计算几何作业报告

朱林 2200011028

2025 年 11 月 25 日

引言

本次作业旨在通过使用计算几何库CGAL，完成对蛋白质结构的凸包和alpha shape的计算，并实现2D凸包和Delaunay剖分算法。报告内容包括CGAL的安装与配置、利用CGAL求蛋白质的凸包和alpha shape，以及2D凸包和Delaunay剖分算法的实现与测试结果展示。如果要查看完整代码和对应的输入输出文件，请参考对应的Readme文件中的说明。（务必要先安装CGAL）

1 CGAL的安装与配置

1.1 Windows系统下的安装步骤

本次作业在Windows系统环境下完成全部任务。CGAL的安装步骤如下：

1. 安装Git到Windows系统
2. 使用Git克隆vcpkg仓库：`git clone https://github.com/Microsoft/vcpkg.git`
3. 进入vcpkg目录，运行安装批处理程序：`bootstrap-vcpkg.bat`
4. 使用vcpkg工具安装CGAL：`.\vcpkg install cgal`
5. 在Visual Studio中使用时需要将C++标准定为17以上

2 利用CGAL求蛋白质的凸包

2.1 蛋白质结构数据获取

本次作业使用的源数据来自RCSB PDB数据库[1]，数据格式为PDB格式。Protein目录下包含了十个蛋白质的本地PDB数据。蛋白质和对应中文名称如下表所示：

表 1: PDB ID与蛋白质中文名称对照表

PDB ID	蛋白质中文名称	分类与简介
1ZNI	胰岛素	**激素：** 调节血糖的激素。结构简单经典，由A、B两条链通过二硫键连接。
1MBN	肌红蛋白	**储氧蛋白：第一个被解析的蛋白质三维结构**，主要含α-螺旋，是结构生物学里程碑。
1UBQ	泛素	**调控蛋白：**广泛存在于真核细胞，用于标记需要被降解的蛋白质，结构非常紧凑稳定。
1BPI	胰蛋白酶抑制剂（BPTI）	**酶抑制剂：**一种小分子蛋白质抑制剂，通过多对二硫键保持结构稳定，是经典研究模型。
2HHB	血红蛋白	**运输蛋白：**运输氧气。是展示蛋白质**四级结构**和**变构效应**的完美范例。
1LYZ	溶菌酶	**酶：第一个被解析三维结构的酶**，能降解细菌细胞壁，具有明显的活性位点。
1EMA	绿色荧光蛋白	**荧光蛋白：**2008年诺贝尔化学奖明星，独特的β-折叠桶结构，内部有自发形成的发光团。
1TUP	p53 肿瘤抑制蛋白（DNA结合结构域）	**肿瘤抑制蛋白：**著名的"基因守护者"，其突变与超过50%的癌症相关。
1HHP	HIV-1 蛋白酶	**病毒酶：**艾滋病病毒复制的关键酶，是抗艾滋病药物的重要靶点，结构为对称二聚体。
1BL8	钾离子通道（KcsA）	**膜通道蛋白：**2003年诺贝尔化学奖成果，解决了离子选择性难题，只允许钾离子通过。

程序以血红蛋白(2HHB)为示例进行凸包计算。如果需要对于其它蛋白质求凸包，请更改程序中的路径为对应蛋白的PDB文件路径，如果所需蛋白质不在本地目录中，请自行查找PDB文件并放入protein目录下。

2.2 蛋白质文件读取和可视化

对于 2HHB（血红蛋白），利用常用蛋白质可视化软件 Chimer X 对 pdb 文件直接可视化如图1所示。

在本次作业中，我们只读取这个蛋白质中每个原子的位置，并不考虑不同原子的大小区别，而是将这个蛋白质看作一个三维空间上的点集，据此来判断其凸包。将PDB格式转换为点集合后，使用tecplot可视化结果如图2所示。

[1] https://www.rcsb.org/

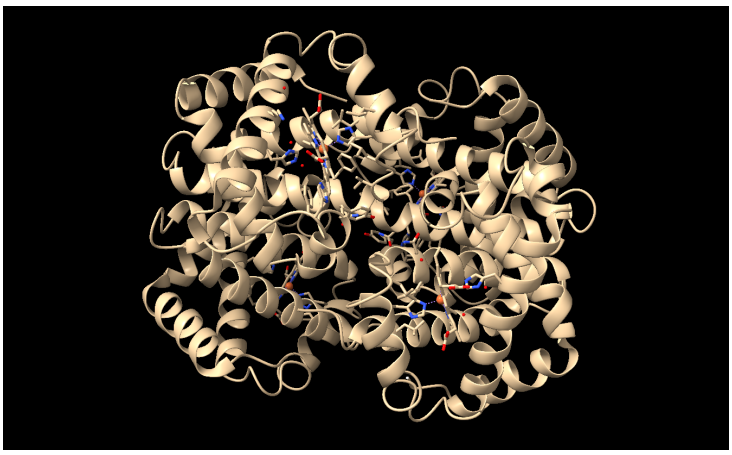

图 1: 2HHB在ChimerX下的可视化

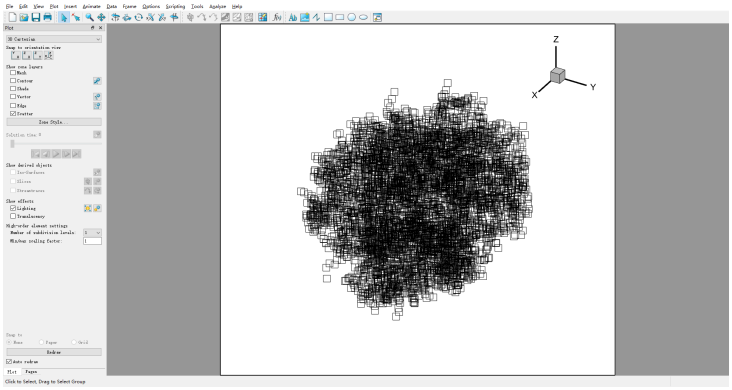

图 2: PDB格式转换为点集合后的可视化

2.3 使用CGAL计算凸包

使用CGAL中的`CGAL::convex_hull_3`函数直接计算凸包。计算结果如图3所示，其中凸包上的点被单独标记出来以便观察。

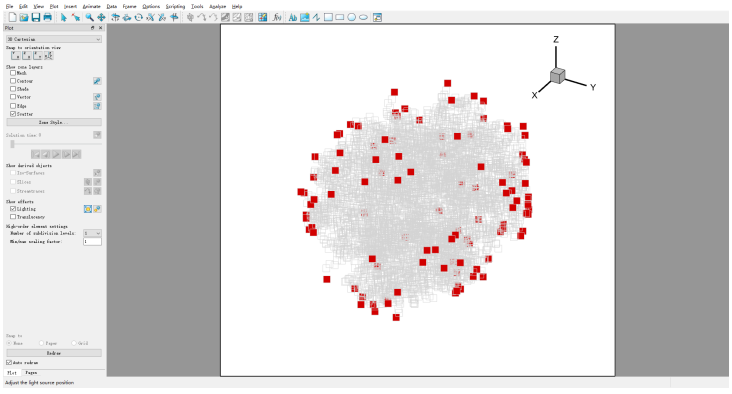

图 3: 凸包上的点单独标记后的可视化结果

3 利用CGAL计算蛋白质的alpha-shape

对于2HHB，直接调用CGAL中计算alpha shape对应的函数即可。下图中第一为所有点的可视化结果，第二为仅显示alpha shape边界点的可视化结果，alpha值取10.0。

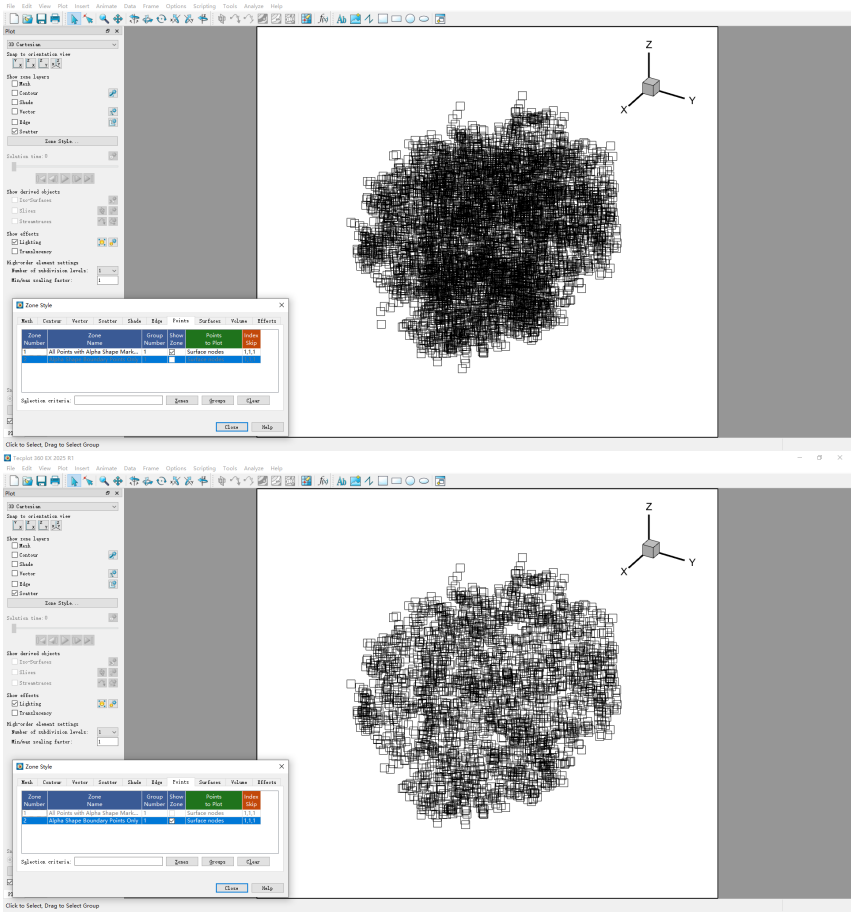

图 4: 2HHB的alpha shape可视化结果

4 2D凸包程序的实现

4.1 算法描述

本次作业采用卷包裹法(Gift Wrapping Algorithm)计算2D凸包，算法步骤如下：

Algorithm 1 卷包裹法计算2D凸包

1: 选取点集中x坐标最小的点作为凸包起点p_0
2: 初始化当前点$p \leftarrow p_0$
3: **repeat**
4: 将当前点加入凸包顶点集合
5: 选择点q，使得对于所有其他点r，向量$\overrightarrow{pq}$与$\overrightarrow{pr}$的叉积为正
6: $p \leftarrow q$
7: **until** $p = p_0$ ▷ 回到起点形成闭合多边形

算法的时间复杂度为$O(nh)$，其中n为总点数，h为凸包顶点数，在最坏情况下退化为$O(n^2)$。

4.2 可视化结果

使用SFML库[2]进行可视化，结果如下：

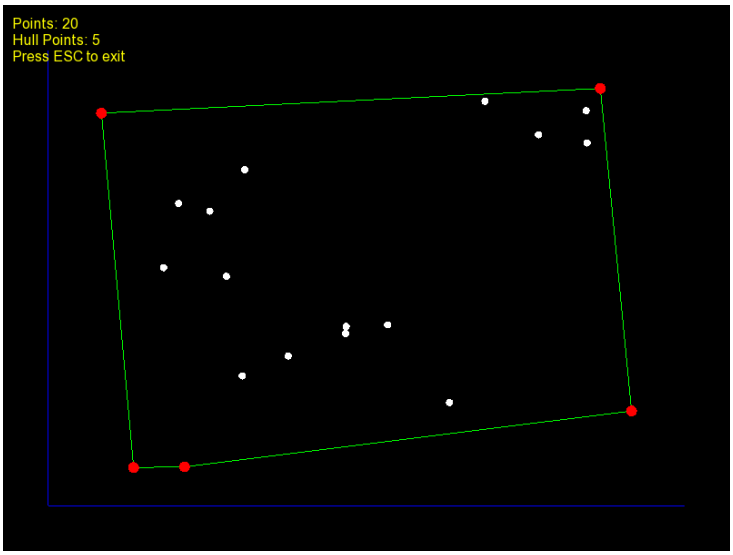

图 5: 2D凸包算法可视化结果

[2]这个库已经附带到作业文件中了，不需要另行安装下载

5 2D Delaunay剖分算法的实现

5.1 Bowyer-Watson算法

采用Bowyer-Watson算法实现2D Delaunay剖分，核心思想是逐点插入，并利用”空圆准则”保证三角网质量。

Algorithm 2 Bowyer-Watson算法实现Delaunay剖分

1: 创建包含所有点的”超级三角形”
2: **for** 每个点p **do**
3: 找出所有外接圆包含点p的三角形（”坏三角形”）
4: 删除所有坏三角形，形成插入多边形
5: 将点p与插入多边形的每个顶点连接，形成新的三角形
6: **end for**
7: 删除所有包含超级三角形顶点的三角形

5.2 时间复杂度分析

- **初始化阶段**：计算点集边界框，复杂度为$O(n)$
- **主循环（逐点插入）**：共执行n次迭代
  - 坏三角形检测：遍历当前所有三角形，单次复杂度$O(m)$，m为当前三角形数
  - 坏三角形删除：复杂度$O(m)$
  - 边去重处理：最坏复杂度$O(k^2)$，k为多边形边数
  - 新三角形生成：复杂度$O(k)$
- **总体时间复杂度**：
  - 最坏情况：$O(n^2)$
  - 平均情况：$O(n \log n)$到$O(n\sqrt{n})$之间

5.3 测试结果

使用可视化结果如下：

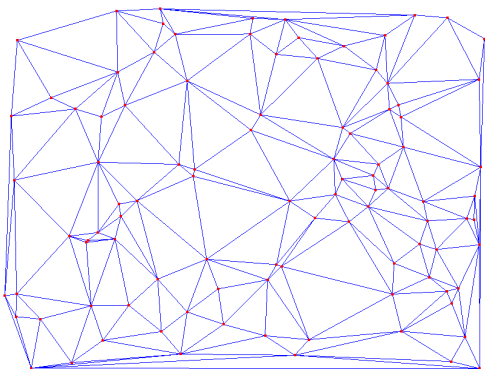

图 6: 2D Delaunay剖分算法可视化结果

使用10组数据的测试结果在results目录下。