向，轴线坐标为 x=2.0, z=3.0，半径为 5.0，轴线范围是从-5.0 到模拟盒子的上边界。

plane 类型：定义一个过指定点(px,py,pz)，法矢为(nx,ny,nz)的平面。所指定的法矢不需要是单位矢量。平面的内部（inside）是指法矢所指的那个半空间。

prism 类型：定义平行六面体式的区域。其原点坐标为(xlo,ylo,zlo)，从原点出发的三个基矢为 A = (xhi-xlo,0,0); B = (xy,yhi-ylo,0); C = (xz,yz,zhi-zlo)。其中 xy,xz,yz 叫做倾斜因子，它们可以取 0.0，正值，或负值。它们实际上是正交盒子变换为平行六面体时各个面的偏斜位移值。

如果要使用 create_box 命令，并以已经定义为 prism 类型的区域为范围定义三斜式的模拟盒子，那么倾斜因子(xy,xz,yz）不能超过各个维度上对应盒子长度的一半。举个例子，如果 xlo=2, xhi=12，那么模拟盒子的长度就是 10，这也就要求 xy 倾斜因子必须在-5 到 5 之间。类似地，倾斜因子 xz 和 yz 必须在-(xhi-xlo)/2 和

+(yhi-ylo)/2 之间。需要注意的是，这并不是一个限制，因为如果最大的倾斜因子应该是 5，那么将其设置为-15, -5, 5, 15, 25.....在几何上都是等效的。
sphere 和 cylinder 类型中参数 radius 可以设置为 equal 类型的变量 variable。如果设置为变量，那么其形式就要求是 v_name，其中 name 是变量名。这种情况下，每个时间步都会计算变量的值，并将得到的值设为半径。
equal 类型的变量可以使用很多数学函数来定义公式，也可以使用 thermo_style 命令中关键字。
union 类型：创建所有列举出区域的并集区域。
intersect 类型：创建所有列举出区域的交集区域。
注意：union 和 intersect 类型的区域依赖于产生它们的每一个子区域，因此在定义 union 或 intersect 类型的区域后，不能删除这些子区域。

---

side 关键字：决定所考虑的区域是所指定几何体的内部还是外部。通过将该关键字与 union 和 intersect 类型配合使用，可以构建复杂的几何区域。举个例子，如果将两个球体的内部分别定义成了区域，并对这两个区域使用类型为 union，side=out 的操作，那么所定义的区域就是整个模拟盒子之内、除了两个球体之外的所有空间。
units 关键字：决定前面所列举的所有用于定义区域的长度量的单位。若取值 box，坐标的单位就与命令 units 说指定的距离单位相同，比如 units 设为 real 或 metal 时，其单位为 Angstroms。若取值 lattice，坐标是以晶格距离为单位的，但前提是必须已经使用 lattice 命令定义了晶格距离。具体到不同类型，在不同维度上是不同的，如下：

- *block 类型*：xlo/xhi 是以 x 方向的晶格距离为单位。类似地，ylo/yhi 和 zlo/zhi 分别是以 y 和 z 方向的晶格距离为单位。
- *cone 类型*：lo/hi 是以 dim 所在方向的晶格距离为单位。c1/c2 则分别以另外两个径向维度方向上的晶格距离为单位。圆锥台上下底面的半径是以与 c1 对应的那个方向上的晶格距离为单位。
- *cylinder 类型*：lo/hi 是以 dim 所在方向的晶格距离为单位。c1/c2 则分别以另外两个维度方向上的晶格距离为单位。圆柱体半径是以与 c1 对应的那个方向上的晶格距离为单位。
- *plane 类型*：px 和 nx 是以 x 方向的晶格距离为单位。类似地，py/ny 是以 y 方向的晶格距离为单位，pz/nz 是以 z 方向的晶格距离为单位。
- *prism 类型*：xlo/xhi 是以 x 方向的晶格距离为单位。类似地，ylo/yhi 和 zlo/zhi 分别是以 y 和 z 方向的晶格距离为单位。xy 和 xz 是以 x 方向的晶格距离为单位，yz 是以 y 方向的晶格距离为单位。

- *sphere 类型*：球形坐标 x,y,z 分别是以 x,y,z 方向的晶格距离为单位，球半径是以 x 方向的晶格距离为单位。

如果使用 move 或 rotate 关键字，那么所定义的区域就变成了动态的，也就是说它的位置和朝向是会随着时间发生变化。这两个关键字不能用于 union 或 intersect 类型的区域。

Instead, the keywords should be used to make the individual sub-regions of the *union* or *intersect* region dynamic. Normally, each sub-region should be "dynamic" in the same manner (e.g. rotate around the same point), though this is not a requirement.
The *move* keyword allows one or more equal-style variables to be used to specify the x,y,z displacement of the region, typically as a function of time. A variable is specified as v_name, where name is the variable name. Any of the three variables can be specified as NULL, in which case no displacement is calculated in that dimension.
Note that equal-style variables can specify formulas with various mathematical functions, and include thermo_style command keywords for the simulation box parameters and timestep and elapsed time. Thus it is easy to specify a region displacement that change as a function of time or spans consecutive runs in a continuous fashion. For the latter, see the *start* and *stop* keywords of the runcommand and the *elaplong* keyword of thermo_style custom for details.

For example, these commands would displace a region from its initial position, in the positive x direction, effectively at a constant velocity:

```
variable dx equal ramp(0,10)
region 2 sphere 10.0 10.0 0.0 5 move v_dx NULL NULL
```

Note that the initial displacemet is 0.0, though that is not required.

Either of these varaibles would "wiggle" the region back and forth in the y direction:

```
variable dy equal swiggle(0,5,100)
variable dysame equal 5*sin(2*PI*elaplong*dt/100)
```

```
region 2 sphere 10.0 10.0 0.0 5 move NULL v_dy NULL
```

The *rotate* keyword rotates the region around a rotation axis *R* = (Rx,Ry,Rz) that goes thru a point *P* = (Px,Py,Pz). The rotation angle is calculated, presumably as a function of time, by a variable specified as v_theta, where theta is the variable name. The variable should generate its result in radians. The direction of rotation for the region around the rotation axis is consistent with the right-hand rule: if your right-hand thumb points along *R*, then your fingers wrap around the axis in the direction of rotation.
The *move* and *rotate* keywords can be used together. In this case, the displacement specified by the *move* keyword is applied to the *P* point of the *rotate* keyword.

使用限制

prism 类型的区域在任何方向上都不能是 0 厚度。对于 2d 模拟而言，在 z 方向上可以定义一个很小的厚度，而 xz 和 yz 必须是 0。

相关命令

lattice, create_atoms, delete_atoms, group

默认设置

side=in, units=lattice, 无移动或旋转。

【LAMMPS 翻译系列】pair_write 命令

由 www.52souji.net 发表于 2013 年 11 月 5 日 || 1,059 浏览

pair_write 命令将原子对间所定义的势函数，以距离作为自变量，将对应的能量和受力写入到文件中。

内容目录 [隐藏]

使用语法

```
pair_write itype jtype N style inner outer file keyword Qi Qj
```

- itype,jtype = 2 atom types
- N = # of values
- style = *r* or *rsq* or *bitmap*
- inner,outer = inner and outer cutoff (distance units)
- file = name of file to write values to
- keyword = section name in file for this set of tabulated values
- Qi,Qj = 2 atom charges (charge units) (optional)

使用举例

```
pair_write 1 3 500 r 1.0 10.0 table.txt LJ

pair_write 1 1 1000 rsq 2.0 8.0 table.txt Yukawa_1_1 -0.5 0.5
```

使用介绍

该命令将原子对间所定义的势函数，以距离作为自变量，将对应的能量和受力写入到文件中。如果你想绘制势函数曲线或调试势函数，这个命令会比较有用。在写文件的时候，如果文件已经存在了，那么数据会添加到文件的末尾，这样就可以将多个能量-受力数据块写入到同一文件中。

在命令中，itype 和 jtype 指定要计算能量和力的原子类型，inner 和 outer 为要计算的距离区间，计算中所用到的其他参数由相应的 pair_coeff 命令指定。如果这里设置的 style 是 r，那么会输出 N 个在 r 上均匀分布的能量-受力；如果 style 是 rsq，那么会输出 N 个距离在 r^2 上均匀分布的能量-受力。

举个例子，如果 N=7，style=r，inner=1.0，outer=4.0，那么会计算的值有：r = 1.0, 1.5, 2.0, 2.5, 3.0, 3.5, 4.0。

如果 style 是 bitmap[位图]，那么会有 2^N 个数据按着特定的格式写入到文件中，其顺序与 pair_coeff 命令读入 pair_style table 所指定文件的顺序是一致的。为了能够保证位图数据表的精度，需要设置：N>=12，inner 小于最近的两个原子间的距离，outer 小于等于势函数的截断。
如果要计算势函数的*原子对*是有电荷的，那么可以为这对原子指定电荷量。如果没有指定，默认会使用 Qi=Qj=1.0。

所写出文件的格式是与 pair_style 命令中 table 类型所需要的输入文件是一样的，其实 keyword 指定的是数据段的名称。写入到文件中的每一行都是由序号(1-N)、原子距离(距离单位)、能量（能量单位）、受力（力单位）构成的。

使用限制

所有原子对间的力场参数和其他类型的相互作用都必须在使用该命令之前已经被设置好了。

受限于程序中原子对间力的计算方式，inner 必须设置为大于 0 的数，即便在 r=0 所对应的相互作用是有限值。

对于 EAM 势函数而言，该命令只能输出其对势项，而不能输出嵌入项。

相关命令

pair_style, pair_coeff

【LAMMPS 翻译系列】mass 命令

由 www.52souji.net 发表于 2014 年 4 月 4 日 || 961 浏览

mass 命令为某一种或几种类型的原子设置质量。

内容目录 [隐藏]

使用语法

```
mass I value
```

- I = 原子类型
- value = 质量值

使用举例

```
mass 1 1.0

mass * 62.5
```

```
mass 2* 62.5
```

使用介绍

为某一种或几种类型的原子设置质量。每种类型原子的质量也可以在 data file 文件中使用 Masses 关键字设置，并由 read_data 命令读入。参考 units 命令，了解不同单位制下质量的单位。

索引 I 可以使用下面两种方式中的任意一种进行设置：

1. 显式地设置为某个数值，为某种特定类型的原子设置质量，如上面的第一个例子。
2. 使用星号通配符，同时为多种类型的原子设置质量。可以接受的通配符样式有：“*” or “*n” or “n*” or “m*n”。如果用 N 表示原子类型的数量，那么：
   - 只使用星号，即“*”，就表示从 1 到 N 的所有原子类型。
   - 以星号开头的样式，如“*n”，就表示从 1 到 n（包括在内）的所有原子类型。
   - 以星号结尾的样式，如“n*”，就表示从 n 到 N（包括在内）的所有原子类型。
   - 星号在中间的样式，如“m*n”，就表示从 m 到 n（包括在内）的所有原子类型。

在 data file 文件中，使用 Masses 关键字设置原子类型的质量与输入脚本中使用 mass 命令类似，区别在于不能使用星号通配符。举例来说，上面的第一个例子在 data file 文件中设置时，只需要在 Masses 关键字下面设置如下的文本：

```
1 1.0
```

需要注意的是，只有当 atom_style 命令定义的原子类型有类原子质量（per-type atom mass）属性时，才需要使用 mass 命令来定义质量。目前来说，除了 sphere, ellipsoid, peri 之外，其他所有的类型都需要定义原子质量。对于这些特殊的类型，需要为每一个粒子设置质量。单原子质量可以通过 read_data 命令从 data file 文件中读入，也可以使用 create_atoms 命令设置为默认值，还可以通过 set mass 或 set density 命令设置为新值。（译注：类原子质量是值一类原子具有相同的质量，而单原子质量是指不同原子有不同的质量。）
另外需要注意的是，pair_style eam 命令所使用的 EAM 势函数文件中有原子类型质量的设置，因此不需要再额外使用 mass 命令重复定义原子质量。

如果你使用包括一种或多种子类型的命令 hybrid atom style，那么你就需要同时定义每种类型的原子质量和子类型中每个原子的质量。程序在运行中，会忽略类原子质量，而使用单原子质量。

使用限制

只有在模拟盒子定义（read_data, read_restart, create_box）好了之后，才能使用该命令。
在运行模拟之前，需要定义所有类型原子的质量。在使用命令 velocity or fix shake 之前，也需要定义原子质量。

设置的质量值必须大于 0。

【LAMMPS 翻译系列】min_style 命令

min_style 命令为 minimize 命令选择一种能量最小化的算法。

内容目录 [隐藏]

使用语法

```
min_style style
```

- style = cg or hftn or sd or quickmin or fire

使用举例

```
min_style cg

min_style fire
```

使用介绍

该命令为 minimize 命令选择一种能量最小化的算法。
[译注：一般用户只需要了解哪种算法效果好就可以了，不需要具体了解算法的细节。]

cg 类型是选择 Polak-Ribiere 版本的共轭梯度法 conjugate gradient (CG)。At each iteration the force gradient is combined with the previous iteration information to compute a new search direction perpendicular (conjugate) to the previous search direction. The PR variant affects how the direction is chosen and how the CG method is restarted when it ceases to make progress. 对于绝大多数的问题来说，PR 版的共轭梯度算法都被认为是效果最好的。

hftn 类型是 Hessian-free truncated Newton 算法。At each iteration a quadratic model of the energy potential is solved by a conjugate gradient inner iteration. The Hessian (second derivatives) of the energy is not formed directly, but approximated in each conjugate search direction by a finite difference directional derivative. When close to an energy minimum, the algorithm behaves like a Newton method and exhibits a quadratic convergence rate to high accuracy. 在大多数情况下，hftn 算法的跟 cg 算法效果相近，不过在 cg 算法出现问题时，可以考虑更换选择 hftn 算法。hftn 类型不受 min_modify 命令影响。

sd 类型是最速下降算法 steepest descent。在每个迭代步中，其搜索方向 is set to the downhill direction corresponding to the force vector （能量的负梯度方向）。一般来说，sd 算法都没有 cg 算法收敛快，但在某些特定的情形下可能会根据稳定。

quickmin 类型是 Sheppard 提出的一种阻尼动力学算法，其阻尼参数与速度矢量沿着每个原子当前力矢量上的投影相关。该类型在进行能量最小化之前把每个原子的速度被初始化为 0.0。
fire 类型是 Bitzek 提出的一种阻尼动力学算法，它与 quickmin 算法类似，but adds a variable timestep and alters the projection operation to maintain components of the velocity non-parallel to the current force vector. 该类型在进行能量最小化之前把每个原子的速度被初始化为 0.0。
quickmin 算法和 fire 算法对于 NEB 计算（使用 neb 命令）会比较有用。
注意：类型 quickmin 和 fire 暂时还不支持 fix box/relax 命令，也不支持在 eFF 模型中包含电子半径[electron radius]的能量最小化。

使用限制

无

相关命令

min_modify, minimize, neb

默认设置

```
min_style cg
```

(Sheppard) Sheppard, Terrell, Henkelman, J Chem Phys, 128, 134106 (2008). See ref 1 in this paper for original reference to Qmin in Jonsson, Mills, Jacobsen. **(Bitzek)** Bitzek, Koskinen, Gahler, Moseler, Gumbsch, Phys Rev Lett, 97, 170201 (2006).

【LAMMPS 翻译系列】reset_timestep 命令

由 www.52souji.net 发表于 2013 年 10 月 20 日 || 908 浏览

reset_timestep 命令将时间步的计算器设置为指定值。

内容目录 [隐藏]

使用语法

```
reset_timestep N
```

- N：时间步的步数。

使用举例

```
reset_timestep 0

reset_timestep 4000000
```

使用介绍

该命令将时间步的计算器设置为指定值。当使用命令 read_restart 读入重启动文件，或在运行一个模拟过程的时候，时间步数就会被设置；但如果你希望重置时间步数为某个数值，可以使用该命令。
命令 read_data 和 create_box 会将时间步数设置为 0；read_restart 命令会将时间步数设置为重启动文件中写入的值。

使用限制

当你定义了一些记录 运行的时间步，并基于此进行一些与时间有关的操作时，则不能使用该命令。命令 fix deposit 和 fix dt/reset 就是这样的两个例子。前面一个命令是在指定的时间步里添加原子；后面一个命令是记录累积的时间。

很多 fix 命令会使用当前的时间步来计算相关的量。如果时间步被重新设置的了，这可能会带来难以预测的结果。因为即便时间步被重置了，LAMMPS 也会运行定义的 fix 命令。举例来说，对系统进行恒温控制的命令，比如命令 fix nvt，允许你指定一个目标温度，并通过一定的时间步将温度从从 Tstart 变为 Tstop。但如果你改变时间步数，这可能会带来的结果就是目标温度瞬时被改变。
如果 compute 命令已经计算出了一些量，但在之后使用了重置该命令重置了时间步数，那么它就会清除与 compute 命令相关的标记。这就是说，compute 命令已经计算出来的这些量不能再通过变量进行引用，除非你又重新运行了新的计算。参考变量 variable，了解更多细节。

相关命令

rerun

【LAMMPS 翻译系列】timestep 命令

由 www.52souji.net 发表于 2013 年 10 月 23 日 || 877 浏览

timestep 为该命令之后的分子动力学模拟设置时间步长。

内容目录 [隐藏]

使用语法

```
timestep dt
```

- dt：时间步长（以时间为单位）。

使用举例

```
timestep 2.0

timestep 0.003
```

使用介绍

该命令为之后的分子动力学模拟设置时间步长。参考命令 units，了解时间步长的单位。默认的时间步长依赖于模拟中所采用的单位类型，参考下面的默认设置。如果命令 run_style 设置为 respa，那么参数 dt 设置的是最外层循环的时间步长。

相关命令

fix dt/reset, run, run_style respa, units

默认设置

- timestep = 0.005 tau for units = lj
- timestep = 1.0 fmsec for units = real
- timestep = 0.001 psec for units = metal
- timestep = 1.0e-8 sec (10 nsec) for units = si or cgs

【LAMMPS 翻译系列】fix 命令

由 www.52souji.net 发表于 2013 年 12 月 4 日 || 3,067 浏览

fix 命令为一组原子施加 fix 约束。

内容目录 [隐藏]

使用语法

```
fix ID group-ID style args
```

- ID = fix 命令的 ID
- group-ID = 该 fix 命令所作用的原子组的 ID
- style = fix 类型名（最下有列表）
- args = 特定类型 fix 命令所需要的参数

使用举例

```
fix 1 all nve

fix 3 all nvt temp 300.0 300.0 0.01

fix mine top setforce 0.0 NULL 0.0
```

使用介绍

为一组原子施加 fix 约束。在 LAMMPS 中，fix 是施加在分子动力学时间步或能量最小化过程中的某种操作。举例来说，它可能是在时间积分的过程中更新原子的位置和速度，或是控制温度，或是给原子施加约束力，或是强制某种边界条件，或计算过程诊断，等等。在 LAMMSP 中有一些列可以使用的 fix 命令，具体参考后面的列表。

fix 命令会在时间步中的不同过程中执行相关的操作。如果有 2 个或多个 fix 命令作用于时间步中的同一过程，它们会按着在输入脚本中定义的顺序依次被激活。

fix 命令的 ID 只能包含字母、数字和下划线。

fix 命令所施加的约束可以被 unfix 命令删去。
注意：使用 unfix 命令是唯一用来关闭 fix 命令所施加的约束的方法。如果只是指定一种新的类似的 fix 命令并不会关闭之前的 fix 命令。这对于进行积分的 fix 命令特别重要。举例来说，如果你先定义了 fix nvt 命令，然后又定义了 fix nve 命令，这样做并不会将先定义的 NVT 时间积分取消掉，而是这两个时间积分同时进行。如果你在定义一个新的 fix 命令时，所使用的 ID 和类型名与某个已经定义的 fix 命令完全相同，那么先前定义的那个 fix 命令就会被删除掉，而新的 fix 命令会被创建，所使用的参数也是重新定义的。这样做就好像是在定义这个新的 fix 命令之前，先使用 unfix 命令取消了旧 fix 命令，除了说它们所作用的顺序会因着定义的

位置不同而有所差异。另外需要注意的是，这种重新定义 fix 命令也会将先前使用 fix_modify 命令定义的，与旧 fix 命令相关的其他任何改变都清除掉。
fix_modify 命令可以用来重置 fix 命令定义的某些设置。具体细节可以参考特定类型的 fix 命令。
有些 fix 命令会在写二进制重启动文件（restart 命令或 write_restart 命令）时保存其内部状态。这些 fix 命令会在重启动该模拟时继续对计算过程起作用。参考 read_restart 命令，了解在读入重启动文件的输入脚本中，如何重新指定 fix 命令。参考具体类型的 fix 命令，了解哪些命令会被写入到重启动文件中。

有些类型的 fix 命令会计算某种类型的量（全局量、单原子量或局部量），这些量可以被其他命令使用或输出。

- 全局量（global）是系统维度的量，比如体系的总能。
- 单原子量（per-atom）是每个原子都具有的量，比如每个原子的位移矢量。不在 fix 命令所约束的组内的原子的该单原子量被设为 0。
- 局域量（local）是每个处理器基于它们所拥有的原子而计算出的量。原子可能有多个或没有局域量。

另外需要注意的是一个单独的 fix 命令只可能会产生这三种类型量的某一种。

全局量、单原子量和局域量都有三种存在形式：单独的标量、一维矢量和二维阵列。每种具体类型的 fix 命令的页面会介绍它会产生哪种类型的变量，以及它的存在形式，比如单原子一维矢量。有些类型的 fix 命令会产生多种存在形式的某种类型的量，比如全局标量和全局矢量。

如果要使用 fix 命令计算出的量，比如使用输出命令将它们输出，可以使用下面的方括号记号来引用它们。其中的 ID 是 fix 命令的 ID。

f_ID	整个标量、整个一维矢量、整个二维阵列
f_ID[I]	一维矢量中的一个元素、二维阵列中的一列
f_ID[I][J]	二维阵列中的一个元素

也就是说，使用使用一个方括号可以将量的维度降低一维，这样矢量就会变成标量，阵列就会变成矢量。使用两个方括号会将维度降低两次，也就是将阵列变为标

量。通过这种方法，使用 fix 标量值作为输入的命令也可以同样处理矢量和阵列的元素。

注意：可以使用 fix 量的命令和变量并不是可以使用任何一种存在形式的量，比如一个命令可能会需要矢量类型的量，而不是标量类型的。这也就意味着在引用 fix 计算的量时不能含糊，而需要注意它的存在形式。这也会在具体介绍这些命令时详细的解释到。

在 LAMMPS 中，fix 命令产生的值有下列的使用方法：

- 全局量可以使用命令 thermo_style custom or fix ave/time 输出，也可以以 equal 类型或 atom 类型的变量进行引用。
- 单原子量可以使用 dump customor fix ave/spatial 命令进行输出，也可以使用 fix ave/atom 命令对时间进行平均，或使用 compute reduce 命令进行降维，或使用 atom-style 类型的变量进行引用。
- 局域量可以使用 compute reduce 命令进行降维，或者使用 fix ave/histo 命令进行直方图化。

fix 命令计算的全局量既可以是内部的，也可以是外部的。“内部”是说其值独立于模拟中的原子数，比如温度。“外部”是说其值的大小与模拟中的原子数有关系，比如总的转动动能。命令 Thermodynamic output 会将外部量的值对体系中的原子数进行规范化，具体看“thermo_modify norm”的设置。但对于内部量，它不会进行规范化。如果使用其他方法对 fix 量进行引用，比如变量 variable，你需要了解它是内部量还是外部量。

每一种类型的 fix 命令都有单独的文档页面来介绍其参数和它具体是做什么的。下面是 lammps 中可用的 fix 命令列表。（译注：fix 命令会经常增加，所以这里列出的很可能并非全部。）

- adapt – change a simulation parameter over time
- addforce – add a force to each atom
- append/atoms – append atoms to a running simulation
- aveforce – add an averaged force to each atom
- ave/atom – compute per-atom time-averaged quantities
- ave/histo – compute/output time-averaged histograms
- ave/spatial – compute/output time-averaged per-atom quantities by layer
- ave/time – compute/output global time-averaged quantities

- bond/break – break bonds on the fly
- bond/create – create bonds on the fly
- bond/swap – Monte Carlo bond swapping
- box/relax – relax box size during energy minimization
- deform – change the simulation box size/shape
- deposit – add new atoms above a surface
- drag – drag atoms towards a defined coordinate
- dt/reset – reset the timestep based on velocity, forces
- efield – impose electric field on system
- enforce2d – zero out z-dimension velocity and force
- evaporate – remove atoms from simulation periodically
- external – callback to an external driver program
- freeze – freeze atoms in a granular simulation
- gravity – add gravity to atoms in a granular simulation
- gcmc – grand canonical insertions/deletions
- heat – add/subtract momentum-conserving heat
- indent – impose force due to an indenter
- langevin – Langevin temperature control
- lineforce – constrain atoms to move in a line
- momentum – zero the linear and/or angular momentum of a group of atoms
- move – move atoms in a prescribed fashion
- msst – multi-scale shock technique (MSST) integration
- neb – nudged elastic band (NEB) spring forces
- nph – constant NPH time integration via Nose/Hoover
- nph/asphere – NPH for aspherical particles
- nph/sphere – NPH for spherical particles
- nphug – constant-stress Hugoniostat integration
- npt – constant NPT time integration via Nose/Hoover
- npt/asphere – NPT for aspherical particles
- npt/sphere – NPT for spherical particles
- nve – constant NVE time integration
- nve/asphere – NVE for aspherical particles
- nve/asphere/noforce – NVE for aspherical particles without forces”
- nve/body – NVE for body particles
- nve/limit – NVE with limited step length
- nve/line – NVE for line segments

- nve/noforce – NVE without forces (v only)
- nve/sphere – NVE for spherical particles
- nve/tri – NVE for triangles
- nvt – constant NVT time integration via Nose/Hoover
- nvt/asphere – NVT for aspherical particles
- nvt/sllod – NVT for NEMD with SLLOD equations
- nvt/sphere – NVT for spherical particles
- orient/fcc – add grain boundary migration force
- planeforce – constrain atoms to move in a plane
- poems – constrain clusters of atoms to move as coupled rigid bodies
- pour – pour new atoms into a granular simulation domain
- press/berendsen – pressure control by Berendsen barostat
- print – print text and variables during a simulation
- property/atom – add customized per-atom values
- reax/bonds – write out ReaxFF bond information recenter – constrain the center-of-mass position of a group of atoms
- restrain – constrain a bond, angle, dihedral
- rigid – constrain one or more clusters of atoms to move as a rigid body with NVE integration
- rigid/nph – constrain one or more clusters of atoms to move as a rigid body with NPH integration
- rigid/npt – constrain one or more clusters of atoms to move as a rigid body with NPT integration
- rigid/nve – constrain one or more clusters of atoms to move as a rigid body with alternate NVE integration
- rigid/nvt – constrain one or more clusters of atoms to move as a rigid body with NVT integration
- rigid – constrain many small clusters of atoms to move as a rigid body with NVE integration
- setforce – set the force on each atom
- shake – SHAKE constraints on bonds and/or angles
- spring – apply harmonic spring force to group of atoms
- spring/rg – spring on radius of gyration of group of atoms
- spring/self – spring from each atom to its origin
- srd – stochastic rotation dynamics (SRD)
- store/force – store force on each atom

- store/state – store attributes for each atom
- temp/berendsen – temperature control by Berendsen thermostat
- temp/rescale – temperature control by velocity rescaling
- thermal/conductivity – 使用 Muller-Plathe 方法计算热导
- tmd – guide a group of atoms to a new configuration
- ttm – two-temperature model for electronic/atomic coupling
- viscosity – Muller-Plathe 动量交换法计算粘度
- viscous – viscous damping for granular simulations
- wall/colloid – Lennard-Jones wall interacting with finite-size particles
- wall/gran – frictional wall(s) for granular simulations
- wall/harmonic – harmonic spring wall
- wall/lj126 – Lennard-Jones 12-6 wall
- wall/lj93 – Lennard-Jones 9-3 wall
- wall/piston – moving reflective piston wall
- wall/reflect – reflecting wall(s)
- wall/region – use region surface as wall
- wall/srd – slip/no-slip wall for SRD particles

另外还有一些用户贡献的 fix 命令也发布在 LAMMPS 程序包中。这些类型的 fix 命令被列在 this page。

也有一些用于加速 CPU 和 GPU 计算速度的 fix 命令发布在 LAMMPS 程序包中。这些类型的 fix 命令被列在 this page。

使用限制

有些类型的 fix 命令被包含在某些特定安装包里。只在安装 LAMMPS 的时候将这些安装包编译进去，它们才是可用的。参考 Making LAMMPS 部分，了解更多关于安装包的信息。如果某种类型的 fix 命令需要某个特定的安装包，会在其单独的介绍页面中介绍。

相关命令

unfix, fix_modify

【LAMMPS 翻译系列】fix_modify 命令

由 www.52souji.net 发表于 2013 年 12 月 5 日 || 951 浏览

fix_modify 命令修改过之前定义过的 fix 命令的一个或多个参数。

内容目录 [隐藏]

使用语法

```
fix_modify fix-ID keyword value ...
```

- fix-ID = 需要修改的 fix 命令的 ID
- 可以添加 1 个或多个关键字
- keyword = *temp* or *press* or *energy*
  - *temp* value = 计算温度的 compute 命令的 ID
  - *press* value = 计算压强的 compute 命令的 ID
  - *energy* value = *yes* or *no*

使用举例

```
fix_modify 3 temp myTemp press myPress

fix_modify 1 energy yes
```

使用介绍

该命令用来修改过之前定义过的 fix 命令的一个或多个参数。只有某些特定类型的 fix 命令支持修改参数。参考具体 fix 命令的页面，了解其是否支持使用 fix_modify 命令进行修改。

关键字 temp 决定 fix 命令如何计算温度。其所指定的 ID 是之前定义过的 compute 命令的 ID，而且必须是可以计算温度类型的 compute 命令。实际上，所有可以计算温度的 fix 命令里面已经默认定义了自己的 compute 方法。所以，如果你使用 fix_modify 命令时，它所定义的计算温度的方法就会覆盖 fix 命令里默认的计算方法。
关键字 press 决定 fix 命令如何计算压强。其所指定的 ID 是之前定义过的 compute 命令的 ID，而且必须是可以计算压强类型的 compute 命令。实际上，所以可以计算压强的 fix 命令里面已经默认定义了自己的 compute 方法。所以，如果你使用 fix_modify 命令时，它所定义的的计算压强的方法就会覆盖 fix 命令里默认的计算方法。

有些对计算体系总势能有贡献的 fix 命令，关键字 energy 会决定在进行热力学信息输出势能时，是否将该贡献值包括在其中。参考 thermo_style 命令，了解势能是如何输出的。另外，该贡献值本身可以通过在 thermo_style custom 命令中使用关键字 f_ID 进行引用输出，其中 ID 是相应 fix 命令的 ID。需要注意的是，如果你在进行能量最小化的过程中，希望 fix 命令产生的能量和力能够成为优化判据的一部分，那么你必须使用该设置。

相关命令

fix, compute temp, compute pressure, thermo_style

默认设置

默认情况下，关键字 temp 指定的 ID 是 fix 命令里默认定义的计算温度的 ID，关键字 press 指定的 ID 是 fix 命令里默认定义的计算压强的 ID，关键字 energy 设置为 no。

【LAMMPS 翻译系列】unfix 命令

由 www.52souji.net 发表于 2013 年 12 月 5 日 || 830 浏览

unfix 命令删除之前使用 fix 命令定义的约束。

内容目录 [隐藏]

使用语法

```
unfix fix-ID
```

- fix-ID = 要删除的 fix 命令的 ID

使用举例

```
unfix 2

unfix lower-boundary
```

使用介绍

删除之前使用 fix 命令定义的约束。它也会删除使用 fix_modify 命令对该约束所进行的修改。

相关命令

fix

【LAMMPS 翻译系列】compute 命令

由 www.52souji.net 发表于 2013 年 12 月 5 日 || 3,049 浏览

compute 命令为一组原子执行某种计算。

内容目录 [隐藏]

使用语法

```
compute ID group-ID style args
```

- ID = compute 命令的 ID
- group-ID = 该 compute 命令所作用的原子组的 ID
- style = compute 类型名（最下有列表）
- args = 特定类型 compute 命令所需要的参数

使用举例

```
compute 1 all temp

compute newtemp flow temp/partial 1 1 0

compute 3 all ke/atom
```

使用介绍

该命令为一组原子执行某种计算。计算出的量是瞬时值，也就是说它们只是原子在当前时间步或迭代步的信息。当然，compute 命令也可以在内部保存体系在之前一个状态的某些信息。

定义 compute 命令的时候并不会执行计算。真正的计算过程是被其他 LAMMPS 命令激活的，比如某些 fix 命令需要计算温度的时候，或这需要产生热力学信息的时候，或者需要 dump 输出到文件中的时候。

compute 命令的 ID 只能包含字母、数字和下划线。

---

compute 命令可以计算出全局量、单原子量或局部量中某一种类型的量。

- 全局量（global）是系统维度的量，比如体系的温度。
- 单原子量（per-atom）是每个原子都具有的量，比如每个原子的动能。不在 compute 命令所指定的组内的原子的该单原子量被设为 0。
- 局域量（local）是每个处理器基于它们所拥有的原子而计算出的量，比如键之间的距离。

计算产生单原子量的 compute 命令在其类型名中含有单词“atom”，比如 ke/atom。计算产生局域量的 compute 命令在其类型名称中含有单词“local”，比如 bond/local。类型名中不包含“atom”或“local”的 compute 命令计算产生的是全局量。

另外需要注意的是一个单独的 compute 命令只可能会产生这三种类型的量的某一种。

全局量、单原子量和局域量都有三种存在形式：单独的标量、一维矢量和二维阵列。每种具体类型的 compute 命令的页面会介绍它会产生哪种类型的变量，以及它的存在形式，比如单原子一维矢量。有些类型的 compute 命令会产生多种存在形式的某种类型的量，比如全局标量和全局矢量。

如果要使用 compute 命令计算出的量，比如使用输出命令将它们输出，可以使用下面的方括号记号来引用它们。其中的 ID 是 compute 命令的 ID。

c_ID	整个标量、整个一维矢量、整个二维阵列

c_ID[I]	一维矢量中的一个元素、二维阵列中的一列
c_ID[I][J]	二维阵列中的一个元素

也就是说，使用使用一个方括号可以将量的维度降低一维，这样矢量就会变成标量，阵列就会变成矢量。使用两个方括号会将维度降低两次，也就是将阵列变为标量。通过这种方法，使用 compute 标量值作为输入的命令也可以同样处理矢量和阵列的元素。

注意：可以使用 compute 量的命令和变量并不是可以使用任何一种存在形式的量，比如一个命令可能会需要矢量类型的量，而不是标量类型的。这也就意味着在引用 compute 计算的量时不能含糊，而需要注意它的存在形式。这也会在具体介绍这些命令时详细的解释到。

在 LAMMPS 中，compute 命令产生的值有下列的使用方法：

- 计算全局温度或全局压强的 compute 命令产生的结果可以被进行恒温或恒压的 fix 命令使用，或者是在创建原子速度的时候被使用。
- 全局量可以使用命令 thermo_style custom or fix ave/time 输出，也可以以 equal 类型或 atom 类型的变量进行引用。
- 单原子量可以使用 dump customor fix ave/spatial 命令进行输出，也可以使用 fix ave/atom 命令对时间进行平均，或使用 compute reduce 命令进行降维，或使用 atom-style 类型的变量进行引用。
- 局域量可以使用 compute reduce 命令进行降维，或者使用 fix ave/histo 命令进行直方图化，或者使用 dump local 命令进行输出。

compute 命令计算的全局量既可以是内部的，也可以是外部的。“内部”是说其值独立于模拟中的原子数，比如温度。“外部”是说其值的大小与模拟中的原子数有关系，比如总的转动动能。命令 Thermodynamic output 会将外部量的值对体系中的原子数进行规范化，具体看“thermo_modify norm”的设置。但对于内部量，它不会进行规范化。如果使用其他方法对 fix 量进行引用，比如变量 variable，你需要了解它是内部量还是外部量。

LAMMPS 会在内部为热力学输出定义默认的 compute 计算，而且总会创建三种计算，分别命名为 “thermo_temp”, “thermo_press” 和 “thermo_pe”。这种默认的定义与在输入脚本中使用了下面的命令效果相同：

```
compute thermo_temp all temp

compute thermo_press all pressure thermo_temp

compute thermo_pe all pe
```

如果热力学输出需要其他类型的量，可以再添加其他相应的 compute 计算。具体可以参考 thermo_style 命令。
可以计算温度或压强的 fix 命令，比如用于恒温或恒压的 fix 命令，在其内部也可以创建 compute 计算。这些在 fix 命令的页面有进行介绍。
上面列举的所有默认定义的 compute 计算，都可以通过在输入脚本中使用 compute 命令定义相应的计算而被替换。这些在 thermo_modify 命令和 fix modify 命令的页面有介绍。
不论是 LAMMPS 内部定义的还是用户自定义的 compute 命令，其设置都可以使用 compute_modify 命令进行修改。
compute 定义的计算可以使用 uncompute 命令删除。

用户也可以自己通过编写代码创建新的 compute 命令，其结果的应用与上面提到的引用方法相同。

每一种类型的 compute 命令都有单独的文档页面来介绍其参数和它具体是做什么的。下面是 lammps 中可用的 compute 命令列表。（译注：compute 命令会经常增加，所以这里列出的很可能并非全部。）

- angle/local – theta and energy of each angle
- atom/molecule – sum per-atom properties for each molecule
- body/local – attributes of body sub-particles
- bond/local – distance and energy of each bond
- centro/atom – centro-symmetry parameter for each atom
- cluster/atom – cluster ID for each atom
- cna/atom – common neighbor analysis (CNA) for each atom
- com – center-of-mass of group of atoms
- com/molecule – center-of-mass for each molecule

- contact/atom – contact count for each spherical particle
- coord/atom – coordination number for each atom
- damage/atom – Peridynamic damage for each atom
- dihedral/local – angle of each dihedral
- displace/atom – displacement of each atom
- erotate/asphere – rotational energy of aspherical particles
- erotate/rigid – rotational energy of rigid bodies
- erotate/sphere – rotational energy of spherical particles
- erotate/sphere/atom – rotational energy for each spherical particle
- event/displace – detect event on atom displacement
- group/group – energy/force between two groups of atoms
- gyration – radius of gyration of group of atoms
- gyration/molecule – radius of gyration for each molecule
- heat/flux – heat flux through a group of atoms
- improper/local – angle of each improper
- inertia/molecule – inertia tensor for each molecule
- ke – translational kinetic energy
- ke/atom – kinetic energy for each atom
- ke/rigid – translational kinetic energy of rigid bodies
- msd – mean-squared displacement of group of atoms
- msd/molecule – mean-squared displacement for each molecule
- pair – values computed by a pair style
- pair/local – distance/energy/force of each pairwise interaction
- pe – potential energy
- pe/atom – potential energy for each atom
- pressure – total pressure and pressure tensor
- property/atom – convert atom attributes to per-atom vectors/arrays
- property/local – convert local attributes to localvectors/arrays
- property/molecule – convert molecule attributes to localvectors/arrays
- rdf – radial distribution function g(r) histogram of group of atoms
- reduce – combine per-atom quantities into a single global value
- reduce/region – same as compute reduce, within a region
- slice – extract values from global vector or array
- stress/atom – stress tensor for each atom
- temp – 计算一组原子的温度
- temp/asphere – temperature of aspherical particles

- temp/com – temperature after subtracting center-of-mass velocity
- temp/deform – temperature excluding box deformation velocity
- temp/partial – temperature excluding one or more dimensions of velocity
- temp/profile – temperature excluding a binned velocity profile
- temp/ramp – temperature excluding ramped velocity component
- temp/region – temperature of a region of atoms
- temp/sphere – temperature of spherical particles
- ti – thermodyanmic integration free energy values
- voronoi/atom – Voronoi volume and neighbors for each atom

另外还有一些用户贡献的 compute 命令也发布在 LAMMPS 程序包中。这些类型的 fix 命令被列在 this page。
也有一些用于加速 CPU 和 GPU 计算速度的 compute 命令发布在 LAMMPS 程序包中。这些类型的 compute 命令被列在 this page。

相关命令

uncompute, compute_modify, fix ave/atom, fix ave/spatial, fix ave/time, fix ave/histo

【LAMMPS 翻译系列】compute_modify 命令

compute_modify 命令用来修改过之前定义过的 compute 命令的一个或多个参数。

内容目录 [隐藏]

使用语法

```
compute_modify compute-ID keyword value ...
```

- compute-ID = 需要修改的 compute 命令的 ID
- 可以添加 1 个或多个关键字
- keyword = *extra* or *dynamic* or *thermo*

- *extra* value = N 要减掉的自由度的数量
- *dynamics* value = *yes* or *no* 计算温度的时候是否重新计算原子数
- *thermo* value = *yes* or *no* 是否考虑 fix 命令计算的势能对总势能的贡献

使用举例

```
compute_modify myTemp extra 0

compute_modify newtemp dynamic yes extra 600
```

使用介绍

该命令用来修改过之前定义过的 compute 命令的一个或多个参数。只有某些特定类型的 compute 命令支持修改参数。

关键字 extra 用来指定在计算温度时，减掉几个自由度作为规范化因子。[原文：The *extra* keyword refers to how many degrees-of-freedom are subtracted (typically from 3N) as a normalizing factor in a temperature computation] 只有那些可以计算温度的 compute 命令可以使用该选项。The default is 2 or 3 for 2d or 3d systems which is a correction factor for an ensemble of velocities with zero total linear momentum.

如果你需要增加自由度，那么你可以将 extra 设置为负值。命令 compute temp/asphere 就是这样的一个例子。

关键字 dynamic 决定在使用 compute 命令计算温度的时候，是否重新计算组内的原子数 N。只有那些可以计算温度的 compute 命令可以使用该选项。默认情况下，N 是一个常量。如果你将一些原子添加到了系统中（比如使用命令 fix pour or fix deposit），或者可能存在原子的丢失（比如由于蒸发），那么这个选项可以确保在计算的温度是规范化的。

关键字 thermo 决定是否将一些 fix 命令计算的势能加入到该 compute 命令计算出的总势能中。目前只有类型名为 pe 的 compute 命令可以使用该选项。

相关命令

compute

【LAMMPS 翻译系列】uncompute 命令

uncompute 命令删除之前使用 compute 命令定义的约束。

内容目录 [隐藏]

使用语法

```
uncompute compute-ID
```

- compute-ID = 要删除的 compute 命令的 ID

使用举例

```
uncompute 2

uncompute lower-boundary
```

使用介绍

删除之前使用 compute 命令定义的计算。它也会删除使用 compute_modify 命令对该计算所进行的修改。

相关命令

compute

【LAMMPS 翻译系列】restart 命令

由 www.52souji.net 发表于 2013 年 10 月 22 日 || 1,265 浏览

restart 命令在计算的时候，按着特定的模式，以一定的时间步为周期，写二进制的重启动文件。

内容目录 [隐藏]

使用语法

```
restart 0

restart N file

restart N file1 file2
```

- N：每 N 步写入重启动文件，N 可以是一个变量。
- file：[单文件模式]重启动文件的文件名。
- file1/file2：[双文件模式]重启动文件的文件名，交替写重启动文件。

使用举例

```
restart 0

restart 1000 poly.restart

restart 1000 restart.*.equil

restart 10000 poly.%.1 poly.%.2

restart v_mystep poly.restart
```

使用介绍

该命令在计算的时候，按着特定的模式（单文件模式和双文件模式），以一定的时间步为周期，写二进制的重启动文件。如果参数是 0，那么就不会写重启动文件。该命令的两种模式是：

- 单文件模式：如果指定了 1 个文件名，那么程序会创建一系列文件名中包含时间步的重启动文件。
- 双文件模式：如果指定了 2 个文件名，那么程序只会以这两个指定的文件名创建两个重启动文件，并在这两个重启动文件间来回写入重启动信息。

注意：在一个输入脚本中，你可以两次使用 restart 命令，一次是单文件，一次双文件。举例来说，你可以以 100000 步的频率写单文件的重启动文件用来存档，同时你也可以以 1000 步的频率写双文件的重启动文件，将它们作为临时的重启动文件。使用 restart 0 可以关掉这两种重启动的输出。

与 dump 命令输出文件类似，restart 命令在输出重启动文件时，也可以包括两种通配符。

- 如果单文件模式的文件名中包括符号“*”，那么该符号会被当前的时间步值替换掉。因此，对于上面例子中的第三个，会创建的重启动文件为：restart.1000.equil, restart.2000.equil, 等。如果单文件模式的文件中不包括“*”，那么时间步值会自动添加在文件名的最后。上面例子中的第二个会创建的重启动文件为：poly.restart.1000, poly.restart.2000, 等。
- 如果重启动文件的文件名中包括符合“%”，那么每个处理器会写一个文件，并且符号“%”会被处理器的 ID（从 0 到 P-1）替换掉。另外，还会有一个文件，文件名是用“base”代替“%”，其中包含了全局的信息。举例来说，如果使用通配符%，并以 1000 时间步的频率写重启动文件，那么会写出的文件有 restart.base.1000，restart.0.1000，restart.1.1000，.....，resta.P-1.1000。使用这种通配符，会创建更小的文件，对于在并行机器中输出和后续的输入都是一种更快的方式。

使用该命令写重启动文件的时间是在时间步是 N 的倍数时，而不是在一个 run 或能量最小化的第一个时间步。如果你希望在一个 run 开始前写重启动文件，那么你可以使用 write_restart 命令。使用 restart 命令也不会在一个 run 的最后一个时间步写重启动文件，除非这最后一个时间步恰好是 N 的倍数。但对于能量最小化过程而言，如果其结果收敛的，并且 N 设置>0，那么即便使用 restart 命令，其最后一个时间步也会写入重启动文件。

N 除了可以是一个数值之外，也可以被指定为 equal 样式的变量，但需要以 v_name 的形式引用（其中的 name 就是变量名）。在这种情形下，变量会在开始一个 run 之前被计算，从而确定下一个写出重启动文件的时间步。在到了那个时间步时，变量会再次被计算以决定下一个输出的时间步。以此类推。因此变量需要返回的时间步的值。函数 stagger(), logfreq(), stride()是与此相关数学变量函数，其他类似的数学函数也可以作为选项被添加。

举个例子，下面的命令会从时间步 1100 到 1200 写重启动文件，这对于调试一个在 1163 步出错的模拟来说是非常实用的。

```
variable   s equal stride(1100,1200,1)
restart                v_s tmp.restart
```

参考命令 read_restart，了解哪些信息被存储在了重启动文件中。
重启动文件可以被命令 read_restart 读入，从而可以从设定的状态重新开始一个新的模拟过程。因为重启动文件是二进制的，因此在其他的机器上可能就不可读（二进制文件是依赖于机器的）。在这种情况下，可以使用 tools 目录下提供的

restart2data 程序将重启动文件转换为文本式的数据文件格式。命令 read_restart 和工具 restart2data 都可以读入由通配符%指定而输出的文件。

相关命令

write_restart, read_restart

默认设置

```
restart 0
```

【LAMMPS 翻译系列】thermo 命令

由 www.52souji.net 发表于 2013 年 10 月 24 日 || 1,051 浏览

thermo 命令用来设置在模拟中计算和打印热力学信息（比如温度、能量、压强）的时间步的频率。

内容目录 [隐藏]

使用语法

```
thermo N
```

- N：输出热力学信息的频率，可以是变量。

使用举例

```
thermo 100
```

使用介绍

该命令用来设置在模拟中计算和打印热力学信息（比如温度、能量、压强）的时间步的频率。热力学信息会在时间步为 N 的倍数以及模拟的开始和最后的时候打印。

被打印信息的内容和格式是由命令 thermo_style 和 thermo_modify 控制的。

N 除了可以是一个数值之外，也可以被指定为 equal 类型的变量，但需要以 v_name 的形式引用，其中的 name 就是变量名。在这种情形下，变量会在开始一个 run 之前被计算，从而确定下一个热力学信息输出的时间步。在到了那个时间步时，变量会被再次计算以决定下一个输出的时间步。以此类推。因此变量需要返回时间步的值。函数 stagger(), logfreq(), stride()是与此相关数学变量函数，其他类似的数学函数也可以作为选项被添加。

举个例子，下面的命令可以保证在时间步 0,10,20,30,100,200,300,1000,2000 等的时候输出热力学信息。

```
variable   s equal logfreq(10,3,10)

thermo          v_s
```

相关命令

thermo_style, thermo_modify

默认设置

```
thermo 0
```

【LAMMPS 翻译系列】write_restart 命令

由 www.52souji.net 发表于 2013 年 10 月 21 日 || 1,004 浏览

write_restart 命令将模拟的当前状态写入到二进制的重启动文件中。

内容目录 [隐藏]

使用语法

```
write_restart file
```

- file：要写入重启动信息的文件的文件名

使用举例

```
write_restart restart.equil

write_restart poly.%.*
```

使用介绍

该命令将模拟的当前状态写入到二进制的重启动文件中。

在一个耗时很长的模拟中，命令 restart 可以按着一定的周期写重启动文件，而命令 write_restart 可以在完成能量最小化之后，或者按着你的需要随时写出当前状态的重启动文件。
与命令 dump 类似，该命令的参数 file 中可以包括两个通配符。如果使用了“*”符号，该符号会被当前时间步的值代替；如果使用了“%”符号，那么每个处理器都会写一个文件，并且%符号会被处理器的 ID 代替（从 0 到 P-1）。另外，还会有一个文件，文件名是用“base”代替“%”，其中包含了全局的信息。举例来说，如果使用通配符%，那么会写出的文件有 restart.base，restart.0，restart.1，.....，resta.P-1。使用这种通配符，会创建更小的文件，对于在并行机器中输出和后续的输入都是一种更快的方式。
写出的重启动文件可以使用命令 read_restart 读入，从而可以从一个特别的状态重新开始一个模拟。因为重启动文件是二进制的，因此在其他的机器上可能就不可读（二进制文件是依赖于机器的）。在这种情况下，可以使用 tools 目录下提供的 restart2data 程序将重启动文件转换为文本式的 data file 格式。命令 read_restart 和工具 restart2data 都可以读入由通配符%指定而输出的文件。
注意：尽管重启动文件的目的是要从写入重启动文件的位置开始一个模拟，但对一个模拟而言，并不是所有的信息都会存储在重启动文件中。举例来说，原输入脚本中指定的 fix 命令不会存储在重启动文件中，这就要求你在新的输入脚本中需要重新指定你需要用到的 fix 命令。即便有些重启动信息存储在了文件中，就像某些 fix 命令，你仍然需要在新的输入脚本中进行定义，从而可以重新使用这些信息。参考命令 read_restart 了解更多关于存储在重启动文件中的信息。

使用限制

该命令要求在写入重启动文件之前进行处理器间的原子迁移。这就意味着在使用这个命令之前，你的系统需要准备好要进行模拟（即完成了力场设置、原子质量初始化等等）。

相关命令

restart, read_restart, write_data

【LAMMPS 翻译系列】delete_atoms 命令

delete_atoms 命令用于删除指定的原子。

内容目录 [隐藏]

使用语法

```
delete_atoms style args keyword value ...
```

- style = *group* or *region* or *overlap* or *porosity*

  - *group* args = group-ID
  - *region* args = region-ID
  - *overlap* args = cutoff group1-ID group2-ID
    - cutoff = delete one atom from pairs of atoms within the cutoff (distance units)
    - group1-ID = one atom in pair must be in this group
    - group2-ID = other atom in pair must be in this group
  - *porosity* args = region-ID fraction seed
    - region-ID = region within which to perform deletions
    - fraction = delete this fraction of atoms

seed = random number seed (positive integer)

- keyword = *compress* or *mol*

  - *compress* value = *no* or *yes*
  - *mol* value = *no* or *yes*

使用举例

```
delete_atoms group edge

delete_atoms region sphere compress no

delete_atoms overlap 0.3 all all

delete_atoms overlap 0.5 solvent colloid

delete_atoms porosity cube 0.1 482793
```

使用介绍

该命令用于删除指定的原子。具体来说，你既可以使用该命令在块体材料中创建空洞（译注：比如空位缺陷等等），也可以删除彼此相距较近的原子（比如晶界附近的原子）。

group 类型：所有属于该组的原子会被删掉。
region 类型：所有在该区域中的原子会被删掉。如果某分子中有原子在该指定区域，那么属于这个分子的所有原子都会被删掉。参考下面对 mol 关键字的介绍。
overlap 类型：在指定的组中查找距离小于指定截断距离的原子对，将其中一个原子删掉。原子对中的第一个原子应属于第一个指定的组，第二个原子属于第二个指定的组。被删掉的那个原子是第一个组中的。

两个组使用相同的 ID（比如使用 all），或者某些原子同时属于这两个组（译注：两个组在定义上有交集）也是可以的。在这种情况下，这对原子中的任何一个都有可能被删除。对于这种特殊情形，可以确保的是在删除操作结束后，所有原子间的距离都满足要求，即不小于截断指定的截断距离，但不能确保删除的原子数目是最小的，或者在使用不同数量的处理器时，删除相同的原子。

porosity 类型：删除指定区域内指定比例的原子。举例来说，如果 fraction 设置为 0.1，那么就会删除 10%的原子。删除的原子是随机选取的。程序不能保证绝对精确的删除比例，也不能保证在运行不同数量的处理器时，删除相同的原子。

compress 关键字：如果设置为 yes，那么指定原子被删除之后，剩下原子的 ID 会从 1 开始重新编号。对于分子系统，可以忽略该关键字的作用，因为重新对 ID 进行编号会搞乱原来原子间键的连接。

mol 关键字：如果设置为 yes，那么属于该分子的任何一个原子被删掉了，所有属于该分子的原子也会被删掉。该关键字只对 region 类型有效。这样做可以确保整个分子都被删掉，而不是一部分原子，否则就会很容易因为剩下原子的键、角、二面角的相互作用而造成运行出错。

使用限制

overlap 类型需要处理器间通信以获取镜像原子来建立邻域列表，这就意味着你在使用该命令之前，已经完成了模拟所需的其他命令的设置（如力场、原子质量等）。又因为查找重叠的原子需要用到领域列表，所以你必须定义 pair_style，并且其中所指定的截断距离要大于或等于 overlap 类型中指定的截断距离。

如果定义了 special_bonds，并且参数都设为 0，那么成键的原子对 (1-2, 1-3, or 1-4) 就不会出现在邻域列表中，因此也就不会被指定为 overlap 类型的命令删除。这样考虑是合理的，因为你大概不会希望将成键的原子对中的一个原子。

相关命令

create_atoms

默认设置

compress=yes, mol=no

【LAMMPS 翻译系列】minimize 命令

由 www.52souji.net 发表于 2013 年 12 月 5 日 || 2,828 浏览

minimize 命令通过不断迭代调整原子坐标的方式对体系的能量进行最小化。

内容目录 [隐藏]

使用语法

```
minimize etol ftol maxiter maxeval
```

- etol = 能量的停止容差（无单位）
- ftol = 力的停止容差（力的单位）
- maxiter = 能量最小化器 minimizer 的最大迭代次数
- maxeval = 计算力或能量的最大次数

使用举例

```
minimize 1.0e-4 1.0e-6 100 1000

minimize 0.0 1.0e-8 1000 100000
```

使用介绍

该命令通过不断迭代调整原子坐标的方式对体系的能量进行最小化。当满足任何一个停止判据（能量或力）时，迭代过程就会终止。迭代结束时刻的构型在很大程度上说就是局域势能最小的构型。更精确地说，这个时候的构型接近目标函数（下面会介绍）的临界点，它有可能是，也有可能不是局域最小值。

优化过程所使用的优化算法是由 min_style 命令进行设置的。其他的选项可以使用 min_modify 命令进行设置。minimize 命令可以与 run 命令交替使用，从而可以交替着进行弛豫和动力学。能量最小化的过程限制了原子在一个迭代步中移动的距离，使用动力学你就可以将高度重叠的原子（具有很大的能量和力）相互推开。还有一种对体系进行弛豫的方法是使用较小的或受限的时间步，对体系运行动力学。或者在运行动力学的时候使用 fix viscous 命令施加阻尼力，慢慢地拖动体系的总动能。在运行动力学的时候，可以使用 pair_style soft 势函数分开重叠的原子。

能量最小化算法 cg、sd、hftn 先在外层迭代中确定坐标改变的搜索方向。然后基于线搜索（line search）算法进行内层迭代。线搜索算法通过不断的计算力和能量来确定新的坐标。目前的程序使用的是一种回溯算法，就计算力的次数来说可能不是最优的算法，但经过测试确是最稳定的算法。Nocedal and Wright's Numerical Optimization 对回溯算法进行了介绍(Procedure 3.1 on p 41)。

能量最小化算法 quickmin 和 fire 是使用欧拉积分步运行带阻尼的分子动力学。因此，需要定义时间步。虽然使用更大的时间步执行效率会更高一些，但一般使用与 running dynamics 相同的时间步。

能量最小化的过程实际上是对下面的目标函数进行最小化的过程。该目标函数是体系的总势能，它是体系中 N 个原子坐标的函数。

$$E(r_1, r_2, \ldots, r_N) = \sum_{i,j} E_{pair}(r_i, r_j) + \sum_{ij} E_{bond}(r_i, r_j) + \sum_{ijk} E_{angle}(r_i, r_j, r_k) + \sum_{ijkl} E_{dihedral}(r_i, r_j, r_k, r_l) + \sum_{ijkl} E_{improper}(r_i, r_j, r_k, r_l) + \sum_{i} E_{fix}(r_i)$$

其中第一项是所有无键作用的对势相互作用的总和，也包括长程库伦相互作用。第二项到第五项分别代表键（bond）、角（angle）、二面角（dihedral）和 improper 的相互作用。最后一项考虑了某些 fix 命令通过约束或在原子上施加力而带来的作用，比如通过与壁面进行作用。参考 fix 命令，了解它是如何影响能量最小化的。

能量最小化的起始点是开始进行能量最小化之时的原子构型。

如果下面任何一个判据满足了，能量最小化过程就会停止。

- 外层迭代的能量改变值小于 etol
- 全局力矢量的长度小于 ftol
- the 2-norm (length) of the global force vector is less than the *ftol*
- 由于步长返回 0.0 而导致线性搜索失败
- 外层迭代次数或时间步超过了 *maxiter*
- 计算力的次数超过了 *maxeval*

对于第一个判据，指定的能量容差 etol 是没有单位的。当两个相邻的迭代步的能量差除以总的能量值小于或等于该容差时，这个判据就满足了。举个例子，如果将 etol 设置为 1.0e-4，就是说能量容差等于总能的 1/10000。使用带阻尼的动力学进行能量最小化的时候，这个判据只有在速度被重置为 0 之后的一些步才会被检查，否则会使能量最小化过程较早的收敛。

对于第二个判据，指定的力容差 ftol 是力的单位，因为它是所有原子的全局力矢量的长度。对于 N 个原子而言，全局力矢量是一个维度为 3N 的矢量。由于在能量最小化之后，力矢量的很多元素都接近 0，你可以认为是 ftol 是任何原子任何方向

上受力的上边界（译注：上边界即最高要求，所有要求都达到才可以；下边界即最低要求，只要达到一个就行）举个例子，如果将 ftol 设置为 1.0e-4，就是说通过能量最小化之后，没有任何原子的在 x，y，z 方向上受力大约 1.0e-4（力的单位）。

etol 和 ftol 都可以同时或单独设置为 0.0。在这种情况下，其他的判据会用来结束能量最小化过程。

在能量最小化的过程中，最外层的迭代次数被作为时间步处理。实际的输出也是以它为参考的，比如热力学输出或 dump 输出或重启动文件输出。

使用 thermo_style custom 命令中的关键字 fmax 或 fnorm，可以有效的监控能量最小化的过程。Note that these outputs will be calculated only from forces on the atoms, and will not include any extra degrees of freedom, such as from the fix box/relax command.

在能量最小化结束后，程序会打印一段统计摘要信息，介绍满足了何种收敛判据，以及能量、受力、最终的线性搜索和跌打次数等。下面是一个例子：

```
Minimization stats:
  Stopping criterion = max iterations
  Energy initial, next-to-last, final =
      -0.626828169302     -2.82642039062     -2.82643549739
  Force two-norm initial, final = 2052.1 91.9642
  Force max component initial, final = 346.048 9.78056
  Final line search alpha, max atom move = 2.23899e-06 2.18986e-05
  Iterations, force evaluations = 2000 12724
```

- 其中的三个能量值分别是进行能量最小化之前的能量、能量最小化结束之前一步的能量、能量最小化结束时的能量。这就是判据参数 etol 所要检查比较的。
- 其中的“Force max component”是指能量最小化之前和之后的全局力矢量的长度。这是判据参数 ftol 说要检查比较的。
- 其中的“Force max component”是指全局力矢量中最大元素的绝对值。
- 其中的“Final line search alpha”，即线性搜索参数 alpha，乘以最后迭代步时的最大力元素（即力矢量中的最大值），就是最后迭代步中原子最大移动距离（max atom move）。如果线搜索不能使能量降低，alpha 就会是 0.0。即使 alpha 不是 0.0，但

如果原子最大移动距离相比于原子坐标太小，就会导致最后一步原子几乎没有移动，进而能量改变很微小，而使得能量最小化过程结束。换句话说，即使受力不为0，但如果这些力太小了，就可能使原子发生移动的距离小于机器精度，从而就不能再使能量更小了。

判据参数 *maxiter* 和 *maxeval* 会分别检查比较迭代次数和力的计算次数。

注意：在 LAMMPS 中有些力场（势函数）会因为不连续或其他近似，使得在进行能量最小化的时候不能使用精度较高的容差。举个例子，在进行能量最小化的是会，你需要使用那些在截断距离过零点的势类型，即使你后面又切换成运行分子动力学。如果你不这样做，但任何一对原子之间的距离从+epsilon 截断变成-epsilon 时，系统的总能会出现不连续，从而导致能量最小化过程表现不稳定[原文： the total energy of the system will have discontinuities when the relative distance between any pair of atoms changes from cutoff+epsilon to cutoff-epsilon and the minimizer may behave poorly]。 Some of the manybody potentials use splines and other internal cutoffs that inherently have this problem. The long-range Coulombic styles (PPPM, Ewald) are approximate to within the user-specified tolerance, which means their energy and forces may not agree to a higher precision than the Kspace-specified tolerance. In all these cases, the minimizer may give up and stop before finding a minimum to the specified energy or force tolerance.

Note that a cutoff Lennard-Jones potential (and others) can be shifted so that its energy is 0.0 at the cutoff via the pair_modify command. See the doc pages for inidividual pair styles for details. Note that Coulombic potentials always have a cutoff, unless versions with a long-range component are used (e.g. pair_style lj/cut/coul/long). The CHARMM potentials go to 0.0 at the cutoff (e.g.pair_style lj/charmm/coul/charmm), as do the GROMACS potentials (e.g. pair_style lj/gromacs).

If a soft potential (pair_style soft) is used the Astop value is used for the prefactor (no time dependence).

The fix box/relax command can be used to apply an external pressure to the simulation box and allow it to shrink/expand during the minimization.

只有一小部分 fix 命令（一般是用来施加力约束的）会在能量最小化的过程中被激活。具体类型的 fix 命令的页面会告诉你它是否会属于这类命令。

注意：有些 fix 命令会与能量最小化过程中被激活，计算一部分势能。如果要把这部分能量包括到体系的总势能中，必须先把 fix_modify 命令中的 energy 选项打开。细节可以参考具体类型的 fix 页面。

使用限制

这里列出了还没有在程序中实现的功能，看看你是否知道如何实现它们：

在能量最小化的时候使用 fix shake 命令会出现错误，因为该命令关闭键的作用，而这些作用却被包括在了体系的势能中。 The effect of a fix shake can be approximated during a minimization by using stiff spring constants for the bonds and/or angles that would normally be constrained by the SHAKE algorithm.
能量最小化的过程也不支持 Fix rigid 命令。虽然定义了这个命令并不会出现错误，但能量最小化的时候却不能使定义为刚体的部分保持为刚体。需要注意，如果刚体内部的键、角被关闭了（比如通过命令 neigh_modify exclude），它们就不会再对总势能有贡献了，这可能跟想象的不一样。

Pair potentials that produce torque on a particle (e.g. granular potentials or the GayBerne potential for ellipsoidal particles) are not relaxed by a minimization. More specifically, radial relaxations are induced, but no rotations are induced by a minimization, so such a system will not fully relax.

相关命令

min_modify, min_style, run_style

【LAMMPS 翻译系列】clear 命令

由 www.52souji.net 发表于 2013 年 10 月 7 日 || 781 浏览

clear 命令用来删除所有的原子、将所有的设置都设为默认值，并释放 LAMMPS 分配的所有内存。

内容目录 [隐藏]

使用语法

```
clear
```

使用举例

```
(commands for 1st simulation)

clear

(commands for 2nd simulation)
```

使用介绍

该命令用来删除所有的原子、将所有的设置都设为默认值，并释放 LAMMPS 分配的所有内存。一旦运行了命令 clear，看起来的效果就像 LAMMPS 重新启动了，但也下面也会提到一些例外。该命令可以让你在一个输入脚本中顺序运行多个作业。

这里列出的设置对命令 clear 来说是例外：工作目录（命令 shell）、日志文件状态（命令 log）、echo 状态（命令 echo）以及输入脚本中的变量（命令 variable）。

- **LAMMPS 命令官方手册**
- **LAMMPS 中文翻译全部命令索引**

【LAMMPS 翻译系列】echo 命令

由 www.52souji.net 发表于 2013 年 10 月 4 日 || 817 浏览

echo 命令决定在 LAMMPS 读入并处理输入脚本中的命令时，是否将它们输出到屏幕和/或日志文件中。

内容目录 [隐藏]

使用语法

```
echo style
```

- style： *none* or *screen* or *log* or *both*

使用举例

```
echo both

echo log
```

使用介绍

该命令决定在 LAMMPS 读入并处理输入脚本中的命令时，是否将它们输出到屏幕和/或日志文件中。如果你的输入脚本有错误，使用这个命令可以让你看到最后一个被处理的命令是什么。

命令行参数-echo 的作用与该命令相同，可以代替该命令。

默认设置

```
echo log
```

【LAMMPS 翻译系列】include 命令

由 www.52souji.net 发表于 2013 年 10 月 11 日 || 821 浏览

include 命令用来打开一个新的输入脚本文件，并开始从这个新文件中执行 LAMMPS 命令。

内容目录 [隐藏]

使用语法

```
include file
```

- file： 要跳转到的输入脚本的文件名。

使用举例

```
include newfile

include in.run2
```

使用介绍

该命令用来打开一个新的输入脚本文件，并开始从这个新文件中执行 LAMMPS 命令。在这个新的输入脚本被执行完了之后，再返回到原脚本（包括该命令的）继续执行。你可以根据自己的需要任意嵌套多层。但如果在你的输入脚本 A 里使用命令 include 包括了脚本 B，又在 B 中使用命令 include 包括了 A，那么 LAMMPS 可能会进入死循环。

如果使用变量（参考命令 variable）定义文件名，可以实现不同的处理器分区运行不同的输入脚本。

相关命令

variable, jump

【LAMMPS 翻译系列】jump 命令

jump 命令关闭当前输入脚本文件，打开命令中文件名所指定的文件，并从那个文件开始读入 LAMMPS 命令。

内容目录 [隐藏]

使用语法

```
jump file label
```

- file： 要跳转到的输入脚本的文件名

- label：[可选]要跳转到的输入脚本中的标签

使用举例

```
jump newfile

jump in.run2 runloop

jump SELF runloop
```

使用介绍

该命令关闭当前输入脚本文件，打开命令中文件名所指定的文件，并从那个文件开始读入 LAMMPS 命令。与命令 include 不同的是，LAMMPS 不会再返回到之前的输入文件了。当然，你也可以在这些输入脚本中使用多个 jump 命令，让 LAMMPS 再返回到之前的输入脚本。

如果文件名被指定为 SELF，那么 LAMMPS 会重新打开当前的输入脚本并读入命令。

注意：如果使用标准输入读入当前输入脚本文件，SELF 选项不一定会起作用。标准输入即是指使用重定向符号，如下所示：

```
lmp_g++ < in.script
```

因为 SELF 选项会调用 C 库函数 rewind()，而在某些系统中，标准输入并不支持这个。但还有两位两种方法可以让 SELF 选项起作用：一个是使用命令行参数-in 读入输入脚本文件；另一个是使用命令行参数-var 读入输入脚本文件，如下所示：

```
lmp_g++ -in in.script


lmp_g++ -var fname n.script < in.script
```

命令 jump 中的第二个参数是可选的。如果指定了，就相当于在输入脚本中定义了一个标签，LAMMPS 会扫描跳转到的输入文件（不会执行），从标签的位置开始向下执行。这种方式可以用来执行输入脚本的一部分。在下面的例子中，这些命令会被执行 10 次，每次会运行 10000 个时间步，并创建 10 个 dump 文件。在程序

运行了 10 次之后，变量 a 取了其定义中的最后一个值，命令 next 就会使脚本跳出循环，而不会再执行命令 jump。

```
variable a loop 10
label loop
dump 1 all atom 100 file.$a
run 10000
undump 1
next a
jump in.lj loop
```

如果参数 file 被指定为一个变量，命令 jump 可以跳转到不同的处理器分区而执行不同的输入脚本。下面的例子中，LAMMPS 在 40 个处理器上运行，分为 4 个分区，每个分区有 10 个处理器。其中的命令 jump 中使用了变量，这样做就会让不同的分区跳到不同的输入文件。

```
mpirun -np 40 lmp_ibm -partition 4x10 -in in.file


variable f world script.1 script.2 script.3 script.4
jump $f
```

下面的例子是使用命令 if 和 jump 实现的双重循环。if 条件满足时，跳出内层循环，开始执行外层循环。

```
label           loopa
variable    a loop 5
 label          loopb
 variable  b loop 5
 print          "A,B = $a,$b"
 run       10000
```

```
  if            $b > 2 then "jump in.script break"
  next          b
  jump          in.script loopb
label           break
variable    b delete


next          a
jump          in.script loopa
```

使用限制

如果命令 jump 跳转到文件中不包括指定的标签，LAMMPS 会自动跳到最后并退出。

相关命令

variable, include, label, next

【LAMMPS 翻译系列】label 命令

由 www.52souji.net 发表于 2013 年 10 月 9 日 || 867 浏览

label 命令用来将该行的字符串指定为一个 ID，供 jump 命令使用。

内容目录 [隐藏]

使用语法

```
label ID
```

- ID： 代表标签名字的字符串

使用举例

```
label xyz

label loop
```

使用介绍

该命令用来将该行的字符串指定为一个 ID，供命令 jump 使用。除非在该命名之前使用了命令 jump，否则该命令不会带来任何影响。但如果使用了命令 jump，并且带有指向当前标签的参数，那么当前输入脚本中该命令之前的所有命令都会被忽略，而从该命令之后开始执行。该命令对于执行输入脚本中的一部分会非常有用，可以参考命令 jump。

【LAMMPS 翻译系列】log 命令

由 www.52souji.net 发表于 2013 年 10 月 6 日 || 997 浏览

log 命令关闭 LAMMPS 当前的日志文件，打开指定的文件，并在其中写入日志信息。

内容目录 [隐藏]

使用语法

```
log file
```

- file： 日志文件的文件名

使用举例

```
log log.equil
```

使用介绍

该命令关闭 LAMMPS 当前的日志文件，打开指定的文件，并在其中写入日志信息。如果将文件名指定为 none，那么就不会写日志文件。

如果使用多个处理器分区，文件名需要是一个变量，这样不同的处理器才不会将日志写到同样一个日志文件中。

对于 LAMMPS 来说，默认的日志文件名是 log.lammps。也可以通过命令行参数-log 来指定日志文件。

默认设置

```
log log.lammps
```

【LAMMPS 翻译系列】next 命令

label 命令用来将该行的字符串指定为一个 ID，供 jump 命令使用。

内容目录 [隐藏]

使用语法

```
next variables
```

- variables： 一个或者多个变量名

使用举例

```
next x

next a t x myTemp
```

使用介绍

该命令需要与使用命令 variable 定义的变量同时使用，用来从为该变量定义的一些列值中将下一个值赋给变量。在使用该命令之后，变量就是一个新值。
参考变量 variable，了解如何在 LAMMPS 输入脚本中定义和使用不同类型的变量。如果变量名字仅仅是一个小写字母，那么在输入脚本中使用该变量的时候可以仅仅在其前面加上符号“$”，例如$a, $z。但如果是多个字母，那么就需要用花括号将变量名括起来，例如${myTemp}。
如果在 next 命令中使用多个变量作为参数，那么所有的变量都必须具有同样的变量类型，即都必须是 *index*, *loop*, *file*, *universe*, or *uloop* 中的一种。一个例外是 universe 类型和 uloop 类型的变量是可以在 next 命令中混合使用的。
在 next 命令中指定的变量在每次运行 next 命令时会从它们自己的变量列表中取下一个值。file 类型的变量会从相关的文件中读入下一行的值。atomfile 类型的变量会从相关的文件中读入下一个系列行（一行一个原子）。*String*- or *atom*- or *equal*- or *world*- 类型的变量不能在命令 next 中使用，因为它们只存有一个值。
当 next 命令中任何一个变量没有值可取时，程序就会自动设置一个标记，使得输入脚本会跳过不执行下一个 jump 命令，这也同时退出了该 next 命令。用完的变量会被删除，这在命令 variable 中有介绍。你可以在输入脚本的后面继续使用或定义该变量。对于 file 类型和 atomfile 类型的变量，当读到文件的最后时，它们就会被认为已经用完了。

- 如果在 next 命令中使用 index 类型或 loop 类型的变量，那么变量的下一个值会分配给所有的处理器。
- 如果在 next 命令中使用 file 类型的变量，相关文件中读入的下一行会被分配给变量。
- 如果在 next 命令中使用 atomfile 类型的变量，相关文件中读入的下一系列行（每行一个原子）会分配给变量。
- 如果在 next 命令中使用 universe 类型或 uloop 类型的变量时，那么变量的下一个值会分配给首先执行该命令的处理器分区。同一个分区中的所有处理器都会分配同样的值。关于如何在多个处理器分区上运行 LAMMPS，在命令行参数-partition 中有介绍。这两种类型的变量在调整到下一个值的时候，会用到文件”tmp.lammps.variable” 和”tmp.lammps.variable.lock”。在 LAMMPS 运行的时候，你可以在相应目录下看到这两个文件。

下面是一个在 next 命令中使用 index 类型的变量实现运行一系列模拟的例子。如果输入脚本的名字是 in.polymer，那么总共会运行 8 个模拟，并且每个模拟会从不同的目录（从 run1 到 run8）中读取数据文件。

```
variable d index run1 run2 run3 run4 run5 run6 run7 run8
shell cd $d
read_data data.polymer
run 10000
shell cd ..
clear
next d
jump in.polymer
```

如果将上面的变量 d 定义为 universe 类型，并在处理器的三个分区上运行该输入脚本，那么最开始的 3 个模拟（run1,run2,run3）会分别在这三个分区上运行。那个分区上的任务先算完，那么变量 d 中的第 4 个模拟就会在那个分区上运行。以此类推直到所有 8 个模拟都完成。

命令 jump 和 next 嵌套使用可以实现多层循环。举个例子，下面的脚本中包括双重循环，总共可以运行 15 次模拟。

```
variable i loop 3
  variable j loop 5
  clear
  ...
  read_data data.polymer.$i$j
  print Running simulation $i.$j
  run 10000
  next j
  jump in.script
next i
jump in.script
```

下面的例子是使用命令 if 和 jump 实现的双重循环。if 条件满足时，跳出内层循环，开始执行外层循环。

```
label           loopa
variable    a loop 5
 label          loopb
 variable  b loop 5
 print          "A,B = $a,$b"
 run       10000
 if             $b > 2 then "jump in.script break"
 next           b
 jump           in.script loopb
label           break
variable    b delete


next            a
jump            in.script loopa
```

相关命令

jump, include, shell, variable

【LAMMPS 翻译系列】print 命令

由 www.52souji.net 发表于 2013 年 10 月 5 日 || 987 浏览

print 命令可以打印文本字符串到屏幕和日志文件中。

内容目录 [隐藏]

- 相关命令
- 默认设置

使用语法

```
print string keyword value
```

- string： 要打印的文本字符串，可以包括变量
- 可以不添加或添加多个 keyword/value
- keyword：*file* or *append* or *screen*
  - file：value = filename，输出到文件，如果文件存在将被覆盖。
  - append：value = filename，追加到文件。
  - screen：value = yes or no，是否输出到屏幕和日志文件中。

使用举例

```
print "Done with equilibration" file info.dat

print Vol=$v append info.dat screen no

print "The system volume is now $v"

print 'The system volume is now $v'
```

使用介绍

该命令可以打印文本字符串到屏幕和日志文件中。每次会输出一行文本。字符串必须是单一的参数，所以如果字符串中单词的数量超过 1 个，就需要用引号括起来。如果包含变量，那么这些变量会被计算，并输出当前的值。

如果使用了 file 或 append 关键字，就需要指定一个文件名，字符串就会输出到这个指定的文件中。如果使用 file 关键字，文件名指定的文件的内容会被覆盖；如果使用 append 关键字，文件名指定的文件如果不存在就会创建，如果存在就会在原来的内容后面追加。

screen 关键字用来控制是否将字符串输出到屏幕和日志文件中。

如果你希望 print 命令被多次执行，有三个方法。

1. 使用命令 fix print，它可以用来将一个字符串以一定的周期输出。
2. 将 print 命令作为 run 命令中 every 选项的一个参数使用。

3. 在循环体（参考命令 jump 和 next）中使用 print 命令。

参考命令 variable，了解 equal 样式的变量。这种类型的变量可以说是 print 命令中最有用的变量，它可以计算包括数学运算符、原子性质、组性质、热力学性质、由命令 compute 和 fix 计算的全局量，以及其他变量的引用的公式。

相关命令

fix print, variable

默认设置

```
log log.lammps
```

【LAMMPS 翻译系列】shell 命令

由 www.52souji.net 发表于 2013 年 10 月 13 日 || 901 浏览

shell 命令用来执行系统 shell 命令。

内容目录 [隐藏]

使用语法

```
shell cmd args
```

- cmd：*cd* or *mkdir* or *mv* or *rm* or *rmdir* 或其他任何命令。
- args：命令的参数与 unix shell 相近。

使用举例

```
shell cd sub1

shell cd ..

shell mkdir tmp1 tmp2 tmp3

shell rmdir tmp1
```

```
shell mv log.lammps hold/log.1

shell rm TMP/file1 TMP/file2

shell my_setup file1 10 file2

shell my_post_process 100 dump.out
```

使用介绍

该命令用来执行 shell 命令。一些简单的基于文件的 shell 命令是直接支持的，与 unix 风格一致。没有列在上面的命令，会被直接传递个 C 库函数 system()，调用系统 shell 中的相应命令。

这是在你的输入脚本中调用其他命令的方法。比如，你希望将文件移动到另外的位置以便运行输入脚本的下一部分；或者你可以运行一个程序来提前处理要输入给 LAMMPS 的数据；再或者你可以运行一个程序来处理 LAMMPS 输出的数据。

除了 cd 这个命令之外，其他所有的命令都是在一个处理器上执行的，因此相关的文件或目录不会被多个处理器同时操作。

- cd 命令执行 unix 的 cd 命令，用来改变工作目录。所有下面的 LAMMPS 命令在读/写文件的时候都使用该目录。所有的处理器都执行该命令。
- mkdir 命令执行 unix 的 mkdir 命令，用来创建 1 个或多个目录。
- mv 命令执行 unix 的 mv 命令，用来重命名或移动一个文件或目录。
- rm 命令执行 unix 的 rm 命令，用来删除一个或多个文件。
- rmdir 命令执行 unixrmdir 命令，用来删除一个或多个目录。只有空目录才能被成功删除。

所有其他命令连同参数，会通过 C 库函数 system()，直接传递给系统 shell。

举个例子，下面的几行命令：

```
variable n equal 10

variable foo string file2

shell my_setup file1 $n ${foo}
```

与在 unix/linux 命令行下输入下面的命令是一样的：

```
% my_setup file1 10 file2
```

都是用运行可执行程序 my_setup，并给了三个参数 file1 10 file2。

使用限制

在这些命令被执行的时候，LAMMPS 不会检测是否出错，也不会打印警告信息。比如如果指定的文件不存储，执行 cd 命令就不会做任何事情。