分类号 O159　　密　级 无
编　号 10028　　学　号 2170502130

# 首 都 师 范 大 学 硕 士 论 文

# 高维最优传输算法的设计与实现

研 究 生 姓 名: 李博

指导教师姓名、职称: 雷娜 教授，于祖焕 教授

学 科、 专 业 名 称: 数学，数字与信息技术

研 究 方 向: 计算几何

2020 年 3 月 25 日

# 论 文 原 创 性 声 明

本人郑重声明: 所呈交的学位论文, 是本人在导师指导下, 独立进行研究工作所取得的研究成果. 除文中已经标明引用的内容外, 本论文不包含任何其他个人或集体已发表或撰写的研究成果. 对本文的研究做出贡献的个人和集体, 均已在文中以明确方式标明. 本声明的法律结果由本人承担.

学位论文作者 (签名):

年　　月　　日

# 首都师范大学学位论文使用授权协议书

本人完全了解首都师范大学有关保留，使用学位论文的规定，学校有权保留学位论文并向国家主管部门或其指定机构送交论文的电子版本和纸质版，有权将学位论文用于非赢利目的的少量复制并允许论文进入学校图书馆被查阅。有权将学位论文的标题和摘要汇编出版。保密的学位论文在解密后使用本规定。

学位论文作者 (签名): _____

学　　号: _____

导师签名: _____

日期:　　年　　月　　日

# 目　　录

# 摘　　要

最优化理论由于较高的经济应用价值，而受到广泛关注。最优传输是最优化理论的热门领域，比如线性规划理论。线性规划解决的是离散集合到离散集合的最优映射，但是最优传输理论讨论的更一般: 源集到目标集可以是连续空间，也就是说最优传输讨论集合到集合的最优映射。由于 Monge 和 Kantorovich 最先意识到这类问题并且 Kantorovich 发明了线性规划而获得诺贝尔经济学奖，所以又称 Monge-Kantorovich 运输问题。Brenier 定理暗示最优传输映射与 Alexandrov 定理的等价关系, 所以和几何产生了联系，并为欧式空间最优传输的算法提供了理论支撑。针对最优传输实现的算法大概包括线性规划方法，最大流方法，Sinkhorn 算法，Benamou 算法，求解蒙日-安培方程方法以及凸几何方法等。

本文实现了高维最优传输, 并且主要创新工作有以下几点:

(1) 使用反对称张量使得空间定向得到统一的解释。

(2) 本文实现了高维的最优传输，并且提供了 c 语言的高维网格库 libcell。

**关键词**: 最优传输，算法，几何

# ABSTRACT

Optimization theory is widely concerned because of its high economic application value. Optimal transportation is a hot field of optimization theory, such as linear programming theory. Linear programming solves the optimal mapping from discreate set to discrete set, so the discussion of optimal transportation theory can discuss more general cases:source and target set can be continuous set. In other words, optimal transportation theory discusses the best mapping from set to set. Because Monge and Kantorovich study this question firstly and Kantorovich won the Nobel Prize in Economics by linear programming, this question is also called Monge-Kantorovich transportation question. Brenier's theorem implies the equivalence between the optimal transfer mapping and Alexandrov's theorem , so it's connected to geometry and provied the theory for the optimal transportation algorithm of Eucidean space. The algorithms for optimal transportation include Linear Programming method, Maximum Flow method, Sinkhorn algorithm, Benamou algorithm, Monge-Ampere equation method and Convex Geometry method.

In this paper, high-dimensional optimal transportation is realized, and the main innovations are as follows:

(1).The use of antisymmetric tensor makes the spatial orientation get a unified explanation.

(2).In this paper, the high-dimensional optimal transportation is realized, and the high-dimensional grid Library of C language is provided.

**Key words:** Optimal Transportation,Algorithm,Geometry

# 第一章　引言

## 1.1　最优传输算法概述

机器学习如日中天，其中 GAN 模型 [1] 尤其火热，GAN 模型的思想设计类似哲学阴阳平衡。一个生成模型，一个判别模型. 判别模型用于判断图片是否属于真实集合，生成模型去尽可能创造和真实数据集相似的图片。WGAN 模型 [2] 将 Wasserstein 距离替换 GAN 模型的 J-S 散度（实际上等价对数字进行 2 进制编码的位数），获得了更好的结果。Wasserstein 距离可以用来衡量两个概率分布的距离，而满足 Wasserstein 距离的映射就是最优传输映射，所以 WGAN 模型可以利用最优传输理论进行解释。顾险峰等在文献 [3] 利用最优传输给出 WGAN 模型的几何观点。正是由于工程行业的追捧才使得最优传输理论得到广泛的关注和学习，而在此之前最优传输只是数学家的研究对象。

最开始 Monge 考虑如下问题: 如何移动土堆到相同体积的容器内，使得移动的代价最小？用数学语言表示为:

$$\min_{T} \int_{X} ||x - T(x)||dx$$

其中 $T$ 是土堆到容器的映射，$X$ 是在土堆的积分区域。

Monge 问题的最优传输方案要求是一一映射，也就是传输方案是一对一的关系。Kantorovich 放宽了这一限制，并且在 1990 年 Kantorovich 由于发明线性规划而获得诺贝尔经济学奖。

由于两人的贡献，统一将以下问题称为 Monge-Kantorovich 问题:

$$\min_{\pi\in\Pi} \int_{\pi} c(x-y)\,d\pi(x,y), \pi \in \Pi : \{\pi[A \times Y] = \mu[A], \pi[X \times B] = \upsilon[B], A \in X, B \in Y\}$$

其中 $(X,\mu),(Y,\upsilon)$ 是两个测度空间，$c(x,y)$ 是代价函数。

Brenier 定理指出：最优传输问题在某些限制条件下等价于某个凸函数的梯度映射。Alexandrov 在文献 [4] 给出了 Alexandrov 问题的存在性和唯一性证明，而 Alexandrov 问题中所要求解的凸包络正是 Brenier 定理中的凸函数。这样最优传输理论和凸几何理论产生了联系。

在数学界, 法国数学家 Cedric Villani 由于在最优传输的工作获得菲尔兹数学奖 [5]。此外当目标集合和源集合的总测度不相等时，此时的最优传输问题称为非平衡最优传输问题 [6]。在实际应用中，问题多为非平衡最优传输问题，故而非平衡最优传输有蓬勃的发展空间。

在算法领域, 丘成桐等人在文献 [7] 给出 Alexandrov 问题的构造性证明，由此给出最优传输算法。苏科华等人在文献 [8] 提出基于 GPU 的最优传输算法，因为在最优传输算法中面积的计算至关重要，而利用 GPU 图像投影可以充分释放算力。

在 2014 年 Bruno Levy 在《A numerical algorithm for L2 semi-discrete optimal transport in 3D》 [9] 一文中通过构造凸能量并优化此能量给出最优传输算法。顾险峰等在文献 [10] 也通过构造凸能量的方式给出最优传输算法。

在《Computational Optimal Transport》 [11] 一书中详细地介绍了基于测度的算法:Sinkhorn。

在流体力学领域，最优传输也存在相应的算法。然而由于计算量非常巨大，流体力学的算法很难实现高维的情况 [12]。

除此之外还有比较新 SOT 算法，由于了解有限，不再介绍 [13]。

在算法应用领域, 顾险峰等在文献 [14] 中利用最优传输算法实现了保面积参数化。结合共形映射使得可以调节参数化的结果是偏向保面积或者保角。Cedric Villani 等在文献 [15] 中将最优传输理论应用到 Ricci flow, 使得 Ricci flow 可以定义到带有测度的图上 [16]。

# 第二章　预备知识

## 2.1　基础知识

**定义 1** (凸函数). [17] 以下的定义限制定义域 $\Omega$ 是凸集。对于 $n$ 维欧式空间凸区域 $\Omega$ 上的 $n$ 元函数 $f$, 如果满足以下条件之一, 则称它是凸函数。

*1)* 对于 $\Omega$ 中的一条线段 $[p_1, p_2]$, 满足 $f(\lambda p_1 + (1-\lambda) p_2) \leq \lambda f(p_1) + (1-\lambda) f(p_2)$, 其中 $0 \leq \lambda \leq 1$。

*2)* 对于 $\Omega$ 中的一条线段 $[p_1, p_2]$, 满足 $f\left(\frac{1}{2}(p_1 + p_2)\right) \leq \frac{1}{2}(f(p_1) + f(p_2))$。

*3)* 对于 $\Omega$ 中的每点存在凸开集 *(*邻域*)*$U$, 使得乘积空间 $U \times [f(U), +\infty]$ 是凸集。

以上定义对连续函数均成立, 无论是否可微。下面的定义要求 $f$ 二阶可微。

*4)* 对于 $\Omega$ 中的每点 $p$, 满足 *Hessian* 矩阵 $\frac{\partial^2 f}{\partial x_i \partial x_j}$ 是半正定矩阵。

**命题 1.** $\Omega$ 是欧式空间的凸集,$f$ 是 $\Omega$ 上的函数。$f$ 是凸函数当且仅当对于 $\Omega$ 中的开覆盖 $U_i$, 在每个 $U_i$ 上 $f$ 是凸函数。

证明:

$\Longrightarrow$:$f$ 在 $\Omega$ 上是凸函数, 那么在 $\Omega$ 的子集上依然是凸函数。

$\Longleftarrow$: 对于线段 $[p_1, p_2]$, 设 $p(t) = p_1 + t(p_2 - p_1)$, 则凸函数定义 1 等价于 $\frac{f(p(t)) - f(p(0))}{t} \leq \frac{f(p(1)) - f(p(t))}{1-t}, 0 \leq t \leq 1$, 也就是平均速度随 $t$ 递增。

对于线段 $[p_1, p_2]$, 随 $t$ 依次经过的开集有 $U_1, ...U_n$, 称时间序列 $t_0 < t_1 < t_2 < ... < t_{2i+0} < t_{2i+1} < .. < t_{2n-2} < t_{2n-1}, 0 \leq i \leq n-1$ 为过渡时间序列, 如果它满足 $p(t_{2i+0}) \in U_i \bigcap U_{i+1}, p(t_{2i+1}) \in U_i \bigcap U_{i+1}$。那么对于线段 $[p_1, \frac{p_1+p_2}{2}]$ 和 $[\frac{p_1+p_2}{2}, p_2]$, 分别取它们的过渡时间序列 $t_0, t_1..t_k$ 和 $t_0', t_1', ...t_l'$。则有 $0 < t_0 < t_1 < ....t_k < \frac{1}{2} < t_0' < t_1' < ...t_l' < 1$, 因为在每个 $U_i$ 上 $f$ 都是凸函数, 所以平均速度在 $U_i$ 上随时间 $t$ 递增。

设以上时间间隔的平均速度为 $s_i$, 由不等式的传递性得到 $s_i < s_j, i \leq j$, 所以 $f\left(p\left(\frac{1}{2}t\right)\right) - f(p(0)) \leq f(p(1)) - f\left(p\left(\frac{1}{2}t\right)\right)$。 □

**命题 2.** $\Omega$ 是欧式空间的凸集,$f$ 是 $\Omega$ 上的函数。那么 $f$ 是凸函数当且仅当 $f$ 满足 $\sum \lambda_i f(p_i) \geq f(\sum \lambda_i p_i), \sum \lambda_i = 1, 1 \geq \lambda_i \geq 0$, 其中 $p_i$ 是在 $\Omega$ 上采样的 $m$ 个点。

证明:　$\Longrightarrow$: 凸函数的定义 (1) 是本命题条件的特例, 得证。

$\Longleftarrow$: 这里采用递推法。假设对 $\Omega$ 区域采样 $n$ 个点 $p_i$, 成立 $\sum \lambda_i f(p_i) \geq f(\sum \lambda_i p_i)$, $\sum \lambda_i = 1, 1 \geq \lambda_i \geq 0$。

现在添加一点 $q$,$\forall \lambda_i \sum_{i=1}^{n+1} \lambda_i = 1, 1 \geq \lambda_i \geq 0$, $\sum \lambda_i f(p_i) + \lambda_{n+1} f(q) =$

$\left(\sum_{i=1\ldots n} \lambda_i\right) \frac{\sum \lambda_i f(p_i) + \lambda_{n+1} f(q)}{\left(\sum_{i=1\ldots n} \lambda_i\right)} \geq \left(\sum_{i=1\ldots n} \lambda_i\right) \left(f(\sum k_i p_i) + \frac{\lambda_{n+1} f(q)}{s}\right)$,

其中 $s = \left(\sum_{i=1\ldots n} \lambda_i\right), k_i = \frac{\lambda_i}{s}$。

再次应用定义 1 的式子得到 $(sf(\sum k_i p_i) + \lambda_{n+1} f(q)) \geq f(\sum \lambda_i p_i + \lambda_{n+1} q)$。 □

**定义 2** (共轭函数). [17] $\Omega$ 是 $n$ 维欧式空间中的凸区域，$g$ 是 $\Omega$ 上的连续函数，则 $g(x)$ 的共轭函数定义为 $g^*(y) = \sup_x (< x, y > -g(x))$。

**引理 1.** 如果 $g(x)$ 是凸集 $\Omega$ 上严格凸函数，且二阶可微，那么它的梯度映射是单射。

证明: 假设 $p_1, p_2$ 是定义域两个点，满足 $p_1 \neq p_2, \nabla g(p_1) = \nabla g(p_2)$. 接下来的推导会出现矛盾。设曲线 $(1-t)p_1 + tp_2$。那么 $g'(t) = \nabla g(x) \frac{\partial x}{\partial t} = \nabla g(x)(p_2 - p_1)$, 根据凸函数第四个定义有:

$g'' = \frac{\partial x}{\partial t} \frac{\partial^2 g}{\partial x_i \partial x_j} \frac{\partial x}{\partial t} + \frac{\partial g}{\partial x} \frac{\partial^2 x}{\partial t^2} = \frac{\partial x}{\partial t} \frac{\partial^2 g}{\partial x_i \partial x_j} \frac{\partial x}{\partial t} = (p_1 - p_2)^T \nabla^2 g (p_1 - p_2) > 0$, 那么我们得到下面两个矛盾的结果

- $g'(0) = g'(1)$
- $g'$ 单调

矛盾。 □

当 $g$ 是一阶可微且严格凸函数时，上述梯度映射同样是单射。将上述证明中 $g'(t)$ 视为速度，那么速度是递增的，仍然会导出矛盾的结果。

为记述方便，下面的符号中 $\nabla g^-$ 表示 $(\nabla g)^-$，即 $g$ 梯度映射的逆映射。

**引理 2.** 当 $g$ 是一阶可微且严格凸函数（意味着梯度映射是单射），$(g^*)^* = g$。

证明: 设 $f(x, y) = g(x) - < x, y >$, 那么对于定义域两点 $x_1, x_2$, 有

$f\left(\frac{x_1+x_2}{2}, y\right) - f(x_1, y) \leq f(x_2, y) - f\left(\frac{x_1+x_2}{2}, y\right)$, 即 $f$ 关于 $x$ 是凸函数。则 $\inf_x f(x, y)$ 的 $x$ 满足 $\nabla g(x) = y$, 由上面的引理知道,$x = \nabla g^-(y)$。也就是 $g^*(y) = -\inf_x f(x, y) = \langle y, (\nabla g)^{-1}(y) \rangle - g\left((\nabla g)^{-1}(y)\right)$。接下来证明 $\nabla g^* = \nabla g^-$。

$$
\begin{aligned}
\nabla (g^*(y)) &= \langle \nabla x, y \rangle + x - \langle \nabla g(x), \nabla x \rangle \\
&= \langle \nabla x, y \rangle + x - \langle y, \nabla x \rangle \\
&= x \\
&= \left(\nabla g^-\right)(y)
\end{aligned}
$$

然后代入得到

$$\begin{aligned} g^{**}(y) &= \langle y, (\nabla g^*)^{-1}(y)\rangle - g^*\left((\nabla g^*)^{-1}(y)\right) \\ &= \langle y, (\nabla g)(y)\rangle - g^*((\nabla g)(y)) \\ &= \langle y, (\nabla g)(y)\rangle - \langle y, (\nabla g)(y)\rangle + g(y) \\ &= g(y) \end{aligned}$$

□

现在记 $\tilde{g}(x)$ 为集合 $\{p : g(y) \geq (p, (y-x)) + g(x), \forall y\}$, 记 $\tilde{g}^-(z)$ 为 $\{p : \tilde{g}(p) = z\}$, 也就是 $\tilde{g}(x) = z$ 的原像集合。有时为了区分符号,$\tilde{g}^-$ 也可写作 $(\tilde{g})^-$。[1]

**引理 3.** $g^*(y) = \sup_x < x, y > -g(x) =< l, y > -g(l), l \in \tilde{g}^-(y)$。

证明: 可以证明 $\forall x, (l, y) - g(l) \geq (x, y) - g(x)$, 因为由上面 $\tilde{g}$ 和 $\tilde{g}^-$ 的定义知道 $(x - l, y) + g(l) \leq g(x)$。且立马得到对于任意两个 $l \in \tilde{g}^-(y), l_1 \in \tilde{g}^-(y)$, 有 $(l, y) - g(l) = (l_1, y) - g(l_1)$。 □

其实上面定义的 $\tilde{g}(x)$ 是 $g$ 在 $x$ 处的承托超平面[2]的梯度, 而 $< l, y > -g(l)$ 与在 $l$ 处的承托超平面的截距 $h$ 相差个负号。

所以如果记 $h(l)$ 为 $g$ 在 $l$ 处承托超平面的截距, 有下面的推论:

**推论 2.1.1.** $g^*(y) = -h(l), l \in \tilde{g}^-(y)$。

**引理 4.** *当 $g$ 是连续凸函数时*, $(g^*)^* = g$。

证明: 先证明 $\tilde{g}^*(x) = (\tilde{g})^-(x)$, 按照定义只需证明 $\forall p \in \tilde{g}^-(x), \forall y, st, g^*(y) \geq (p, y - x) + g^*(x)$,

$$\begin{aligned} &(l, y) - g(l) \geq (p, (y - x)) + (l_1, x) - g(l_1), l \in \tilde{g}^-(y), l_1 \in \tilde{g}^-(y), p \in \tilde{g}^-(x) \\ &(l - p, y) - g(l) \geq (l_1 - p, x) - g(l_1) \\ &(p - l, y) + g(l) \leq (p - l_1, x) + g(l_1) \\ &(p - l, y) + g(l) + (l_1 - p, x) \leq g(l_1) \\ &(p - l, y) + g(l) + (l_1 - p, x) \leq g(p) + (l_1 - p, x) \leq g(l_1) \end{aligned}$$

现在证明 $g^{**} = g$:

$g^{**}(y) = (l, y) - g^*(l), l \in \tilde{g}^{*-}(y)$。

$(l, y) - g^*(l)$ 展开就是 $(l, y) - (l, p) + g(p), l \in \tilde{g}^{*-}(y), p \in \tilde{g}^-(l)$。由上面的引理3知道 $l$,$p$ 在集合的取值不影响函数的结果, 得到 $p = y$, 得证。 □

[1]的定义也就是次微分, 它在文献 [18] 中定义。

[2]承托超平面在文献 [18] 中有定义

对于 $n$ 维欧式空间凸区域 $\Omega$, 我们可以采样足够多的点 $p_i$，给每个点 $p_i$ 配一个数值 $f_i$. 对于这些点可以构造 $n$ 维单形的剖分 (不一定使用所有点)，记为网格 $m$。对于每个单形 $p_0, ...p_n$ 内部的点 $\lambda_0 p_0 + \lambda_1 p_1 + .. + \lambda_n p_n, 1 \geq \lambda_i \geq 0, \sum \lambda_i = 1$, 配有数值 $\sum f_i \lambda_i$, 这样构成的网格区域 $m$ 上函数记为 $f_m$。

明显网格的函数 $f_m$ 由网格的单形剖分唯一决定。

**定义 3** (凸包络). 网格的一种剖分,使得网格上的函数 $f_m$ 满足 $\sum \lambda_i f_i \geq f_m(\sum \lambda_i p_i)$, $1 \geq \lambda_i \geq 0, \sum \lambda_i = 1$。

**定义 4** (退化点). 对于点集 $p_i$ 上凸包络，如果没有使用到 $p_0$ 点，那么称 $p_0$ 是这个凸包络的退化点。

**定义 5** (凸调整). 寻找网格的凸区域 $C$: 在 $C$ 上的函数 $f_m$ 不是凸包络。那么调整这个区域 $C$ 上的网格剖分使 $f_m$ 在 $C$ 上为凸包络。这样的网格调整称为凸调整。

给一个初始网格 $m_0$，记算子 $\mathcal{A}$ 为一次凸调整，$m_i = \mathcal{A}(m_{i-1})$。

**命题 3.** 网格 $m_i$ 经过有限凸调整后不再改变，且为凸包络。

证明: 定义一种函数上的偏序关系: 记 $f \geq g$, 如果 $\forall x \in \Omega, f(x) \geq g(x)$。
那么 $f_{\mathcal{A}(m_i)} \leq f_{m_i}$, 且网格上的凸包络就是它的下界。由佐恩引理知道，这个偏序记必存在极小元 [19]。而有限点的剖分情况总是有限的，所以 $m_i$ 必定经过有限步取到下界。又由命题1知道:$f_m$ 在每点周围的单形区域是凸函数，那么 $f_m$ 在整个网格必定是凸函数。也容易用反证法证明它是凸包络。 □

工程算法在二维 Delaunay 剖分时，常常采用一种不断翻转边的方法，那么问题是：是否这种网格调整会陷入循环当中？上面的命题回答了这个问题，实际上还要先证明 Delaunay 剖分跟凸包络的等价性, 还要证明这种翻转边的调整等价凸调整。

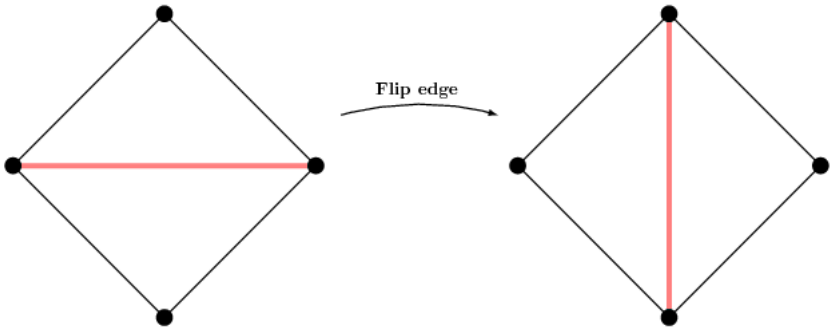



图 2.1: 翻转边

**命题 4.** 如果在某个凸函数 $f$ 的定义域上采样 $m$ 个点 $p_i$, 那么配有数值 $f_i = f(p_i)$ 的点集 $p_i$ 生成的凸包络必定没有退化点。

证明: 不妨假设 $f(p_0)$ 位于凸包络的上方，那么 $p_0$ 必定落在由 $p_i$ 组成某个单形内，并且可以表示为 $p_0 = \sum p_i\lambda_i, \sum \lambda_i = 1, 1 \geq \lambda_i \geq 0$。因为 $f$ 是凸函数，所以由命题2得到不等式 $f(p_0) \leq \sum \lambda_i f(p_i)$, 矛盾。 □

比如下图在 $x$ 轴采样点集 $x_i$，并配有数值 $f_i = x_i^2$, 那么这些点集的凸包络必定没有退化的点。

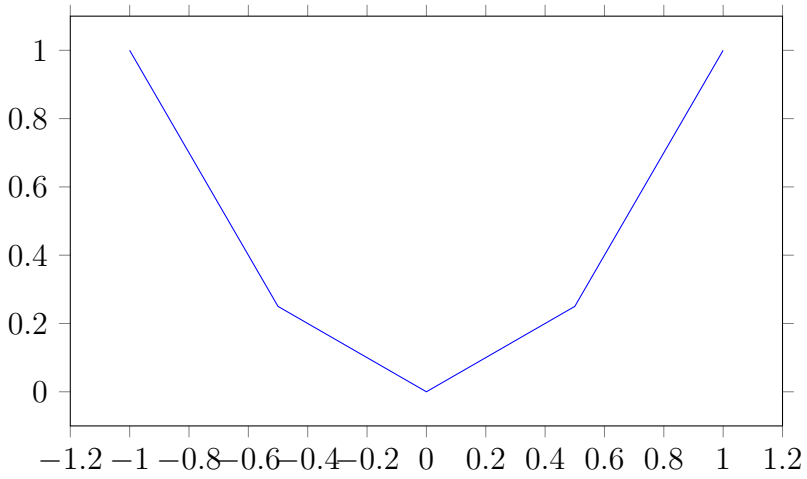



图 2.2: 没有退化点的凸包络

**命题 5.** *$n$ 维欧式空间有点集 $p_i$, 那么 $p_i$ 的 Delaunay 剖分等价于 $p_i$ 配有数值 $f_i = \frac{|p_i|^2}{2}$ 的凸包络。*

证明: 由凸函数的命题1知道，定义域每个开覆盖上面是凸函数那么整体一定是凸函数。也就是对于网格相邻的两个 $n$ 维单形 $x_0, ...x_n$ 和 $x_0...x_{n-1}, \overline{x}$, 其中点 $x_0...x_n$ 的顺序要满足 $\begin{vmatrix} 1 & x_0 \\ .. & .. \\ 1 & x_n \end{vmatrix} > 0$。映射 $f(x) = \frac{|x|^2}{2}$, 那么 $y_i = (x_i, f(x_i))$ 是凸包络等价于不等式 $\begin{vmatrix} 1 & y_0 \\ .. & .. \\ 1 & y_n \\ 1 & \overline{y} \end{vmatrix} = \begin{vmatrix} 1 & x_0 & f(x_0) \\ .. & .. & .. \\ 1 & x_n & f(x_n) \\ 1 & \overline{x} & f(\overline{x}) \end{vmatrix} > 0$, 可以证明 $x_i$ 点的整体平移不影响上述行列式的值。

接下来的变形会证明凸包和 Delaunay 的等价性。

$$\begin{vmatrix} 1 & x_0 & \frac{x_0^2}{2} \\ 0 & x_1-x_0 & \frac{x_1^2-x_0^2}{2} \\ 0 & .. & .. \\ 0 & x_i-x_0 & \frac{x_i^2-x_0^2}{2} \\ 0 & .. & .. \\ 0 & \overline{x}-x_0 & \frac{\overline{x}^2-x_0^2}{2} \end{vmatrix} = \begin{vmatrix} x_1-x_0 & \frac{x_1^2-x_0^2}{2} \\ .. & .. \\ x_i-x_0 & \frac{x_i^2-x_0^2}{2} \\ .. & .. \\ \overline{x}-x_0 & \frac{\overline{x}^2-x_0^2}{2} \end{vmatrix} = \begin{vmatrix} A & b \\ \overline{x}-x_0 & \frac{\overline{x}^2-x_0^2}{2} \end{vmatrix} = |A| \begin{vmatrix} I & A^{-1}b \\ \overline{x}-x_0 & \frac{\overline{x}^2-x_0^2}{2} \end{vmatrix}$$

因为 |A|>0。

$$\begin{vmatrix} I & A^{-1}b \\ 0 & \frac{\overline{x}^2-x_0^2}{2}-(\overline{x}-x_0)A^{-1}b \end{vmatrix} = \frac{\overline{x}^2-x_0^2}{2}-(\overline{x}-x_0)A^{-1}b > 0$$

可以证明 $A^{-1}b$ 是单形 $x_0...x_n$ 外接圆的圆心 O，那么 $(\overline{x}-O)^2 > (x_0-O)^2$ 也就是 $\frac{\overline{x}^2-x_0^2}{2}-(\overline{x}-x_0)A^{-1}b>0$。 □

**命题 6.** *二维 Delaunay 剖分的凸调整等价翻转边操作。*

命题的详细证明比较繁杂，现简述如下：

证明: 这里我们用反证法。如果凸调整有其它方式，必定有退化的点。然而 Delaunay 剖分等价于 $\left(p_i, \frac{|p_i|^2}{2}\right)$ 的凸包，由命题4知道 Delaunay 剖分没有退化的点。 □

**定义 6** (凸包络). *借助上面的偏序关系可以重新定义等价的凸包络：网格区域上的凸函数 $f$(不依赖网格剖分的定义) 称为凸包络，如果满足条件 $f(p_i) \le f_i$ 且对于满足这个条件的任何函数 $g$，有 $f \ge g$。*

现在记 $Gram(p_1,p_2,...,p_n) = A^TA, A = (p_1,p_2,...,p_n)$。

**命题 7.** *$n$ 维单形由 $P_0,...P_n$ 点组成，则它的的体积是*

$$\frac{1}{n!}\sqrt{|Gram(P_1-P_0,P_2-P_0,...,P_n-P_0)|}$$

。

证明: 积分区域 $\Omega = \{\sum_{i=0}\lambda_iP_i : \lambda_i > 0, \sum\lambda_i = 1\}$ 则有积分 $\int_\Omega dx_1 \wedge ...dx_n = \int_{\Omega_1}\frac{\partial x}{\partial\lambda}d\lambda_1 \wedge ..\lambda_n$ 其中 $\Omega_1 = \{\lambda_1...\lambda_n : \sum_{i=1}\lambda_i \le 1\lambda_i > 0\}$,$\frac{\partial x}{\partial\lambda}$ 是雅可比矩阵行列式，其值等于 $\sqrt{|Gram(P_1-P_0,P_2-P_0,...,P_n-P_0)|}$。

而 $\int_{\Omega_1} d\lambda_1 \wedge ..\lambda_n = \int_0^1 d\lambda_1 \int_0^{1-\lambda_1} d\lambda_2 ... \int_0^{1-\lambda_1...-\lambda_{n-1}} d\lambda_n = \frac{1}{n!}$。 □

**命题 8.** 对于任意维背景空间，有 $m$ 个向量 $P_1, ...P_m$，则它张成的体积 $|dP_1 \wedge ...dP_m|$ 为 $\sqrt{AA^T}$, 其中 $A = (P_1, ..P_m)$。

## 2.2 Brenier 理论

Brenier 证明说，凸区域上代价函数为欧式距离平方的最优映射，一定是其上某个凸函数的梯度映射。虽然 Brenier 的结论源自泛函，但是它和 Alexandrov 的几何定理有深刻的联系。

在 Villani 的《Topics in Optimal Transportation》一书中详细介绍了 Brenier 定理。Kantorovich 问题是这样的最优化问题：有测度空间 $(X,\mu),(Y,\upsilon)$ 和代价函数 $c(x,y)$，求概率分布 $\pi$，其中

$$\pi\in\Pi:\{\pi[A\times Y]=\mu[A],\pi[X\times B]=\upsilon[B],A\in X,B\in Y\}$$

使得下式值最小

$$\int_{X\times Y}c(x,y)\,d\pi(x,y)$$

**定理 2.2.1.** [5] *对于 Kantorovich 问题，它等价于寻找一个费用 $\varphi(x)+\psi(y)$ 使得总费用 $\int_{X\times Y}\varphi(x)+\psi(y)\,d\pi(x,y)$ 最大，并且这个费用满足 $J(\varphi,\psi)=\varphi(x)+\psi(y)\leq c(x,y)$。*

Villani 的证明简述如下。

证明: 对 $\inf\limits_{\pi\in\Pi}\int_{X\times Y}c(x,y)\,d\pi(x,y)$ 用类似 Lagrange 乘子法可以验证上式等于 $\inf\limits_{\pi}\sup\limits_{\varphi,\psi}\{\int_{X\times Y}c(x,y)\,d\pi(x,y)+\int_{X\times Y}\varphi dx+\int_{X\times Y}\psi dy-\int_{X\times Y}(\varphi+\psi)\,d\pi(x,y)\}$。

由极大极小原则 [5]，交换 inf 和 sup 得到 $\inf\limits_{\pi\in\Pi}\int_{X\times Y}c(x,y)\,d\pi(x,y)=\sup\limits_{\varphi,\psi}\inf\limits_{\pi}\{c(x,y)+\int\varphi dx+\int\psi dy-\int(\varphi+\psi)\,d\pi(x,y)\}$，由于 inf 作用，若 $\varphi(x)+\psi(y)\neq c(x,y)$，则 $\pi(x,y)=0$，故而上式化为 $\sup\limits_{\varphi,\psi}\int_{X\times Y}\varphi dx+\int_{X\times Y}\psi dy$[3]。 □

在下面的记号中 $\int$ 的积分区域默认为 $X\times Y$。

在一般线性规划问题中:

$$\min_x f(x),h_j(x)=0$$

加入 Lagrange 乘子项 $\max\limits_{\alpha_i}\sum\alpha_i h_i$，得到 $\min\limits_x\{f(x)+\max\limits_{\alpha_i}\sum\alpha_i h_i\}=\min\limits_x f(x),h_j=0$。那么上面定理的证明中 $\int\varphi dx+\int\psi dy-\int(\varphi+\psi)\,d\pi(x,y)$ 的由来类似

$$\max_{\varphi\psi}\sum(\pi[X\times B]-\upsilon[B])\,\psi(B)+\sum(\pi[A\times Y]-\mu[A])\,\varphi(A)$$

记 $f^c(x)=\inf\limits_y c(x,y)-f(y)$，那么有下面的命题:

**命题 9.** [5] *对于 Kantorovich 对偶中的 $\varphi,\psi$，它们满足 $\varphi=\phi^c,\psi=\phi^{cc}$。*

[3]证明过程告诉我们传输方案中有很多点的测度是 0

接下来的代价函数我们考虑的是欧式距离,$(X,\mu),(Y,\upsilon)$ 是在 $R^n$ 的测度空间，那么 Kantorovich 问题变成

$\underset{\pi\in\Pi}{inf}\int\frac{(x-y)^2}{2}d\pi(x,y)=\int\frac{x^2}{2}dx+\int\frac{y^2}{2}-\underset{xy}{sup}\int xyd\pi(x,y)$, 令 $M_2=\int\frac{x^2}{2}dx+\int\frac{y^2}{2}dy$。它的对偶是 $\underset{xy}{sup}\int\varphi dx+\int\psi dy,\varphi(x)+\psi(y)\leq\frac{(x-y)^2}{2}$, 我们让 $\tilde{\varphi}(x)=\frac{x^2}{2}-\varphi,\tilde{\psi}(y)=\frac{y^2}{2}-\psi$, 那么 $J(\varphi,\psi)=M_2-J\left(\tilde{\varphi},\tilde{\psi}\right)=\underset{\pi\in\Pi}{inf}\int\frac{(x-y)^2}{2}d\pi(x,y),\tilde{\varphi}+\tilde{\psi}\geq c(x,y)$。

那么 $J\left(\tilde{\varphi},\tilde{\psi}\right)=\underset{x,y}{sup}\int xyd\pi(x,y)$ 应用上面的命题9, 有 $\tilde{\varphi}=\phi^*,\tilde{\psi}=\phi^{**}$, 我们可以把这两个函数简写为 $\varphi,\varphi^*$。

**命题 10.** *[5]* 对于上面的 $\varphi,\varphi^*$,$xy=\varphi(x)+\varphi^*(y)\iff y\in\partial\varphi(x)$[4] $\iff x\in\partial\varphi^*(y)$。

证明: 因为 $xy\leq\varphi(x)+\varphi^*(y)$, 所以 $xy\geq\varphi(x)+\varphi^*(y)\iff\forall z\in R^n,xy\geq\varphi(x)+yz-\varphi(z)\iff\varphi(z)-\varphi(x)\geq(y,z-x)\iff y\in\partial\varphi(x)$。 □

**定理 2.2.2** (Brenier's Theorem). *[20]* 上述最优传输方案 $y(x)$ 是一个凸函数的梯度映射。

**命题 11.** 凸函数 $\varphi$ 的梯度映射 $\nabla\varphi$ 的逆映射是 $\nabla(\varphi^*)$。

证明: $\nabla\varphi^*(x)=\nabla\left(\underset{y}{sup}(xy-\varphi(y))\right)=y$, 其中 $\nabla\varphi(y)=x$。所以若令 $x=\nabla\varphi(z)$ 替换 $x$，上式写作下面: $\nabla\varphi^*(\nabla\varphi(z))=y$, $\nabla\varphi(y)=\nabla\varphi(z)$, 由引理2得到 $y=z$。□

[4]次微分

## 2.3 Alexandrov 理论

**定义 7** (Minkowski 问题). [4] 在 3 维空间给定 $m$ 个单位向量 $n_i$(要限制这些向量不位于某个平面的一侧), 那么任给 $m$ 个正数 $A_i, \sum A_i n_i = 0$, 是否存在唯一的凸包, 使得这个凸包的多边形以 $n_i$ 作为向外的法向量, $A_i$ 作为面积。

Minkowski 问题的形式可以推广到高维。

Alexandrov 推广了导师 Minkowski 的问题, 它允许把凸包换成凸包络（分片线性), 允许曲面向外无限延伸, 或者有边界。

**定义 8** (Alexandrov 问题). [4] 在 3 维空间给定 $m$ 个单位向量 $n_i$(要限制这些向量位于某个平面的一侧), 和一个凸区域 $\Omega$, 那么任给 $m$ 个正数 $A_i > 0$, 是否存在唯一的凸包络:

$$\max_i < x, p_i > + h_i$$

这个凸包络以 $p_i$ 为法向量的胞腔向 $\Omega$ 区域的投影面积是 $A_i$。

Alexandrov 问题的形式也可以推广到高维。

下面的证明介绍主要基于 Alexandrov 的《Convex Polyhedra》。在 Alexandrov 的《Convex Polyhedra》一书的第七章不仅给出了 3 维凸包的存在性定理, 还推广到任意维。第十一章介绍了任意维的刚性定理。

**定理 2.3.1** (Minkowski Uniqueness Theorem). 在 $R^n$ 空间, 若有两个凸多面体 $P_1, P_2$, 保持其每个面的面积相等, 法向量相等, 则 $P_1 = P_2$。

**定理 2.3.2** (Alexandrov Uniqueness Theorem). 在 $R^n$ 空间, 若有两个曲面（分片线性, 可能无限延伸, 可能有边界）$P_1, P_2$, $k$ 个平面, 它们围成一个闭或开的凸多面体。保持 $k$ 个平面每个面的面积相等, 法向量相等, 则 $P_1 = P_2$。

Alexandrov 关于 Minkowski 的存在性定理的证明思路: 基于域不变定理证明映射即开又闭从而得到映射 $\varphi$ 是满射。

Minkowski 的证明方法更加简单且可以推广到任意维。

首先介绍 Lagrange 乘子法, 它的作用是判断原函数在限制条件下的极值（不是最值）满足的一阶微分条件。这句话首先注意两点。

- 它首先要求原函数在定义域内一阶可导。
- 其次它是判断原函数在限制条件下极值点满足的一阶微分条件。比如定义域为 $[0, 1]$ 的函数 $f$, 在某限制条件下存在最大值, 而这个最大值是在定义域的边界 0 处取得的, 那么它不是极大值, Lagrange 乘子法得到的一阶微分条件不适用它。

设 $Q_i$ 是以 $n_i$ 作为法线的超平面, 那么 $Q_i$ 上存在一点 $c_i$ 到原点的距离最短, 并且有 $c_i = h_i n_i$。可以知道当 $n_i$ 固定时 $h_i$ 唯一确定 $Q_i$。其实 $h_i$ 就是原点到平面 $Q_i$ 的带符号的距离。

那么以 $Q_i$ 作为边界且以 $n_i$ 作为向外法向量的凸包（若存在）记为 $P$, 那么 $P$ 是以 $h_i$ 作为变量。记 $Volume\,(P)$ 是 P 的体积,$F_i$ 为 $Q_i$ 和 $P$ 相交区域的面积。

**命题 12.** $\frac{\partial Volume(P)}{\partial h_i} = F_i$。

证明: 当 $h_i$ 逐步减少, 那么 $Q_i$ 扫过 $P$。那么假设当 $h_i$ 减少到 $h_i^0$ 时, $Q_i$ 刚好扫过 $P$(也就是 $F_i\,(h_i^0) = 0$)。

对区间 $[h_i^0, h_i]$ 进行划分 $h_i^0, h_i^1, ...h_i^n, h_i$, 那么其实 $Volume\,(P)$ 就等于 $\lim\limits_{|T|\to 0} \sum\limits_j \Delta h_i^j *$ $F_i\left(h_i^j\right), \Delta h_i^j = h_i^{j+1} - h_i^j$,$|T|$ 表示划分的间隔的最大值。所以有 $\Delta Volume\,(P) = \Delta h_i F_i$。

□

为了后面应用 Lagrange 乘子法, 我们需要限制 $h_i$ 的定义域, 设 $\Omega = \{h_i : F_i\,(h_i) > 0\}$。

这是因为如果不限制 $h_i$ 在 $\Omega$ 的话, 那么 $h_i$ 在整个欧式空间可导, 就是导数满足上面命题12的条件, 那么就存在 $Volume\,(P)$ 为负的情况。

比如下面的情况:

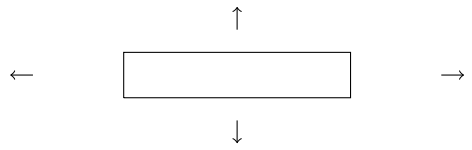
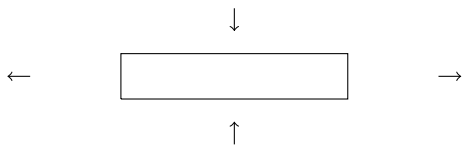


图 2.3: 面积为正

箭头代表直线 (超平面) 的方向, 当上面的直线向下运动, 下面的直线向上运动, 然后会变成下图。



这时凸包的按照可导的条件应该为负, 所以我们要定义凸包体积为负的情况。为了避免这种情况, 我们限制定义域为 $\Omega$。

**命题 13.** $Volume\,(P)$ 的 $h_i$ 在 $\Omega$ 上的的取值等于在 $[0,+\infty)\oplus[0,+\infty)\oplus..\oplus[0,+\infty)$ 的取值。

证明: 可以将 $P$ 整体平移, 使得 $P$ 包含原点, 所以 $h_i \geq 0$, 而 $Volume\,(P)$ 不变。□

如果添加限制条件 $h_i A_i = 0$, 那么上述命题依然成立:

**命题 14.** $Volume\,(P)$ 的 $h_i$ 在 $\Omega, h_i A_i = 1$ 上的取值等于在 $\Omega, h_i A_i = 1, h_i \geq 0$ 上的取值。证明用到了 $\sum A_i n_i = 0$ 的条件。

**命题 15.** $Volume\,(P)\,, h_i \in \Omega$ 在限制条件 $h_i A_i = 1$ 下必存在上确界，且在 $\Omega$ 上一定能取到。所以它是内点，自然也就是极值点。

证明: 由命题14知道,$Volume\,(P)\,, h_i \in \Omega$ 在限制条件 $h_i A_i = 1$ 下的取值相当于在有界集的取值，所以必定存在上确界，只需证明它的极值点不在 $\Omega$ 的边界上。当 $F_i = 0$ 时，设 $F_j$ 是与其相邻区域面积，那么可以证明 $\overline{h_i} = h_i - \frac{\epsilon}{A_i}, \overline{h_j} = h_j + \frac{\epsilon}{A_j}$ 仍然满足限制条件，且当 $\epsilon$ 足够小时，$Volume\,(P)$ 是增大的。 □

**定理 2.3.3.** （*Minkowski* 存在定理），对于 *Minkowski* 问题，存在（且唯一）凸包是问题的解。

证明: $\max\limits_{h_i} Volume\,(P)\,, h_i \in \Omega$ 在限制条件 $h_i A_i = 1$ 下根据 Lagrange 乘子法得到 $\max\limits_{h_i} (Volume\,(P) - \lambda\,(\sum h_i A_i - 1))\,, h_i \in \Omega$, 得到极大点满足的一阶微分条件是 $\frac{\partial Volume(P)}{\partial h_i} = F_i = \lambda A_i$, 也就是存在凸包它满足 $F_i = \lambda A_i$, 对这个凸包进行缩放就得到 Minkowski 存在定理。 □

不论是 Minkowski 问题还是 Alexandrov 问题，凸包和凸包络都唯一由每个超平面的截距 $h_i$ 确定，每当我们确定一组 $h_i$ 的值，那么凸包或者凸包络确定，它上每个面的面积也确定，这样构成一个自然映射 $\varphi : h_i \to A_i$, 其中 $A_i$ 为凸包上法向量为 $n_i$ 的的面积。刚性定理和存在性定理分别保证映射 $\varphi$ 是单的，满的。

## 2.4 基于变分原理的构造性证明

丘成桐团队在《Variational Principles for Minkowski Type Problems, Discrete Optimal Transport, and Discrete Monge-Ampere Equations》[7] 一文通过构造凸能量的方式利用变分原理给出了 Alenxandrov 定理的构造性证明，这是也本文迭代方法的核心来源。Aurenhammer,Hoffmann 和 Aronov 在 90 年代利用 Power Voronoi Diagram 给出的构造性证明，Levy 在 2014 年通过构造凸能量给出了构造性证明。鉴于水平有限，上文细节不能一一介绍。

给定一些梯度固定的超平面，它们被圆柱面截取。

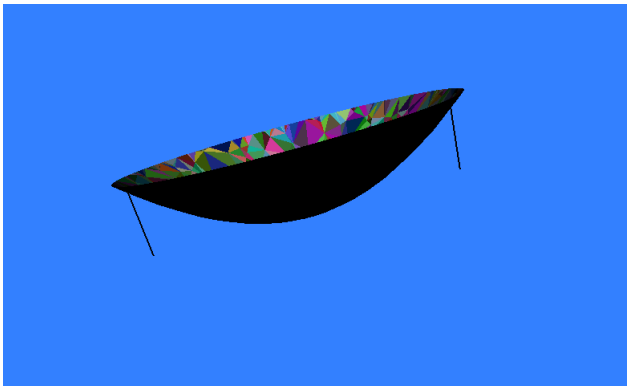


图 2.4: Alexandrov 问题

如上图的包络沿边界向下作柱面，并和 $xy$ 平面共同包围的体积当作能量 $G$。

还是上面的 Alexandrov 问题，这个上包络是由每个超平面的截距 $h_i$ 确定的。超平面截距的定义是 $h_i : f_i = (x, p) + h_i$。

那么 $G$ 是 $h_i$ 的函数，记 $A_i$ 为超平面 $f_i$ 在 $\Omega$ 区域 (这里 $\Omega$ 区域为单位圆) 的投影面积。以下积分区域默认为 $\Omega$。

**引理 5.** $\frac{\partial G}{\partial h_i} = A_i$。

证明: 假设面积为 $A_i$ 的区域如下:

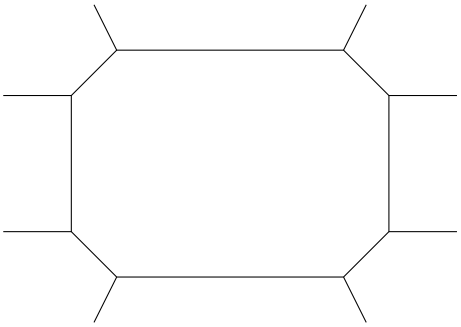


图 2.5: 投影的胞腔

那么 $\partial G = \partial \int f(x)\, dA(x) = \int (\partial f * dA + f * \partial dA)$, 其中 $\int (f * \partial dA) = 0$, 因为局部的投影面积总是不变的。

所以 $\partial G = \int (\partial f * dA) = A_i * \partial f_i = A_i * \partial h_i$。

□

上述结论对任意维 Alexandrov 问题都成立，证明方法几乎逐字不变。

现在记 $p_i$ 为超平面 $f_i$ 的梯度，$A_i \bigcap A_j$ 是胞腔 $A_i$ 和 $A_j$ 的相交区域。

**引理 6.** $\frac{\partial A_i}{\partial h_j} = -\frac{A_i \bigcap A_j}{|p_i - p_j|}, j \neq i$。

证明: 采用微元的思想。$f_i = (x, p_i) + h_i, f_j = (x, p_j) + h_j$, 那么 $A_i \bigcap A_j$ 区域满足方程 $(x, p_i - p_j) = h_j - h_i$, 也就是 $A_i \bigcap A_j$ 随 $h_j$ 平移变化。那么 $A_i(h_j + \theta) - A_i(h_j) = (A_i \bigcap A_j + \epsilon) * l$,l 是 $A_i \bigcap A_j$ 沿 $p_j - p_i$ 方向移动的距离, 所以 $l = -\frac{\theta}{|p_j - p_i|}$。其中 $\epsilon \to 0$ 当 $\theta \to 0$。代入 $\lim_{\theta \to 0} \frac{A_i(h_j+\theta) - A_i(h_j)}{\theta} = -\frac{A_i \bigcap A_j}{|p_i - p_j|}$。 □

**命题 16.** $\frac{\partial A_i}{\partial h_i} = -\sum_{j \neq i} \frac{A_i}{h_j}$。

证明: 这是由总面积的不变性质得到的。 □

**命题 17.** 能量 $G$ 是凸能量。

证明: 只需证明 Hessian 矩阵 $\frac{\partial^2 G}{\partial^2 h_i h_j}$ 是半正定矩阵。由上面的讨论知道它是主对角占优的对称矩阵，所以它只有 0 特征向量 $(1, 1, 1..1)$。 □

上面的结论都在 $n$ 维 Alexandrov 问题成立，实际上这些证明没有限制维度。如果固定某个 $h_0$ 不变，也就是说 $G$ 是 $(h_i, i \neq 0)$ 的函数，那么 $G$ 就是严格凸函数。

这些结论在算法部分都会起到作用。

**定理 2.4.1** (Alexandrov 定理的构造性证明)**.** 给定一些正数 $W_i$, 满足 $\sum W_i$ 等于 $\Omega$ 区域面积。那么使凸能量 $\overline{G} = G - \sum W_i h_i$ 的最小的凸包络，也就是 *Alexandrov* 问题要求的凸函数。

证明: 因为 $\overline{G}$ 的极小点是满足 $A_i = W_i$ 的 $h_i$, 这等同于 Alexandrov 问题。固定一个 $h_0$, 那么由上面知道能量是凸的，因为 Minkowski 存在性定理，所以极小点必定不是无穷远点, 得证。 □

# 第三章　算法

## 3.1　反对称张量和高维凸包算法

反对称张量的定义可以参考文献 [21]，它的性质在文献 [22] 中有详细的介绍。

引入反对称张量前，凸包算法必须在 $n$ 维背景空间或者想办法转移到 $n$ 维背景空间中计算它的边界 ($n-1$ 维流形)。

在没有引入反对称张量时，我们最多可以定义 $n$ 维空间 $n-1$ 个向量的外积向量，单形有向面积通过行列式定义，这些都要求在 $n$ 维背景空间中计算 (或者 $n+1$ 维背景空间)。

利用反对称张量定义线性空间的方向具有统一性，并且兼容之前利用叉积和行列式定义方向。

**定义 9** (线性相关)**.** 向量 $p_0, ..p_m$ 线性相关当且仅当 $p_0 \wedge p_1 \wedge .. \wedge p_m = 0$。

**定义 10** (同向, 反向)**.** 对于两个 $m$ 阶反对称张量 $p$ 和 $q$，称其同向，当且仅当 $kp = q, k \geq 0$, 称其反向，当且仅当 $kp = q, k \leq 0$。

在计算 $n$ 维凸包时，时常要判断叉积的方向，或要要定义有向面积。这些定义总是限制背景空间，有了定义方向的反对称张量的工具后就可以摆脱这种限制。

**定义 11** (模长)**.** 给定线性空间一组基 $e_i$, 那么 $m$ 阶反对称张量可以表示成 $c = \sum_i f_i e_l \wedge ... \wedge e_k$, 那么称 $\sqrt{\sum f_i^2}$ 是 $c$ 在基 $e_i$ 下的模长。其中 $e_l \wedge ... \wedge e_k$ 线性无关 (这里的线性无关说的是 $e_l \wedge ... \wedge e_k$ 不能相互表示)。

**命题 18.** 形似 $(\sum a_i e_i) \wedge (\sum b_i e_i)$ 的二阶反对称张量的模长在正交基变换下是不变量。

证明:　只需证明 $\sum a_i e_i \wedge \sum b_i e_i$ 的模长等于 1, 其中 $a_i$ 和 $b_i$ 是正规正交向量。也就是要证明 $|\sum a_i e_i \wedge \sum b_i e_i| = \sum (a_i b_j - a_j b_i)^2 = 1$, 根据条件有 $\sum a_i^2 = 1, \sum b_i^2 = 1, \sum a_i b_i = 0$, 可以验证 $\sum (a_i b_j - a_j b_i)^2 = (\sum a_i^2)(\sum b_i^2) - (\sum a_i b_i)^2 = 1$, 得证。这个命题在 1 阶张量是显然的，在 $n-1$ 阶张量也成立。

□

**命题 19.** 形似 $p_1 \wedge p_2 ... \wedge p_{n-1}$ 的 *n-1* 阶反对称张量的模长在正交基变换下是不

变量 *(*其中 $p_i$ 是 $n$ 维向量空间的向量*)*。

证明: 同理不妨假设 $p_i$ 正规正交，因为反对称张量就是天然的施米特正交操作。可以证明 $p_1 \wedge p_2 \wedge ... \wedge p_{n-1} \wedge q$ 的模长为矩阵 $(p_1, ...p_{n-1}, q)$ 的行列式的绝对值。假设 $p_1 \wedge p_2 ... \wedge p_{n-1} = \sum l_i e_1 \wedge ... \wedge \overline{e_i} .. \wedge e_n$, 那么 $p_1 \wedge p_2 \wedge .. \wedge p_{n-1} \wedge q$ 模长等于 $\sum l_i q_i$ 的绝对值，而当 $q = p_i$ 时，模长为 0, 也就是 $l_i$ 作为向量和 $p_i$ 正交，所以当 $q = \sum l_i e_i$ 时，矩阵 $(p_1, ...p_{n-1}, q)$ 行列式为 $\pm 1$, 也就是向量 $l_i e_i$ 的模长为 1。 □

如果可以证明形似 $p_0 \wedge ... \wedge p_m$ 的 m 阶反对称张量的模长在正交基变换下是不变量（其中 $p_i$ 是向量），那么我们可以利用反对称张量忽略背景空间计算向量张成空间的体积，因为它的推论是 $|p_0 \wedge ... \wedge p_m|^2 = |Gram\,(p_0, ...p_m)\,|$, 其中 $p_i$ 是向量。

### 3.1.1 凸包算法的实现

目的: 我们要计算 $n$ 维欧式子空间中的点集 $p_i$ 形成的凸包。

**定义 12.** *给定一些点集的顺序是 $p_0, ...p_m$，那么我们称这个顺序决定的反对称张量是指 $(p_1 - p_0) \wedge (p_2 - p_0) \wedge ... \wedge (p_m - p_0)$。*

对于一个 $n$ 维单形它的点的顺序是 $p_0, ...p_n$, 那么由上面的定义它对应一个反对称张量。那么它的边界也是由 $n-1$ 维单形组成的, 那么这些单形的点顺序如何定义（等价于张量如何定义)。

**定义 13.** *因为反对称张量计算是向后添加, 假设 n-1 维单形 $s$ 的点由 $p_0, ...\overline{p_i}, ...p_n$ 组成，那么如果 $(-1)^{n-i}$ 小于 0, 则 $s$ 的点顺序是 $p_1, p_0, p_2, ...\overline{p_i} ... p_n$(也就是交换 $p_0, p_1$ 的位置)，否则，$s$ 的点的顺序不变。*

其实这样的操作可以重复下去，我们可以得到直到 2 维单形的点顺序。这也告诉我们对于单形可以知道它所有结构信息。这在实际应用中非常重要。

上面这样的定义使得反对称张量计算符合向后添加的习惯，并且有利于以后各种量的计算。下面罗列几个重要步骤:

第一个步骤：因为凸包存在在 $n$ 维欧式子空间中，我们要首先定义这个空间的方向，以便于下面各个环节的操作。具体方法是: 在这个子空间找到 $n$ 个线性无关向量，并决定好它们的顺序 $a_0, ...a_{n-1}$，则反对称张量 $a_0 \wedge a_1 \wedge ... \wedge a_{n-1}$ 就是这个子空间的方向。

第二个步骤: 在点集 $p_i$ 选择 $n+1$ 个点, 并创建初始凸包。具体方法是这几个点由定义12定义的反对称张量和子空间的方向相反，则交换第一和第二个点的顺序。这样操作完成 n 维单形的点顺序定义 (或者反对称张量的定义)。由定义13, 我们可以得到它下一层单形的结构信息。

第三个步骤：使用增量凸包算法：对于给定凸包，添加一点 $q$, 计算新的凸包。具

体方法是: 遍历凸包所有的 $n-1$ 维单形，假设单形为 $s$, 它的点顺序是 $p_0, ...p_{n-1}$, 那么计算由点 $p_0, ...p_{n-1}, q$ 对应的反对称张量，如果它和子空间的方向相反，则删除这个单形。

在实际算法中我们采用的是广度优先搜索法：先找到凸包中使得 $p_0, ...p_{n-1}, q$ 对应的反对称张量和子空间的方向相反的单形，再以此单形为基础进行广度优先搜索。

第四个步骤: 最后最后这个凸包被删除了一些单形，形成带边流形，这个边界也是由单形组成，它们的点顺序结构也可以获得。那么依次遍历这些 $n-2$ 维边界单形 $k$，并创建 $n-1$ 维单形（形成点顺序是 $k, q$, 其中 $k$ 是 $n-2$ 维单形点顺序), 这样的操作相当于缝合带边流形是拓扑球。

**Algorithm 1:** Convex Hull

**Input:** 在 $n$ 维欧式子空间的 $p_i$ 点集。

**Output:** 点集的凸包。

1 **Initialize** 第一步：先定义子空间的方向 (反对称张量)。第二步：在 $p_i$ 选择 $n+1$ 个点, 并创建初始凸包。

2 **for** 选择未被添加的点 $p_i$ 进行增量凸包算法，直到所有点都被添加完。 **do**

3 第三步: 增量凸包：遍历凸包的每个 $n-1$ 维单形，每个单形的点是有顺序的，记为 s, 那么对 $n+1$ 个点 $(s, p_i)$ 计算它的反对称张量。如果反对称张量与子空间方向相反, 删除这个单形。

4 第四步：缝合带边流形为拓扑球。

5 **end**

下图是 3 维 Delaunay 剖分。

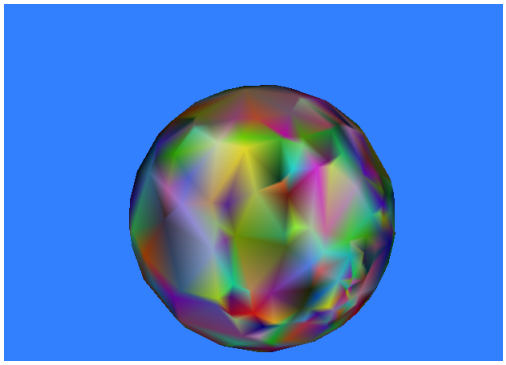


图 3.1: 3 维 Delaunay 剖分

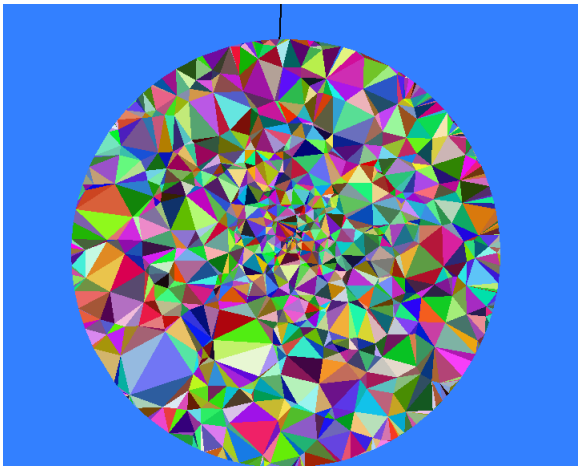


图 3.2: 2 维 Delaunay 剖分

### 3.1.2 高维凸包算法出现的问题，原因，以及解决办法

**问题:**

当点非常稠密时会造成数据结构问题。

**原因:**

经过分析和排查发现是由精度造成的，计算机精度有限，当点非常稠密，计算某些非常小的量会储存为 0, 使得凸包算法出现问题。

在图3.3展示了二维欧式空间凸包算法出现的问题:

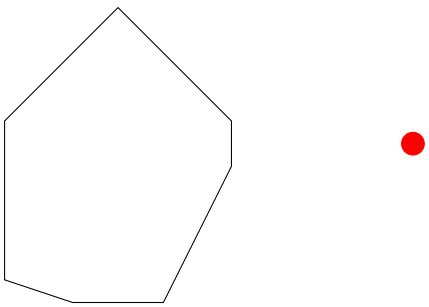


图 3.3: 新添加的红色点

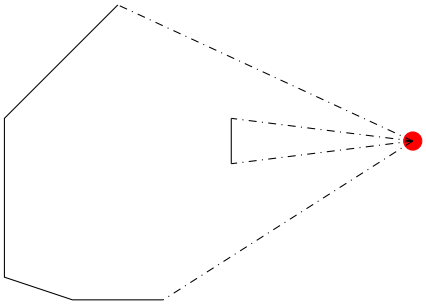


图 3.4: 由于精度问题只删除了两条边，虚线部分为即将创建的新边

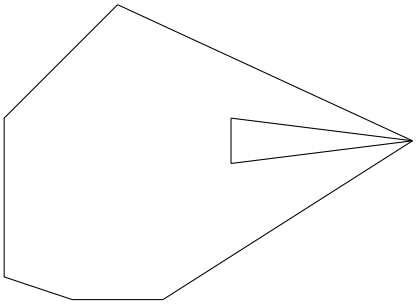


图 3.5: 结果造成网格的结构问题

上图以二维凸包为例，进行增量凸包计算时，由于精度问题会造成删除的单形（这里是边）不单连通，结果造成结构问题。

实际上当要删除的单形形成如下情况时都会造成数据结构问题。

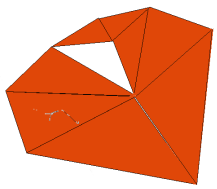


图 3.6: 不是流形

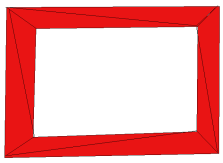


图 3.7: 不是单连通的流形

**定义 14** (单连通空间). [23] 拓扑空间 $X$ 中的任何封闭曲线和 $X$ 中的一点同伦等价。则称拓扑空间 $X$ 是单连通空间。

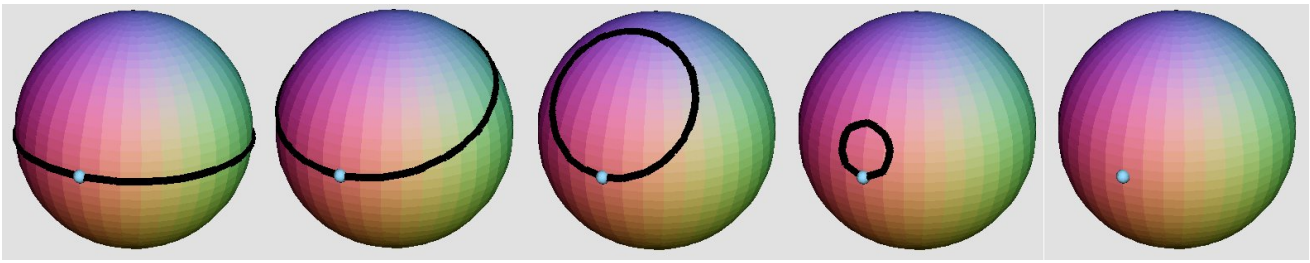


图 3.8: 单连通的球面 (引用自 [24])

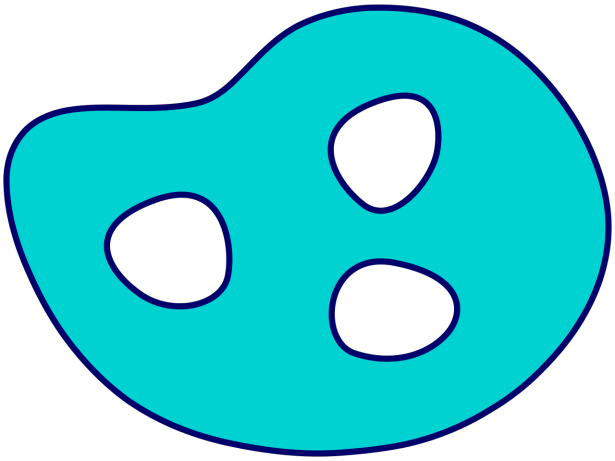


图 3.9: 非单连通区域 (引用自 [24])

**解决办法：**

针对上面的问题，有两种解决办法。

第一种：维持数据结构的稳定。因为本身计算机的精度计算就不是准确的结果，而且出现上面的问题，数据的精准性已经退而为次考虑的问题。我们可以保障在删除单形时维持流形的结构和单连通性。要判断流形是否单连通需要知道流形的边界数和欧拉示性数[23]，这些计算较为复杂。

第二种：可以引入高精度库。此方法属于工程方法。

由于第一中方法需要计算流形的边界和欧拉示性数，这些计算在高维情况非常复杂，故而采用第二种方法。

## 3.2 最优传输算法

**问题:** $n$ 维单位球内部的点集 $p_i$, 每个点配有测度 $w_i, \sum w_i = \frac{\pi^{\frac{n}{2}}}{\Gamma\left(\frac{n}{2}+1\right)}$, 求单位球区域 $\Omega$ 到这些点的最优传输，代价函数为欧式距离。

由 Brenier 和 Alexandrov 理论知道这个运输函数由 $\Omega$ 上的凸函数确定。这个凸函数是分片线性函数 $\max\limits_{h_i}\{(x,p_i)-h_i\}$，由 $h_i$ 唯一确定。现在给定 $\Omega$ 上的一个函数 $f_0 = \max\limits_{h_i}\{(x,p_i)-h_i^0\}$。

### 3.2.1 基础讨论

**命题 20.** $f_0^*$ 是点集 $p_i$ 上的凸包络。

这是在第一节关于共轭函数定义，它本身就是超平面集合的上包络。

**命题 21.** $f_0^*(p_i) = h_i^0$。

证明: 这是推论 2.1.1的结果。 □

**命题 22.** $f_0^{**} = f_0$。

证明: 这是引理4的结论。 □

这告诉我们我们要计算得到 $\Omega$ 上的一个函数 $f_0 = \max\limits_{h_i}\{(x,p_i)-h_i^0\}$，只需要计算 $p_i$ 点集上的凸函数，准确的说是凸包络。

所以 $\Omega$ 上的凸函数由 $f_i$ 唯一确定。这告诉我们可以调整点集 $p_i$ 上的数值 $f_i$ 获得最优传输映射。$\Omega$ 上每个胞腔的体积记为 $A_i$, 那么 $\frac{\partial A_i}{\partial f_j^*} = \frac{A_i \bigcap A_j}{|p_i - p_j|}$, 这是由命题16得到的，同理得到 $\frac{\partial A_i}{\partial f_i^*} = -\sum\limits_{i\neq j}\frac{A_i \bigcap A_j}{|p_i - p_j|}$。

我们采用牛顿法进行迭代。

牛顿法是利用切线或者切平面计算零点的迭代方法。如下图:

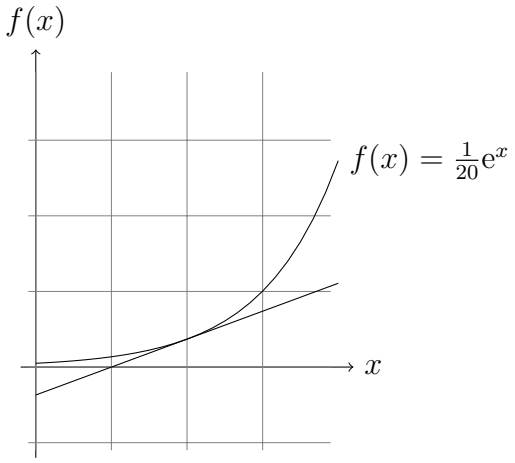



图 3.10: 牛顿法示意图

在图中切点 $x_0$ 处做切线, 与 x 轴交于 $\overline{x_0}$ 点,那么迭代式子为 $x_1 = (\overline{x_0} - x_0) * t + x_0$, 其中 t 为步长。

牛顿法也可以用来求凸函数的极值点，因为凸函数的一阶导 $\nabla \overline{G} = 0$ 的点必为极值点。$\nabla \overline{G} = \nabla^2 \overline{G}(x_0) * (x - x_0) + \nabla \overline{G}(x_0)$, 令上式等于 0, 得 $-\nabla \overline{G}(x_0) = \nabla^2 \overline{G}(x_0) * (x - x_0)$, 当 Hessian 矩阵正定时我们可以求得 $x - x_0$ 的值。现记 $s = x - x_0$，那么迭代公式就是 $x_1 = s * t + x_0$, 其中 $t$ 为步长。

### 3.2.2 算法讨论

我们要优化的能量是 $\overline{G} = G - W_i h_i$,$G$ 是变分原理构造性证明中上包络，高维圆柱面，$n$ 维欧式空间截取的体积（在二维最优算法中，高维圆柱面就是以 $z$ 为轴的圆柱面，$n$ 维欧式空间就是 $xy$ 平面）。

以下是几个重要步骤:

第一步：给定一组 $f_0^i$, 我们首先计算它的凸包络，但是这些分片线性函数的对偶点（梯度）可能落在 $\Omega$ 之外，所以我们要把这些单形删除。

对于这样的带边流形，我们再添加一个抽象的点，依次与带边流形的边界创建单形，形成同胚于 $n$ 维球面的流形。这个抽象的点在共轭函数中对应高维圆柱面。

第二步：对于上面的拓扑球面，由于每个点都可以代表一个曲面，比如上面的抽象点代表的是圆柱面，其它点 $p_i$ 代表的是超平面 $f_i^* = (x, p_i) - f_0^i$。这样，每个单形周围点所代表的曲面必定有个交点，这个交点就是这个单形对偶点。计算它。

第三步：对于每个点 $p_i$ 它对应于共轭函数的曲面或者胞腔（超平面与共轭函数的交），我们可以计算这些胞腔的面积。具体方法是：由于上步我们计算胞腔的关键点，利用凸包算法计算这些关键点的凸包。这个凸包结构由单形组成, 由命题7可以计算它们的面积，它们相加即为胞腔面积。

第四步：计算 Hessian 矩阵 $\frac{\partial^2 \overline{G}}{\partial^2 h_i h_j} = -\frac{\partial^2 \overline{G}}{\partial^2 f_i f_j}$。

第五步: 利用牛顿法对 $f_i$ 值迭代，因为可以固定一个 $f_0$ 不变，那么 Hessian 矩阵就是正定阵。可以利用 Cholesky 矩阵分解计算线性方程的解。

**Algorithm 2:** Optimal Transportation

**Input:** $p_i$ 点集，每个点的 Dirichlet 测度 $W_i$，阙值 $\epsilon$。

**Output:** 最优传输映射, 也等价于使凸能量 $\overline{G}$ 最小的凸包。

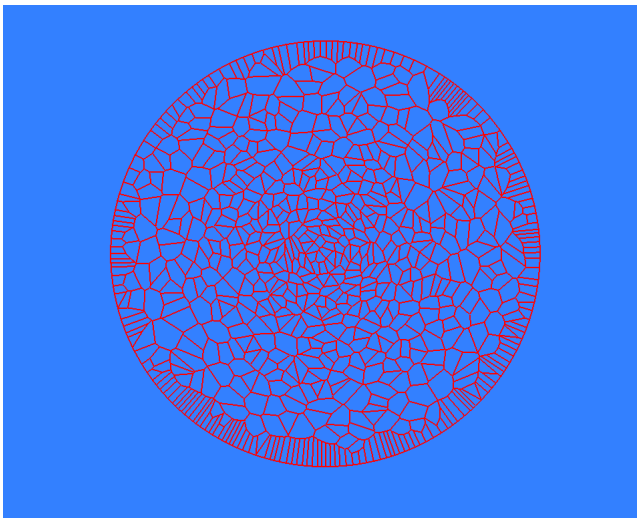
**Initialize** 在 $p_i$ 给定初始值 $f_i\frac{p_i^2}{2}$。

**for** 根据 $f_i$ 计算 $p_i$ 的凸包络，当能量的梯度的模长小于阙值 $\epsilon$ 为止。 **do**

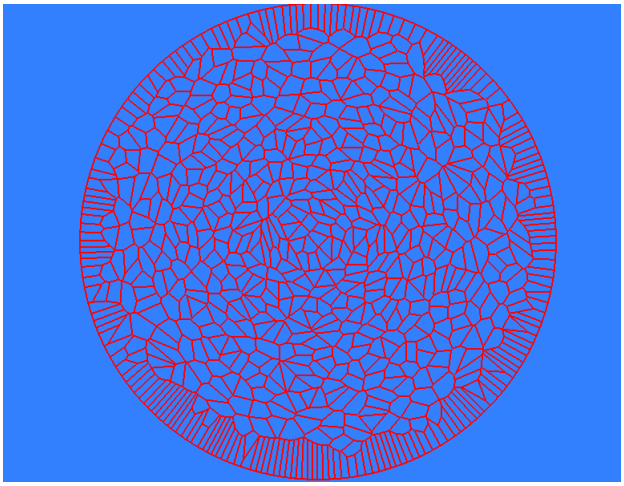
- 进行第一步，计算凸包络，并对其进行剪切。
- 进行第二步，加入抽象点，使包络成为拓扑球。计算单形的对偶点。
- 进行第三步，利用凸包算法计算 $\overline{G}$ 的一阶导，也就是共轭函数的胞腔面积。
- 进行第四步, 利用凸包算法计算 $\overline{G}$ 的二阶导，也就是 $\frac{\partial^2\overline{G}}{\partial^2 h_i h_j}$。
- 进行第五步，利用 Cholesky 矩阵分解和牛顿法进行迭代，并计算一阶导的模长和阙值 $\epsilon$ 比较。

**end**

### 3.2.3 实验结果

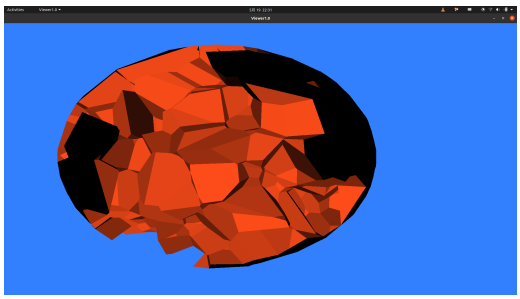


(a) 初始映射对应的胞腔

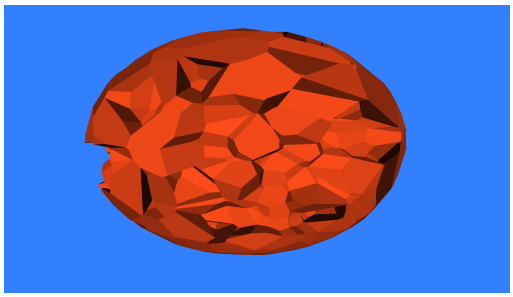


(b) 迭代后的映射对应的胞腔

图 3.11: 二维最优传输可视化



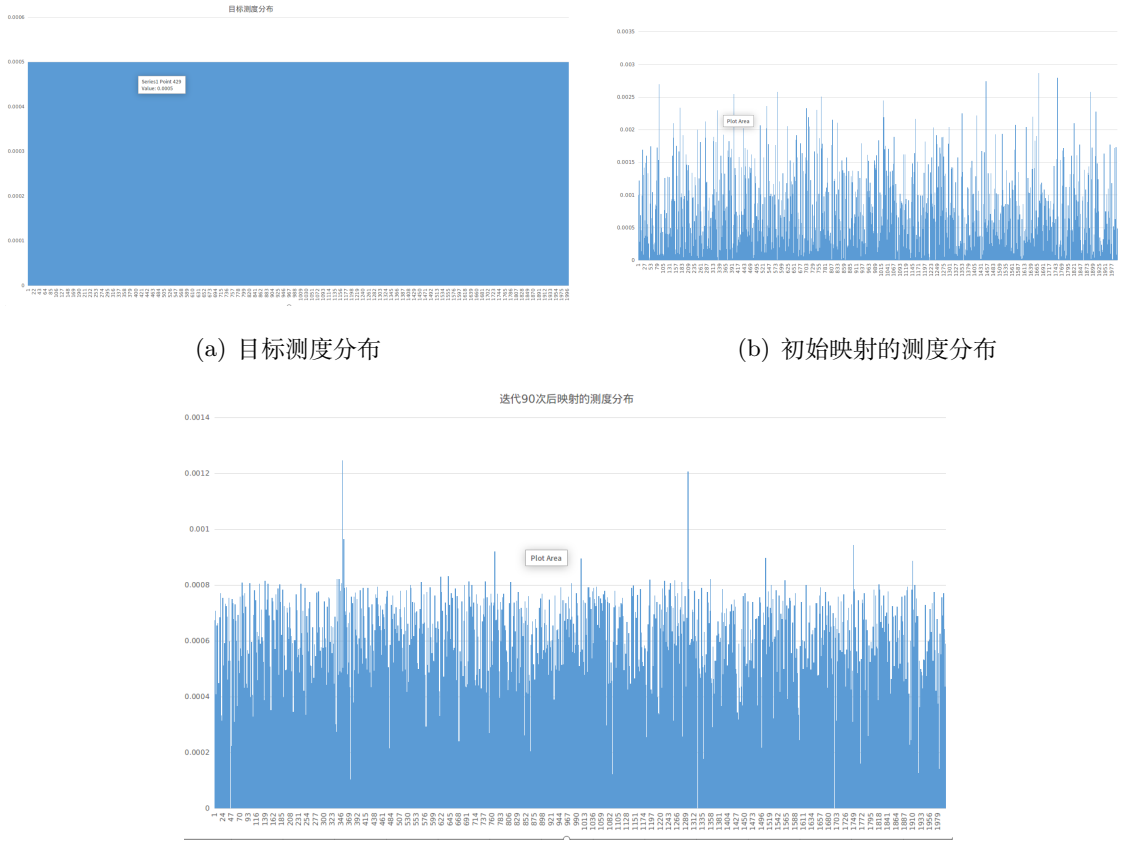
(a) 初始映射对应的胞腔



(b) 迭代后的映射对应的胞腔

图 3.12: 三维最优传输可视化

下面是 2000 个点三维最优传输测度分布情况。




(a) 目标测度分布

(b) 初始映射的测度分布

(c) 迭代 90 次后映射的测度分布

图 3.13: 三维最优传输测度分布情况

将结果的测度和目标测度相除，得到的比率的频率分布图，X 轴表示比率，Y 轴表示频率。所以预期的结果是频率集中在 1 附近:

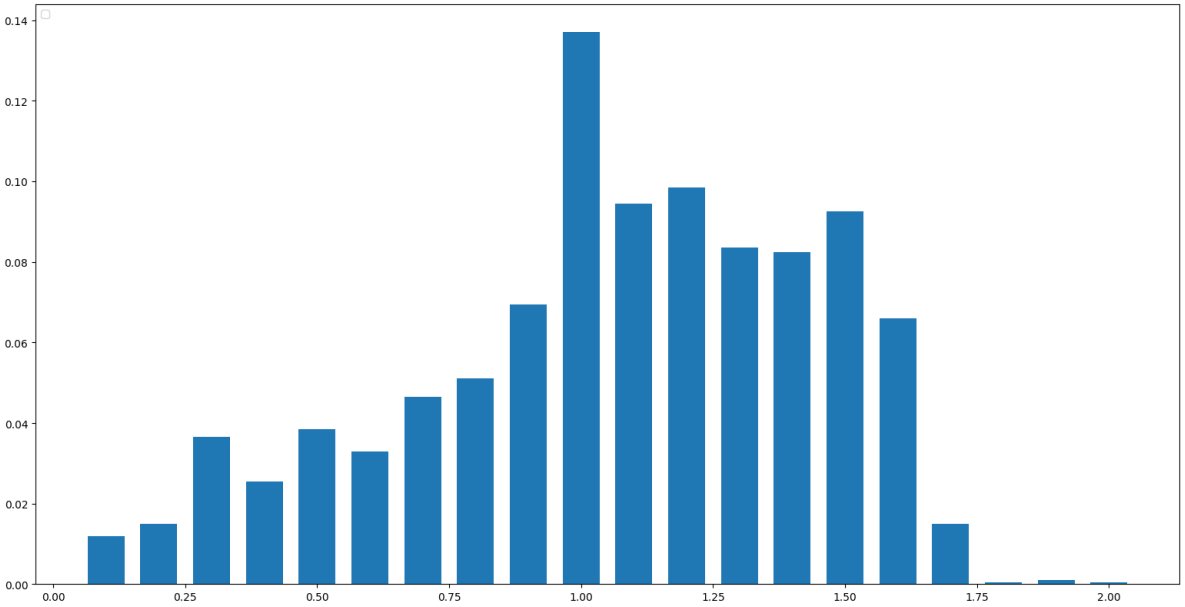
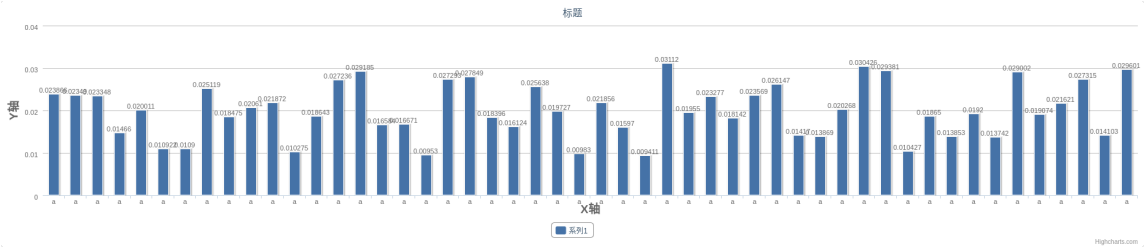
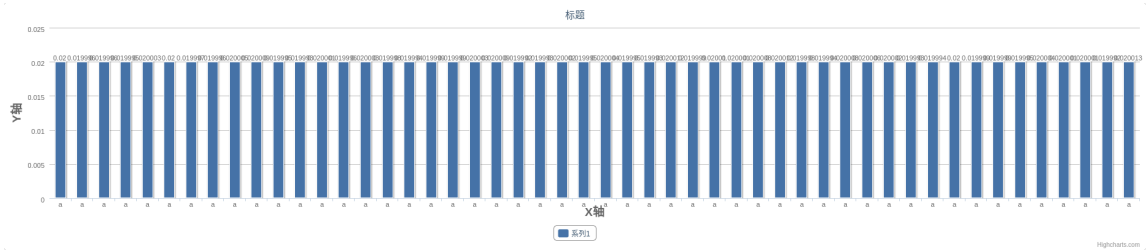



图 3.14: 三维最优传输测度比率的频率图

下面是四维最优传输测度分布情况。




(a) 初始映射的测度分布




(b) 迭代 90 次后映射的测度分布

图 3.15: 四维最优传输测度分布情况

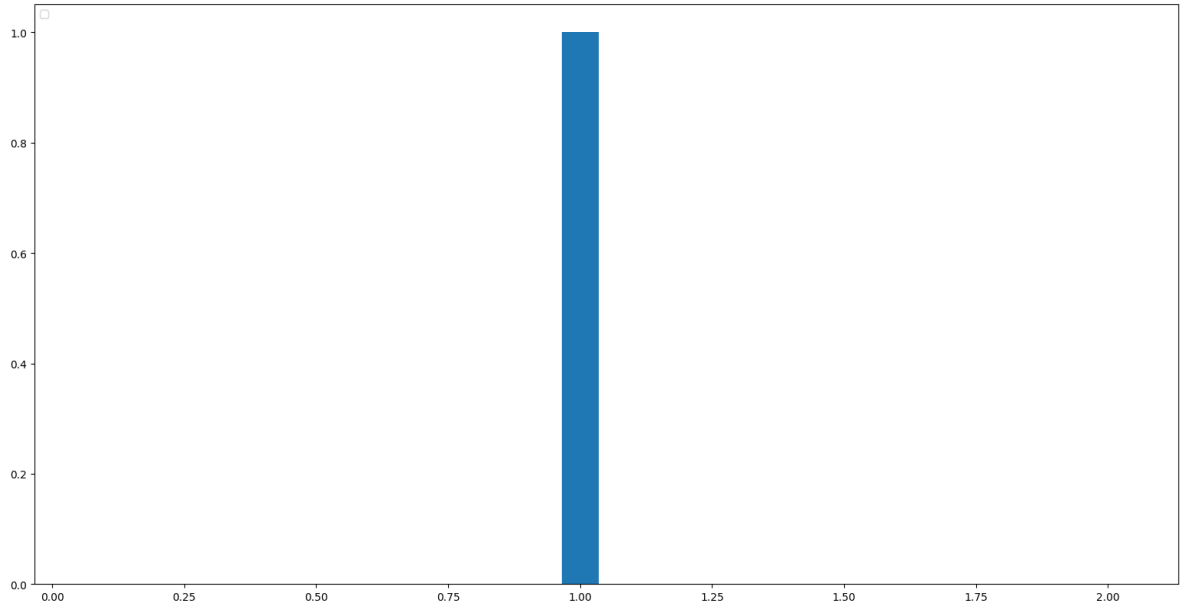
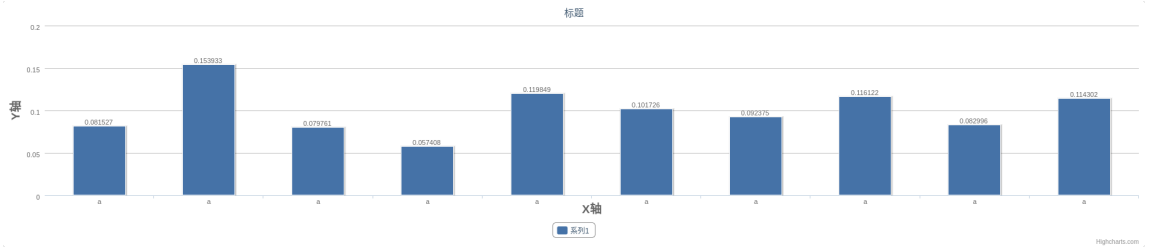
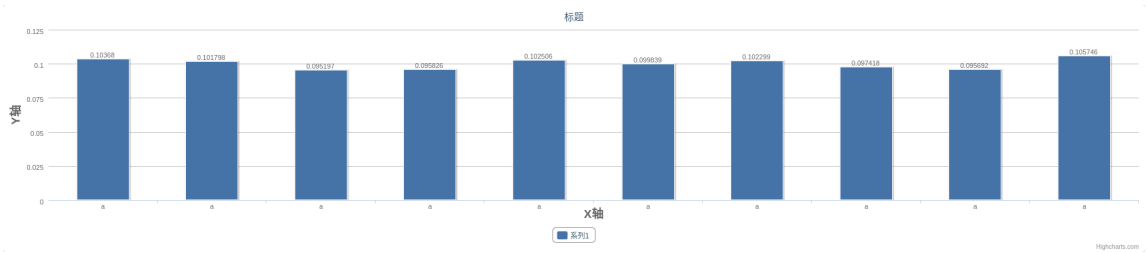



图 3.16: 四维最优传输测度比率的频率图




(a) 初始映射的测度分布




(b) 迭代 20 次后映射的测度分布

图 3.17: 五维最优传输测度分布情况

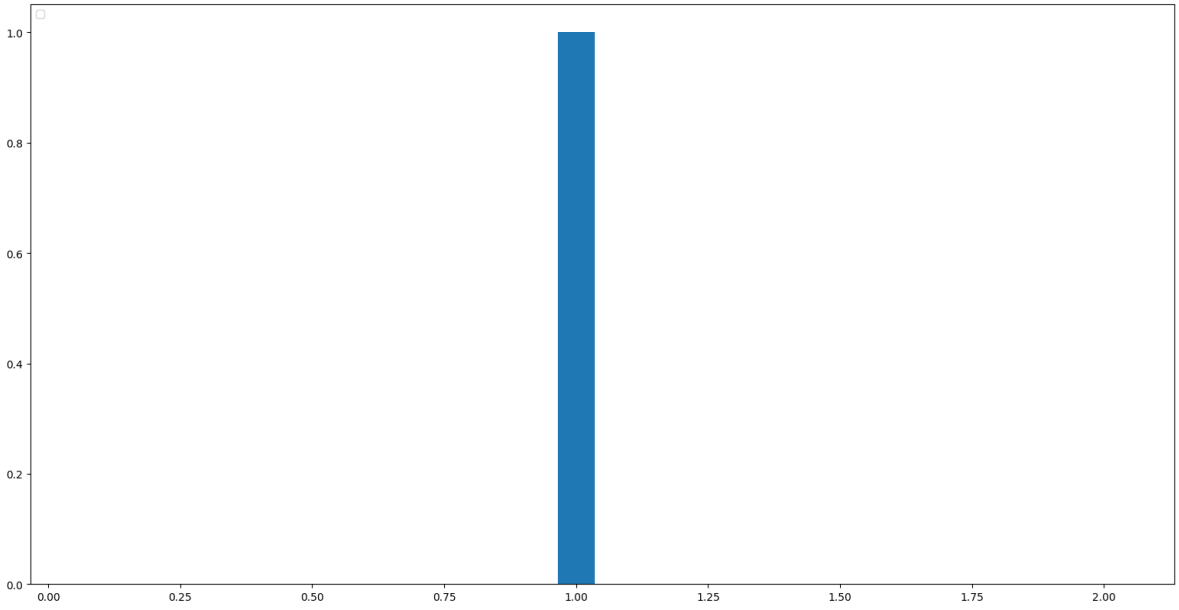



图 3.18: 五维最优传输测度比率的频率图

### 3.2.4 实验结果说明

在本文实验的数据上，高维最优传输算法表现稳定，每次只需要输入目标点集和目标测度，无需更改其它设置。本文实验结果表明随着维数的增加，计算量指数增长。故而为了便于观察数据结果，随着维数增加，本文测试的目标点数在减少。

# 第四章　结论

## 4.1　本文的结果和说明

本文给出通过凸包算法的高维最优传输的算法。

第二章的基础知识部分,是作者在算法实践过程独立完成的。Brenier 理论和 Alexandrov 理论，变分原理的构造性证明则是介绍性内容。第三章的算法和反对称张量是作者主要工作内容。

除了论文中明确说明的他人工作，作者的主要工作有:

通过第二章基础知识部分的讨论，作者发现原函数值和其共轭函数相应点的截距有密切的关系，意识到最优映射的关键信息完全可以由原函数得到。在第三章，作者通过反对称张量的有力工具统一了空间定向问题和空间面积计算。然后作者给出了基于凸包的高维最优传输算法的流程。

作者在算法实践过程中实现了 c 语言的高维网格库 libcell 和包含反对称张量的代数库 Algebras, 这些已经在 Github 上开源。

# 参考文献


[1] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adversarial Networks. *arXiv e-prints*, page arXiv:1406.2661, June 2014.

[2] Martin Arjovsky, Soumith Chintala, and Leon Bottou. Wasserstein GAN. *arXiv:1701.07875*, 2020.

[3] Na Lei, Zhongxuan Luo, and David Xianfeng Gu Shing-Tung Yau. Geometric Understanding of Deep Learning. *arXiv:1805.10451*, 2018.

[4] Alexandrov. *Convex Polyhedra*. Springer Monographs in Mathematics, 1950.

[5] Cedric Villani. *Topics in Optimal Transportation*. American Mathematical Society, Graduate Studies in Mathematics 58, 2003.

[6] Lenaic Chizat, Gabriel Peyré, Bernhard Schmitzer, and François-Xavier Vialard. Unbalanced optimal transport: Dynamic and kantorovich formulation, 2015.

[7] Xianfeng GU, Feng Luo, Jian Sun, and S-T Yau. Variational Principles for Minkowski Type Problems, Discrete Optimal Transport, and Discrete Monge-Ampere Equations . *arXiv:1302.5472*.

[8] 周宇明, 苏科华. 使用 GPU 渲染的离散最优传输算法. 计算机辅助设计与图形学学报, 2019.

[9] Brouno Levy. A numerical algorithm for L2 semi-discrete optimal transport in 3D. *arXiv:1409.1279*, 2014.

[10] 顾险峰. 最优传输理论 (4). https://mp.weixin.qq.com/s/B3i16sL1j9XeC4Wl5loQDA, 2016.

[11] Gabriel Peyre and Marco Cuturi. Computational optimal transport. *arXiv:1803.00567*, 2018.

[12] 顾险峰. 流体力学下的最优传输. https://mp.weixin.qq.com/s?__biz=MzA3NTM4MzY1Mg==&mid=503331022&idx=1&sn=ce120fa62348037b56492d7791c0544e&chksm=

0485cec533f247d390a9522ba9d057cd5761942dd37309b2f8d2d97d7696711c36c30702dc68&
mpshare=1&scene=1&srcid=&sharer_sharetime=1585204805615&
sharer_shareid=8888a07a6b5ddbd17f1d5fa6d12590b7&exportkey=
A10Y3NcR3fTtpMx8s3QOCiQ%3D&pass_ticket=obK0uQFnhY%
2Bg0tQgi4VvKqNNlifRAUVoWa9SVly3BVYBHNIW6IG6%2Fi%2BgNoz8V9ru#rd,
2016.

[13] Marco Cuturi, Olivier Teboul, and Jean-Philippe Vert. Differentiable Ranks and Sorting using Optimal Transport. *arXiv:1905.11885*, 2019.

[14] Xin Zhao, Zhengyu Su, Xianfeng David Gu, and Arie Kaufman. Area-preservation mapping using optimal mass transport. *IEEE Transactions on Visualization Computer Graphics*, 19(12):2838–2847, 2013.

[15] John Lott and Cedric Villani. Ricci curvature for metric-measure spaces via optimal transport. *Annals of Mathematics*, 169(3):903–991, May 2009.

[16] Yann Ollivier. Ricci curvature of markov chains on metric spaces. *Journal of Functional Analysis*, 256(3):810–864, Feb 2009.

[17] Eschrig H. *Legendre Transformation*. Vieweg+Teubner Verlag, Wiesbaden, 1996.

[18] 张恭庆，林源渠. *泛函分析*. 北京大学出版社, 2001.

[19] 李克正. *抽象代数基础*. 清华出版社, 2007.

[20] Y. BRENIER. Polar factorization and monotone rearrangement of vector-valued functions. *Communications on Pure Applied Mathematics*, 44, 1991.

[21] 陈维桓. *微分流形初步*. 高等教育出版社, 2002.

[22] 刘新东. *张量分析*. 国防工业出版社, 2009.

[23] Allen Hatcher. *Algebraic topology*. 清华大学出版社, 2005.

[24] Wikipedia. Simply connected space. https://en.wikipedia.org/wiki/Simply_connected_space, 2020.

# 附录

## A 高维网格库

libcell 库是本文的依赖库。

libcell 可以表示 $n$ 维流形，并且不仅可以表示单形，其它任何形式也可以（其实它的要求非常宽松，甚至不是流形的图也可以表示）。

但是 libcell 只保存了三个层次的信息。vertex(0 维子流形)，face(n-1 维流形)，cell(n 维流形)（对于单形我们在上文已经讨论过，你完全可以推导出各层结构信息。）。

例如 $n = 2$。说明你要表示一张曲面，那么 cell 是多边形，face 是边，vertex 是点。

$n = 3$ 说明你要表示三维体结构，那么 cell 是一个多面体，face 是面，vertex 是点。依次类推。

其次 libcell 借鉴了 openmesh 优秀统一的接口遍历方式，使用 libcell 的遍历操作就像 openmesh 一样简洁，方便。还有要注意的是 libcell 底层函数都是基于 c 写的，基于 c++ 都是操作符重载。这样虽然既保障效率又保障便利，但是你需要手动释放某些内存。下面是 libcell 节点容器 Node 的用法：

```
Node* temp_node;
for(Node n_it=*temp_node;*n_it!=NULL;n_it++)
{}
//以上是对Node* temp_node遍历
//以下直接对结构体Node遍历，两者一样
Node temp_node1;
for(Node n_it=temp_ndoe1;*n_it!=NULL;n_it++)
{

}
```

想要了解更多 Node 接口，可以在 tools_node.h 中定义了 Node 常用的函数，lib_cell_iterator.h 中对 Node 进行了 ++,* 的运算符重载。

```
Node* temp_node;
for(Node n_it=*temp_node;*n_it!=NULL;n_it++)
{}
//以上是对Node* temp_node遍历
//以下直接对结构体Node遍历，两者一样
```

```
Node temp_node1;
for(Node n_it=temp_ndoe1;*n_it!=NULL;n_it++)
{

}
```

libcell 还有仿 openmesh 的网格遍历接口。

```
for(auto fit=mesh.faces.begin();fit!=mesh.faces.end();fit++)
   //对面的遍历
{
    for(auto fvit=mesh.fv_begin(&mesh,*fit->second);fvit!=
       mesh.fv_begin(&mesh,*fit->second);fvit++)//面上点的遍历
    {
        (*fvit) //点(结构体实例, 对它修改不会影响到真实的点)
         quote(cvit)//点(引用指针, 直接修改的话会影响到实体)
    }
}
```

还可以用下面的遍历方式。

```
for(auto fit=mesh.faces.begin();fit!=mesh.faces.end();fit++)
   //对面的遍历
{
        for(int i=0;i<fit->second->vertices_size;i++)//面上点
           的遍历
    {
        (template_v*)fit->second->vertices[i]//点
    }
}
```

```
for(auto cit=mesh.cells.begin();cit!=mesh.cells.end();cit++)
   //对cell的遍历
{
    for(auto cvit=mesh.cv_begin(&mesh,*cit->second);cvit!=
       mesh.fv_begin(&mesh,*cit->second);fvit++)//cell对点的遍
       历
    {
        (*cvit)//点(结构体)
```

```
6           quote(cvit)//点(引用指针, 直接修改的话会影响到实体)
7       }
8   }
9   //还可以用以下方法
10  for(auto cit=mesh.cells.begin();cit!=mesh.cells.end();cit++)
        //对cell的遍历
11  {
12      for(int i=0;i<cit->second->vertices_size;i++)//cell对点的
            遍历
13      {
14          (template_v*)cit->second->vertices[i]//点
15      }
16  }
17  for(auto cit=mesh.cells.begin();cit!=mesh.cells.end();cit++)
        //对cell的遍历
18  {
19      for(auto cfit=mesh.cf_begin(&mesh,*cit->second);cfit!=
            mesh.cf_begin(&mesh,*cit->second);cfit++)//cell对face的
            遍历
20      {
21          (*cfit)//face(结构体实例)
22          quote(cfit)//face指针(引用指针, 直接修改的话会影响到
                实体)
23      }
24  }
25  for(auto vit=mesh.vertices.begin();vit!=mesh.vertices.end();
        vit++)//对点的遍历
26  {//点对cell的遍历
27      for(auto vcit=mesh.vc_begin(&mesh,*vit);vcit!=mesh.vc_end
            (&mesh,*vit);vcit++)
28      {
29                  (*vcit)//
30          quote(vcit);
31      }
32  }
33  for(auto vit=mesh.vertices.begin();vit!=mesh.vertices.end();
```

```
    vit++)//点的遍历
{
    //点对face的遍历
    for(auto vfit=mesh.vf_begin(&mesh,*vit);vfit!=mesh.vf_end
        (&mesh,*vit);vfit++)
    {
                (*vfit)//
        quote(vfit);//
    }

}
```

```
template_f* f;
f->halfface[0]->cell//face访问halfface,再访问到cell, 更多的用
    法完全仿照openmesh
```

对比 openmesh 的遍历接口学习成本会降低。libcell 还有针对 n 维流形求边界的算子。

```
mesh->external_cell_init_(mesh);
```

下面这些规定和反对称张量一节的讨论是一致的。

**规定 1.**

一个 cell 或者 face,halfface 的点顺序为 $p_0p_1...p_n$, 那么对应的张量或者矩阵是 d $(p_1-p_0)\wedge d(p_2-p_0)...\wedge d(p_n-p_0)$。

**规定 2.**

张量运算向后添加. 给一个单形 $p_0p_1...p_n$,, 对于子流形的顺序 $p_0...\hat{p_i}..p_n$ 是否交换, 看 $(-1)^{n-i}$ 的值。这样给一个 halfface 的点顺序是 $p_0p_1..p_n$, 再给一个点 $p_{n+1}$, 那么组成的 cell 的点顺序是 $p_0...p_np_{n+1}$, 符合张量向后添加。

**规定 3.**

一个 $n$ 维流形, 它是 $n+1$ 个点组成的凸包, 那么它的张量方向是。

$d(p_1-p_0)\wedge d(p_2-p_0)\wedge ..d(p_{n+1}-p_0)$。

**规定 4.**

cell 是 n 维流形, 它的边界 halfface 的点是 $p_0,p_1,...p_l, l<n$, 那么这个 halfface 的反对称张量是 $(p_1-p_0)\wedge(p_2-p_0)\wedge...(p_{n-1}-p_0)$, 也就是由前 $n-1$ 个点确定。

# B 可视化库

在 Viewer 库中所有皆 Something, 就像 linux 系统中一切皆文件。

比如纹理，曲面，相机，线段，点. 交互器等等这些都是由结构体 Something 表示。

Viewer 的框架分为两层，一层为抽象描述层，也就是描述所有 Something 的一层，底层（第二层）是解释器层，根据你的抽象描述 (这些描述与工程框架无关) 对接工程框架再显示出来。

Viewer 的解释器现在只提供 OpenGl 的工程框架，你也可以写其它的工程框架进行解释而不需要修改抽象层。简单地讲，抽象层面向用户，所以用户的代码与平台，工程框架无关。

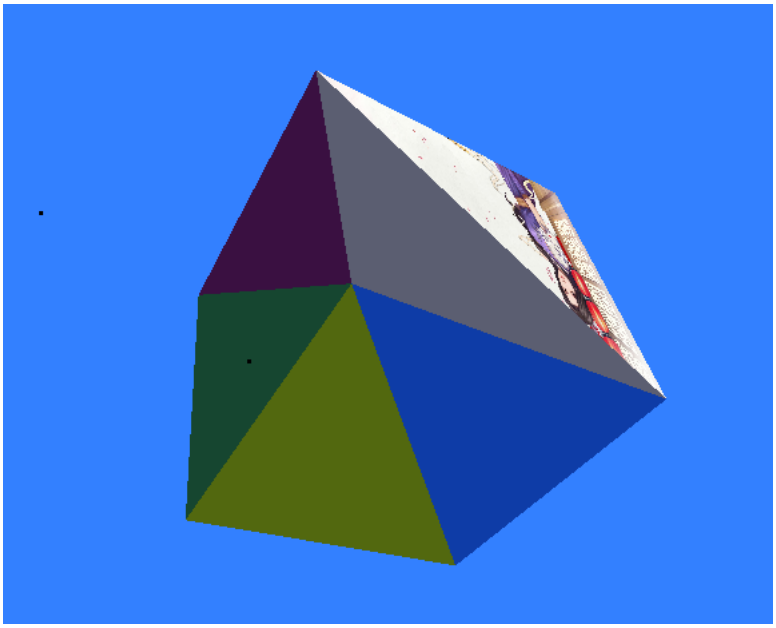


图 1: viewer 库展示

如上图展示了纹理，曲面，点，交互器，相机的综合测试。

VIewer 也支持 3 维拾取。之所以称之为抽象层，因为用户在抽象层描述你要绘制的 something 根本不需要考虑工程细节。

比如下面的代码解释:

```
1  char ch[]="faces";
2  Node* n=Mesh_viewer_world_create_something(&mw,ch);
3  Mesh_viewer_something* ms=(Mesh_viewer_something*)(n->
       value);
```

```
4    Mesh_viewer_faces* mf=(Mesh_viewer_faces*)(ms->evolution)
         ;
5    free_node(node_reverse(n));
6    mf->color_rows=12;mf->normal_rows=12;
7    //曲面颜色的个数，法向量的个数。
8    float *v=(float*)malloc(sizeof(float)*8*3);//给点集数据声
         明大小
9    unsigned int *f=(unsigned int*)malloc(sizeof(unsigned int
         )*4*12);
10   //给曲面索引声明大小，支持多边形
11   float *texcoords=(float*)malloc(sizeof(float)*12*7);
12   //给纹理坐标声明大小
13   memset(texcoords,0,sizeof(float)*12*7);
14   mf->random_color(mf);
15   //曲面颜色随机赋值，如果mf->color_rows等于点数，那么随机
         给点赋值，否则给面赋随机颜色
16   //ms->disappear=1;
17   //是否将曲面消失（不绘制）
18   v[0*3+0]=-0.500000;v[0*3+1]=-0.500000;v[0*3+2]=0.500000;
19   v[1*3+0]=0.500000; v[1*3+1]=-0.500000; v[1*3+2]=0.500000;
20   v[2*3+0]=-0.500000;v[2*3+1]= 0.500000;v[2*3+2] =0.500000;
21   v[3*3+0]=0.500000;v[3*3+1]=0.500000;v[3*3+2] =0.500000;
22   v[4*3+0]=-0.500000;v[4*3+1]= 0.500000;v[4*3+2]=
         -0.500000;
23   v[5*3+0]=0.500000;v[5*3+1]= 0.500000;v[5*3+2]= -0.500000;
24   v[6*3+0]=-0.500000;v[6*3+1]= -0.500000;v[6*3+2]=
         -0.500000;
25   v[7*3+0]=0.500000;v[7*3+1]= -0.500000;v[7*3+2]=
         -0.500000;
26   f[0*4+0]=3; f[0*4+1]=3;f[0*4+2]=2;f[0*4+3]=1;
27   f[1*4+0]=3; f[1*4+1]=1;f[1*4+2]=2;f[1*4+3]=0;
28   f[2*4+0]=3; f[2*4+1]=5;f[2*4+2]=4;f[2*4+3]=3;
29   f[3*4+0]=3; f[3*4+1]=3;f[3*4+2]=4;f[3*4+3]=2;
30   f[4*4+0]=3; f[4*4+1]=7;f[4*4+2]=6;f[4*4+3]=5;
31   f[5*4+0]=3; f[5*4+1]=5;f[5*4+2]=6;f[5*4+3]=4;
32   f[6*4+0]=3; f[6*4+1]=1;f[6*4+2]=0;f[6*4+3]=7;
```

```
33      f[7*4+0]=3;  f[7*4+1]=7;f[7*4+2]=0;f[7*4+3]=6;
34      f[8*4+0]=3;  f[8*4+1]=5;f[8*4+2]=3;f[8*4+3]=7;
35      f[9*4+0]=3;  f[9*4+1]=7;f[9*4+2]=3;f[9*4+3]=1;
36      f[10*4+0]=3;  f[10*4+1]=2;f[10*4+2]=4;f[10*4+3]=0;
37      f[11*4+0]=3;f[11*4+1]=0;f[11*4+2]=4;f[11*4+3]=6;
38      for(int  i=0;i<12;i++)
39      {
40          texcoords[i*7]=3;
41
42      }
43     /* texcoords[0]=3;
44      texcoords[1]=1.0;texcoords[2]=0.0;
45      texcoords[3]=0.0;texcoords[4]=0.0;
46      texcoords[5]=0.0;texcoords[6]=1.0;*/
47      texcoords[7]=3;
48      texcoords[8]=1.0;texcoords[9]=0.0;
49      texcoords[10]=0.0,texcoords[11]=1.0;
50      texcoords[12]=1.0;texcoords[13]=1.0;
51      mf->Data=v;
52      mf->Data_index=f;
53      mf->Data_rows=8;
54      mf->Data_index_rows=12;
55      //ms->disappear=1;
```

下面展示交互器控制相机的代码。

```
1       char Intera[]="Intera";
2       Node* n=Mesh_viewer_world_create_something(mw,Intera);
3       Mesh_viewer_something *ms=(Mesh_viewer_something*)(n->
            value);
4       Mesh_viewer_Intera* mi=(Mesh_viewer_Intera*)(ms->
            evolution);
5       //mi->g_info=mw->g_info;
6       Mesh_viewer_camera* mc=0;
7
8       free_node(n);
9
```

```
10      char camera[]="Camera";
11      Node*id=Mesh_viewer_world_find_species(mw,camera);
12
13      std::map<int,std::map<int,Mesh_viewer_something*>*>::
            iterator iter=mw->species2somethings.find(*((int*)(id->
            value)));
14      for(auto iter1=iter->second->begin();iter1!=iter->second
           ->end();iter1++)
15      {
16          mc=(Mesh_viewer_camera*)(iter1->second->evolution);
17          if(iter1->second->disappear==0&&mc->is_using==1)
18          {
19              break;
20
21          }
22           else
23           {
24               mc=0;
25           }
26      }
27
28      Mesh_viewer_Arcroll* ma=(Mesh_viewer_Arcroll*)malloc(
            sizeof(Mesh_viewer_Arcroll));
29      Mesh_viewer_Arcroll_init(ma);
30      mi->representation=(void*)ma;
31      ma->mc=mc;
32      mi->cursor_position_callback=
            Mesh_viewer_Arcroll_cursor_position_callback;
33      mi->scroll_callback=Mesh_viewer_Arcroll_scroll_callback;
34      mi->mouse_button_callback=
            Mesh_viewer_Arcroll_mouse_button_callback;
35      free_node_value(id);
36      free_node(id);
```

# 致　　谢

三年的学习生活在即将划上一个句号，而对于我的人生来说却仅仅只是一个逗号，我将面对新的征程的开始。本研究及论文是在我的导师雷娜的亲切关怀和耐心的指导下完成的。伟人，名人固然为我所崇拜，可是我更迫切地想要把我的敬意献给两位人:我的导师雷娜和于祖焕老师。也许我不是您最出色的学生，但您确是我最尊敬的老师。您是如此的治学严谨，思想深刻，您用心为我营造一种良好的学术氛围，让我的论文更加严谨，同时，我还要感谢实验室的同学们，你们优秀的素质和广阔的知识面让我受益匪浅，如果没有你们三年的相伴，我是无法解决这些困难和疑惑的。

至此论文付梓之际，我的心情无法保持平静，从开始选课题到论文顺利答辩，有无数可敬的师长，朋友给了我很多帮助，在这里请你们接受我诚挚的谢意。

最后感谢我的母校，首都师范大学的三年是充实的三年，平静的三年，珍贵的三年。