Chapter 1

The Design Document of Simplicial Complex

1.1 数学概念

该节所有定义出自 Rotman(1988).

1.1.1 simplex

Definition 1.1. Let $\{p_0, p_1, \ldots, p_m\}$ be an affine independent subset of $\mathbb{R}^n$. The convex set spanned by this set, denoted by $[p_0, p_1, \ldots, p_m]$, is called the (affine) *m-simplex* with *vertices* $p_0, p_1, \ldots, p_m$.

Definition 1.2. Let X be a topological space. *A (singular) n-simplex* in X is a continuous map $\sigma : \Delta^n \to X$, where Δ^n is the standard *n-simplex*.

1.1.2 simplicial complex

Definition 1.3. If s is a *simplex*, then a *face* of s is a *simplex* s' with $\mathrm{Vert}(s') \subset \mathrm{Vert}(s)$.

Definition 1.4. A finite *simplicial complex* K is a finite collection of *simplexes* in some euclidean space such that:

(i) if $s \in K$, then every *face* of s also belongs to K;

(ii) if $s, t \in K$, then $s \cap t$ is either empty or a common *face* of s and of t.

Definition 1.5. If K is a *simplicial complex*, define its *dimension*, denoted by $\dim K$, to be

$$\dim K = \sup_{s \in K} \{\dim s\}$$

(of course, a *q-simplex* has dimension q).

Definition 1.6. Let K be a *simplicial complex* and let $p \in \mathrm{Vert}(K)$. Then *the star of* p, denoted by $\mathrm{st}(p)$, is defined by

$$\mathrm{st}(p) = \bigcup_{\substack{s \in K \\ p \in \mathrm{Vert}(s)}} s^\circ \subset |K| .$$

1.2 c++ 类设计

1.2.1 Simplex

功能

1. SimplicialComplex 的基础成员.

2. 记录组成 simplex 的所有 vertices.

数据成员

1. `using Vertex = unsigned int;`
   vertex 通过 unsigned int 唯一表示.

2. `set<Vertex> vertices;`
   构成 Simplex 的 vertex 集合, size 等于 simplex 的 dimension + 1.
   (对 vertices 排序, 便于对 Simplex 排序和哈希, 用于容器保存.)

函数成员

1. `int getDimension() const;`
   输出: simplex 的 dimension.

1.2.2 SimplicialComplex

功能

1. 包含组成 simplicial complex 的所有 simplexes.

2. 记录并提供所有 vertices 的相邻信息.

3. 添加和删除 simplex 并保持数据一致性.

数据成员

1. `vector<set<Simplex>> simplexes;`
   simplicial complex 包含的所有 simplexes, simplexes[i] 是 i-simplex 的集合.
   (添加和删减 simplicial complex 中的 simplex 时需要保证 simplex 的唯一性, 所以采用 set 保存.)

2. `using SimplexIter = set<Simplex>::iterator;`
   Simplex 在 simplexes 中的迭代器.

3. `unordered_map<Vertex, set<SimplexIter>> mVertex2Simplex;`
   冗余数据, 记录包含对应 vertex 的所有 Simplex 在数据成员 simplexes 中的迭代器.
   (在计算 vertex 的 star, link 和删除 simplex 时, 添加该冗余数据能提高搜索效率.)
   (在 erase 中需要计算和删除一些 simplex, 使用 set 存储第二项能优化这一过程.)

函数成员

1. `int getStarClosure(Vertex p, SimplicialComplex& closure) const;`
   输入: vertex p.
   输出: 返回值 0, 1 表示 p 是否被 complex 包含.
   效果: 返回 1 时, closure 赋值为 st(p) 的闭包.

2. `int getLink(Vertex p, unordered_set<Vertex>& res) const;`
   输入: vertex p.
   输出: 返回值 0, 1 表示 p 是否被 complex 包含.
   效果: 返回 1 时, res 赋值为所有与 p 相邻的 vertex 的集合.

3. `int insert(const Simplex& s);`
   输入: simplex s.
   输出: 返回值 0, 1 表示 s 是否被 complex 包含.
   效果: 返回 1 时, 将 s 的所有的 face 和 s 插入到 complex 中.

4. `int erase(const Simplex& s);`
   输入: simplex s.
   输出: 返回值 0, 1 表示 s 是否被 complex 包含.
   效果: 返回 1 时, 将 complex 中所有以 s 为 face 的 simplex 和 s 移除.

5. `int getDimension() const;`
   输出: simplicial complex 的 dimension.

1.2.3 YinSet

模板: `template<int Dim, int Order>`

Dim 表示空间维数.(目前仅考虑 Dim = 2)
Order 表示 YinSet 边界的近似阶数.

功能

1. 由于 YinSet 组成的 Yin 空间和 YinSet 的表示空间 Jordan 空间的同构性, 记录 YinSet 边界来表示 YinSet.

新增功能

1. 二维情况, 记录 YinSet 边界上一个 0-simplicial complex 表示的 kinks.

2. 将 simplicial complex 中的 vertex 单射映射到 YinSet 边界上的 knot.

原有数据成员

1. `vector<Curve<Dim, Order>> segmentedCurves;`
   表示 YinSet 的边界 Curve 集合.

替代数据成员

1. `vector<OrientedJordanCurve<Dim, Order>> orientedJordanCurves;`
   表示 YinSet 边界的 Jordan Curve 集合.

新增数据成员

1. `SimplicialComplex kinks;`
   记录 orientedJordanCurves 上的一些特殊点.

2. `using PointIndex = pair<unsigned int, unsigned int>;`
   orientedJordanCurves 检索 knots 的 index 类型.

3. `map<Vertex, PointIndex> mVertex2Point;`
   将 kinks 的 vertex 映射到 orientedJordanCurves 的 knot.

4. `map<PointIndex, Vertex> mPoint2Vertex;`
   将 orientedJordanCurves 上的部分 knots 映射到 kinks 的 Simplexes 的 vertices 中.

新增函数成员

1. `const SimplicialComplex& getKinks() const;`
   输出: 数据成员 kinks 的 const 引用.
   (直接修改 kinks 可能会导致 YinSet 内 vertex 和 point 无法一一对应, 输出 const 引用读取同时禁止修改)

2. `void resetAllKinks(const vector<PointIndex>& vertices);`
   输入: 新 kinks 在 orientedJordanCurves 中的 PointIndex 集合 vertices.
   效果: 重置 kinks, 根据新尖点拟合所有 Jordan Curves.

3. `int vertex2Point(Vertex p, PointIndex& index) const;`
   `int vertex2Point(Vertex p, rVec& point) const;`
   `int point2Vertex(const PointIndex& index, Vertex& p) const;`
   输入: 检索的 Vertex 或 PointIndex
   输出: 返回值 0, 1 表示是否成功.
   效果: 引用参数赋值查询结果.

4. `int insertKink(const PointIndex& index);`
   输入: knot 的 PointIndex.
   输出: 返回值 -1, vertex 分别表示 index 已经在 kinks 中和插入的 vertex.
   效果: 返回非负值时, kinks 插入 vertex, 重新拟合相关的 Jordan Curves.

5. `int eraseKink(Vertex p);`
   输入: vertex p.
   输出: 返回值 -1, vertex 分别表示 p 不在 kinks 中和移除的 vertex.
   效果: 返回非负值时, kinks 删除 vertex, 重新拟合相关的 Jordan Curves.