数据结构与算法课程设计小组报告

Team Report of Data Structure and Algorithm Course Design

文亦质 1800011702，尚游皓 1800011714

*College of Chemistry and Molecular Engineering, Peking University*, 100871

**摘要：**有机化合物的构象在基础有机化学以及后续的高等有机化学、生物化学的学习与理解中具有重要的作用。当分子的构型接近于正常键长和正常键角时，分子力学作为一种有效的近似计算方式可以得到很好的结果。在本次课程设计中，我们希望完成一个方便易用的、通过分子力学计算来观察有机化合物的优势构象的小程序。

* 需求分析

## 界面设计

1. 主窗口，作为所有内容的载体。
2. 一个3D显示界面，能够以立体方式显示出最终的分子构象。
3. 主窗口上的文本框，用于输出计算过程以及计算结果。
4. 主窗口上的菜单栏，用于进行各类操作；对分子参数进行微调的操作窗口。

## 3D绘图功能

从分子结构信息以及原子坐标绘制分子的立体结构同时响应用户的键盘与鼠标操作实时进行画面的改变。

支持在3D显示界面上进行的原子的选择以及响应对键角、键长、二面角的调整而改变画面以及原子坐标。

## 分子结构信息载体

实现一个类用来在程序中不同的模块间传递分子的结构信息（原子种类、键连关系、键级、电荷、原子的空间坐标）。

能够从文档中读取分子的结构信息，以及将分子的结构信息以某种方式存入文档。

## 计算部分

根据当前分子结构信息对分子结构进行分析，对原子进行分类。

根据对分子结构的解析代入力场数据进行计算寻找尽量优化的构象，并实时输出计算信息以及修改分子坐标使得当前分子形状能够在3D界面上显示出来；

计算时的用户可能的操作操作：启动、暂停、继续、停止。

* **非计算部分**

包括实现基础gui框架的gui.py，实现用于修改分子参数弹窗的toplevels.py，用于实现基于matplotlib库的3d绘图的ddd\_plot.py，用于储存分子基本信息与提供分子中键参数（键长、键角、二面角）的修改的molecule.py，用于实现兼容Gaussian View（一个商业化的成熟的三维分子构造的软件，可以利用建造3d分子的模型并导出文件，供其他的计算化学软件使用）输入文件的文件输入输出的file\_io.py。

## **Gui与三维绘图**

Gui 的实现基于tkinter，界面设计如前所述，而3D的绘图界面利用的matlpotlib的3D绘图功能。

gui.py中有Main\_windows、\_Main\_menu、\_DDD\_windows、\_DD\_windows、Log五个类。但是\_DD\_windows为二维的绘图类，由于人手不足和功能可以被3D绘图替代而放弃，没有被实现。

Main\_windows类是主窗口类，同时又拥有所有其他类的实例，实现主窗口的基础设定的同时初始化程序。\_Main\_menu实现了一个简易的菜单和一些基础的菜单功能。

\_DDD\_windows是为了将matlpotlib所绘的图嵌入tkinter所写的容器类，关键的参考了前人的方法1。同时注册了键盘事件和鼠标滚轮的事件，可以用键盘和鼠标滚轮调整3D视图的视角和距离。此外利用matplotlib的事件实现了对于3D图像中原子的选择和高亮。本来预计删除2D绘图的方式后想直接采用3D界面绘制分子，但是由于时间不足没有得以完成。

Log类实现了作为计算log输出的文本框和3D绘图的更新采用了tk的自定义事件，分别为log文本框更新和3D视图的更新在主窗口类中注册了<<UpadateLog>>和<<Upadate3DView>>两个自定义事件。

ddd\_plot.py中的DDD\_plot是基于matplotlib的绘图类，实现了基于多面体的原子的绘制和键的绘制。支持键级不同而不同的键的绘制，并且原子和键的颜色会随原子的不同而变化。

file\_io.py中的File\_IO类管理所有的文件输入和输出本质上所有的输入和输出文件都是纯文本文件，可以直接读取Gaussian View生成的输入文件。

toplevels.py为gui中的修改分子键参数的弹窗的设置。其中有三个类分别是设定键长、键角和二面角的窗口，因为需要设定三个复杂的窗口所以这个文件非常的长。主要是设定了许多的文本标签、输入框等，同时实现了通过点击选择原子的方法，更改键参数调用了Molecule类中对应的修改方式。

molecule.py中的Molecule类保存了分子的所有信息，是各个类之间交换分子信息的载体。同时分别定义了对于单原子和基团（后缀分别为A和G）调整键长、键角、二面角这三个键参数还有获取这三个键参数的的方法，方便使用。

原本使用的是自己写的向量的方法，但在处理基团的旋转的时候出现了难以处理的情况，而且计算速度较慢。后改为采用矩阵乘法的方法。通方法\_\_get\_ritation\_matrix获得以空间中的任意向量为轴旋转的矩阵，再将原子坐标与这个旋转矩阵相乘得到旋转后原子的坐标。

但是由于数学上叉乘和浮点数精度的限制，在角度接近于0度和180度的时候调整角度会出现一些难以预测的问题。

* **计算部分**

mm模块，含义为molecular mechanics。包括\scr\mm下的atomtype.py、bonding\_information.py和computing.py文件，此外使用到的力场数据在\datas下。

## 关于基础数据

首先需要分子力学的相关力场参数。由于分子力学的优势主要体现在对接近正常构型的大分子的计算中，当前能够找到的、比较成熟的力场参数主要是针对生物大分子，尤其是蛋白质与核酸的。通过在网络上的寻找，最终采用了amber03力场（来自开源软件gromacs的数据文档）。

Amber03力场中考虑了分子中的键长、键角、二面角以及非键相互作用（范德华力）。其中由于不同成键形式的同种原子（例如C）在其所成键的性质上有着巨大的差异，需要首先进行对化合物的结构解析以对这些原子进行分类。

## 结构解析 (bonding\_information.py)

找出化合物中的环系对原子种类判断起着关键作用（需要由此判断芳香性等特殊性质）。为此作者在mm模块中专门构建了一个类（bondingmap）用以给出所需的结构解析数据。其中的静态方法\_\_getframework()和\_\_findrings()用作解析化合物中的环系。以下是对结构解析（寻找环系）过程的算法实现的描述。

有机化合物的成键关系具有图的数据结构形式，在通用的Molecule类通过类似于邻接表（临接字典）的方式进行了实现，前面已经进行了叙述。在解析并储存结构的Bindingmaps类中，为了方便不同的查找方式（遍历所有的边、角、二面角，或是从某个顶点出发按照一定条件进行深度优先遍历），以Molecules类表示键连关系的有向图为基础翻译出对应的无向图以及有向图的邻接表。（Bondingmaps类的\_\_init\_\_()函数）基于不同的需要（以及编程时的前后纠结），一些类属性中存储了图的边上的信息而另一些没有。



图1 用于文中举例分析的两个环系。

鉴于环系的交错、并行，对图的单纯的深度优先搜索或是宽度优先搜索在这里不易实现（示例如图1。若简单记录已经搜索过的原子那么在对左侧的环搜索成功后对右边的环系的搜索就会存在障碍）。因此作者在静态方法\_\_findrings()中采用了递归形式的拓扑简化策略用以解析化合物中的环系。该思路的描述如下：

首先，只成一根键的原子（如H，Cl，F等）不可能存在于环系中。那么在图中反复删除度数为1的顶点即可得到只含有所有属于某个环系或是连结在两个环系之间的链的顶点（原子）的图（以邻接表的方式实现），在此处我们称之为framework.

此时在framework中的顶点可以分为两类：度数为2的顶点以及度数高于2的顶点（用crossings记录）。此时整个图可以抽象为以crossings中的顶点为顶点、以连接两个上述顶点的路径为边的图,我们暂且将其记为b。（在这个图中可能存在从某个顶点指向其自身的边，在实际实现时不将这些边记入b中；同时可能存在两个顶点间的多条边，在进一步的抽象中只保留一条，或将图b中的边理解为顶点间存在连接关系）。于是framework中的环可能是:

(1) 某个顶点以及从这个顶点指向其自身的路径；

(2) 从一个顶点到另一个顶点的两条路径加上这两个顶点;

(3) 在图b中成为环形路径的顶点与边的集合。

对于 (1)、(2) 中的环，在将framework抽象化的过程中便可以将其加入记录结果的列表里。抽象化的过程包括从某一个属于crossings的顶点出发进行图的深度优先搜索直到找到另一个属于crossings的顶点并将路径储存在字典中。而对 (3) 中的情况，需要找到图b中的环，此时可以进行递归调用。

由于应用上的考量，对于化合物中的环系，我们通常只关心较小的环系。例如上图中左边的十氢化萘中环路共有三个：(1, 2, 3, 4, 5, 6)；(5, 4, 7, 8, 9, 10)；(1, 2, 3, 4, 7, 8, 9, 10, 5, 6)，但我们只关心其中的两个六元环。因此在两个顶点间具有多条路径的情况下，我们选择用其中的最短路径与其他路径拼接来得到所有有效环系中尽量小的组合。这是 (2) 类环的具体实现方式。同样的，对 (3) 类环系，在得到图b的（合理最小）环形分量以后用每两个顶点间的最短路径连接即可得到对应于这些环形分量的最小环系。自此完成了对某张图的合理最小环形分量组合的寻找，返回记录结果的rings列表。

## 原子分类（atomtype.py）

此处主要是根据原子的杂化类型（可通过顶点度数来判断）、邻接原子类型、次近原子种类、所在环系类型对原子进行分类。由于N的判断需要用到C的分类，H的判断需要用到C、N、O的分类，在Atoms类中用了一个类属性来记录所有已经判断过的原子类型。依次用C/N/O/H/其他原子生成Atoms类的实例化但不保留引用，最终读取Atoms类的记录属性即可得到所有原子的分类。（这一部分的执行是在Computing类的实例化过程中实现的）

## 能量优化计算（computing.py）

计算所需的数据、临时变量、有关操作封装在Computing类中。由于运算量相对较大，计算过程中尽量使用numpy实现。程序改进历史可以在对应的GitHub仓库中查看到。

i. 计算基本思路与最初朴素的梯度下降法的实现

最初版本的计算思路采用了Gromacs文档中对最速下降法的介绍2。

朴素的梯度下降法在计算化学中又称最速下降法。一个化合物分子的能量作为分子中所有原子坐标的函数，可以理解为3N+1维空间中的一张曲面。从某一个初始坐标开始，计算该处的梯度，此时负梯度对应的向量就是势能函数下降最快的方向。于是可以尽量快的达到一个能量极小值点。

在临近能量极小值时，若保持恒定的步长不变，容易出现振荡现象。因此在发生能量上升时缩小步长；而当能量下降时缓慢增大步长以提高效率。

在Computing类的实例化对象中已经存储了待计算的分子的（此处能够考虑到的）所有相互作用信息。通过对这些相互作用的遍历并带入相应的势能计算函数得到能量。接下来通过计算所有相互作用的能量的偏导得到梯度，由此更新坐标以得到新的位置，开始下一轮计算。使用一个类属性记录计算次数。当计算一定次数后输出信息更新界面（以保证用户确认这个程序还在运行）。

经过初步测试，这种朴素的梯度下降算法能够迅速找到极小值点。

1. if self.\_times == 0:
2. self.\_potential = newpotential
3. else:
4. if newpotential < self.\_potential:
5. self.\_step \*= 1.2
6. self.\_potential = newpotential
7. self.sites = self.\_newsites[:]
8. else:
9. self.\_step \*= 0.6

ii. 加入随机因素的梯度下降算法

进一步的测试表明，上述朴素的梯度下降算法在很多时候找到的不是全局最小值，甚至不能算是稳定的极小值点。此外不同的开始很容易得到完全不同的结果使得计算不具有重现性。于是我们意识到在3N维空间中分子的总势能面并不是一个平滑的、自然导向全局最小值的曲面。这个曲面的形态我们甚至可以用“坑坑洼洼”来形容，其上充满了许许多多的局部极小的“浅坑”。实际过程中热涨落足以使得分子离开这个局部极小值而得到另一个构象（在实际中我们甚至不会称刚才的极小是一个稳定构象）。于是自然的思路是考虑模拟这种自然的热涨落来克服某些不稳定的极小值。

于是作者添加了两个用于计数的类属性，希望实现当能量基本收敛到局部极小时开始添加随机因素重新进行优化，并将此处的局部极小和之前的局部极小做比较。随机因素作用在整个分子上，用numpy的random.random()方法实现。修改后的判断代码如下：

1. if self.\_times == 0:
2. self.\_potential = newpotential
3. self.\_privpotential = self.\_potential
4. else:
5. if newpotential < self.\_potential:
6. self.\_step \*= 1.2
7. self.\_potential = newpotential
8. self.sites = self.\_newsites[:]
9. else:
10. self.\_step \*= 0.6
11. self.\_step \*= 0.5
12. self.\_countmin += 1
13. if self.\_countmin > 4:
14. if self.\_potential < self.\_privpotential:
15. self.\_privpotential = self.\_potential
16. self.\_privsites = self.sites
17. self.\_jumpstep = 0.01
18. self.\_countstop = 0
19. else:
20. self.\_countstop += 1
21. self.\_jumpstep \*= 0.01
22. forces = np.random.random((len(self.sites), 3)) \* self.\_jumpstep
23. self.\_newsites = self.sites + forces
24. self.\_countmin = 0
25. self.\_step = 0.01
26. return

在测试中，上述随机过程的确能够较好的增强每次计算结果的重现性。

iii. 破坏对称性和有针对性的随机过程

之后我们对环己烷分子进行了多次模拟以测试并调整热涨落的性能。初始状态是一个能量上非常不稳定的平面构型，而我们希望看到它在我们的优化算法下逐渐优化到稳定的椅式构象。但是多次实验的结果总是停留在平面构象，随机过程并没能起到很大的作用。

在某一次尝试中我们发现，当随意修改了某一个二面角以后，随机过程就足以使其到达椅式构象，而两者的能量相差的确非常大。结合对之前结果的分析，作者提出了破坏对称性的设想。平面构象十分难以摆脱的原因很可能是六元环的对称性使本来很大的梯度相互抵消，而由于random函数的均匀分布，模拟的随机热涨落也不易打破其对称性。此外，考虑到在模拟出的构象中通常是不合理的二面角不易破除，在针对所有分子的随机力之后加入了二面角的随机转动方式以破坏对称性以及越过某些能垒。二面角的转动使用了Molecules类中的modify\_dihedral\_angle\_A()方法。

接下来又加入了针对当前构象中的异常相互作用进行处理的改变，以期提高处理异常相互作用的效率。在基础的随机位移结束后首先选取当前相互作用势能最高的相互作用项，只计算该项的梯度进行处理得到新的坐标。经过实验这项操作在某些情况下有一定的用处。

加入二面角随机旋转后的程序在打破对称性、提供更加高效的新路径上表现稳定且令人影响深刻。

iv.实现计算的暂停、继续与中止

在计算的过程中，所有的有关重要变量（参数）都是以类属性的方式保存的，使得最终计算的暂停实现起来非常容易，在不删除原有计算对象的情况下只需添加一个新的类属性并在计算主循环里面检查这个类属性即可实现暂停。将主循环方法run()稍加修改、使其更加整齐，即可在继续计算时也调用run()方法。而直接修改计算中止条件的判据属性（self.\_countstop）即可实现强行中止计算。结果与讨论

* 软件使用说明

Pass

* 评价与总结

经过本次课程设计活动，我们对3D绘图方法、立体几何、分子力学以及随机过程有了较深刻的了解。也对计算化学的复杂度有了清醒的认识。由于python的运行速度问题，程序的看起来有点卡；文亦质同学尝试了使用多线程方法对计算效率进行优化，然而得不到显著的性能提升。

在此次合作开发中，我们尽可能的充分使用了面向对象编程的特点，将各自的板块进行封装以实现各自完成对应工作并取得了不错的效果；但在调试或是处理主界面的时候依然存在一些需要相互察看、阅读对方代码的情况。

最终实现的程序能够实现构象优化，同时如果配合上使用者的界面操作（手动调整角度）还可以探索化合物的不同构象之间的能量差异，对有机化学的学习可以起到一定的帮助。同时我们在这个过程中学到了很多，包括编程技能和计算化学方面的知识与技巧等，对后续的学习以及将来的科研工作有很大的帮助。

* 组内分工以及贡献排名

文亦质：程序基本框架、主界面设计、3D绘图、界面操作、文件处理

尚游皓：理论支持、计算部分

贡献排名：文亦质 > 尚游皓

* 参考文献

[1]https://blog.csdn.net/qq\_28485501/article/details/85329343

[2]https://jerkwin.github.io/GMX/GMXman-3/#3101-最速下降