Yujie Liu
2021.04.27

常用格式简介

一般用于GROMACS模拟的结构文件格式包括`gro`, `pdb`和`g96`这三种，前两种用的最多也最为频繁。

1. `pdb`格式全称为**Protein Data Bank**，顾名思义在蛋白数据库中用的非常多，记录的信息也非常全面，当然一般在GROMACS中用到无非是晶胞参数[CRYST1]，残基名称，原子名，坐标，元素名，温度因子等
2. `gro`格式全称为**GROMOS87**，记录的信息要比`pdb`格式少很多，一般我们关注的与上面提到的类似
3. `g96`格式全称为**GROMOS-96**，记录信息和`gro`差不多，结构和轨迹有区别

数据精度详解

一般而言，对于其他部分都相对固定，比如残基名，原子名等等。但是由于它们都是固定格式，不像`xyz`格式那样自由，因此精度也各不相同。

1. `pdb`格式坐标单位为**Å**，位宽为8，因此总共x、y、z坐标占24位长度，默认采用的是8.3f的格式，因此精度达到0.001 **Å**
2. `gro`格式坐标单位为**nm**, 默认输出位宽为8，因此总共x、y、z坐标占24位长度，默认采用的是8.3f的格式，因此精度达到0.001 **nm**，比`pdb`要低
3. `g96`格式坐标单位为**nm**，输出位宽为15，因此总共占据45位长度，采用%15.9lf的格式，因此精度达到了1E-9 **nm**

GROMACS处理这几类格式坐标的区别：
知道了每一类格式的内容，那么GROMACS分别是如何来读写这几类坐标的呢？

1. `pdb`格式而言，GROMACS的读取策略是固定位每8位宽一组，连续读取三组然后得到xyz坐标，最后转换成**nm**，具体实现**C**代码如下:

   ```
   char nc = '\0';
   for (k = 0; (k < 8); k++, j++)
   {
       xc[k] = line[j];
   }
   xc[k] = nc;
   for (k = 0; (k < 8); k++, j++)
   {
       yc[k] = line[j];
   }
   yc[k] = nc;
   for (k = 0; (k < 8); k++, j++)
   {
       zc[k] = line[j];
   }
   zc[k] = nc;
   x[natom][XX] = strtod(xc, nullptr)*0.1;
   x[natom][YY] = strtod(yc, nullptr)*0.1;
   x[natom][ZZ] = strtod(zc, nullptr)*0.1;
   ```

   因此可以看出如果是你自己写代码弄成`pdb`格式，只要保存的是固定位8位，精度[小数点]可以自己取，即使x、y、z精度不同gmx也是能读的。
   然而gmx写`pdb`格式的坐标部分严格按照`%8.3f%8.3f%8.3f`的格式，因此如果你一开始本身坐标位宽超过了8，写出来的坐标部分就会**错位！**

2. `gro`格式而言，GROMACS的读取方式是首先根据第一行坐标的小数点间隔位宽来确定读取的gro中的数据精度数多少，比如如果给的`gro`文件的坐标部分采用的是`%8.3f%8.3f%8.3f`，那么小数点位宽间隔就是8，然后gmx得到的精度就是8-5=3，因此为1E-3 **nm**。
   注意：如果你自己通过编程创建的坐标非常的大，某个维度数值非小数部分超过了4位数[默认8-(1+3)=4]，比如**12459.236**或者有负数的情况下**-1212.123**，该数字的总位宽就超过了8，所以默认的`%8.3f%8.3f%8.3f`就不再正确，固定格式发生了错位。如果这一行xyz三个坐标**不是**全部都是三位小数和相同位宽，且位于首行，那么gmx就会报错：

   ```
   The spacing of the decimal points in file xxx.gro is not consistent for x, y and z
   ```

   即使你第一行符合`%8.3f%8.3f%8.3f`，gmx此时确认精度为1E-3 **nm**，但后面不符合的依然是采用的第一行的8位位宽来固定读取数据，因此这部分数据实际读取的就是错误的。
   针对以上罕见错误[数据值太大引起的]，如果自己编程，最好是采用变精度方式来进行写坐标，比如坐标格式控制的那部分用`%15.6lf%15.6lf%15.6lf`，这样不仅位宽提升，数据实际精度变成了1E-6 **nm**, 且非小数部分的数值也达到了8位数[15-(1+6)=8]，注意一定要保证每一行的坐标都用相同格式！！
   GROMACS所有写`gro`操作均采用`%8.3f%8.3f%8.3f`，因此数据值过大会使得固定格式错位。
   注意：如果你一开始采用的变精度方式作为输入结构文件进行`pdb2gmx`等操作，然后得到一个输出gro文件，此时的格式就只是`%8.3f%8.3f%8.3f`，对于大数据任然会**部分**错位的哦！！！！！因此下一步使用该文件之前一定要处理一下！！！！

3. `g96`格式而言，GROMACS的读取格式为`%15lf%15lf%15lf`，而g96写格式统一为`%15.9f%15.9f%15.9f`，因此实际数据精度可以达到1E-9 **nm**，但是如前面说的它使用的是`%15lf%15lf%15lf`方式读取坐标部分，这样如果坐标数据出现若干空格[数据差异非常大的时候]，那么实际读取的值也会出错，比如`[space][space][space][space]4.47300005099991.476999998[space][space][space][space]0.416999996`这一种极端值gmx会读取失误，当然常规MD情况是不会的。

使用情形

1. `pdb`一般用于蛋白，小分子均可
2. `gro`一般用于小分子，变精度格式可以用于极端坐标值情况
3. `g96`一般用于小分子，精度非常高，比如进行一些参数拟合过程也可以使用

各自缺点

1. `pdb`冗余信息可能太多，格式控制复杂一些
2. `gro`默认的精度比较低[0.001 **nm**]
3. `g96`高精度会导致占用空间太大，特别是用作轨迹

小结

综上，出现坐标乱的极端情况就是数据太大或者太小的原因，因此自己最开始建模的时候就要尽量不要让某个维度的坐标位宽溢出。如果实在是不得已，最好就采用变精度`gro`格式，虽然降低`pdb`格式坐标精度以换取可用非小数位宽上限的方式也可以。