OpenCascade 中的 Thin Plate Spline 插值

许中兴
PLCT 实验室
智能软件研究中心
中国科学院软件研究所
xuzhongxing@iscas.ac.cn

1 理论背景

OpenCascade 中处理一类带约束的填充曲面生成问题使用的方法是将其转化成 scattered data interpolation 问题。搜索 scattered data interpolation 可以得到一大堆的文献。OpenCascade 所使用的是基于 Thin Plate Spline 函数的插值方法。

Thin plate spline 函数最早出现在 [1] 中，在那里被称为 surface spline，不过在这篇文章中作者没有给出严格的理论基础。值得一提的是，这篇文章的作者 Robert L. Harder 当时是 MacNeal-Schwendler 公司的工程师。如果 MacNeal-Schwendler 这个名字很陌生的话，那么它后来的名字 MSC Software 则是在仿真软件领域众所周知的。这是题外话，按下不表。

Thin plate spline 函数定义如下：

$$\phi(r_i) = r_i^2 log(r_i)$$

这里我们给出的是文献中的标准形式，这个函数也是 Laplace 方程 $\Delta\phi = c\delta$ 的基础解形式。在后面对 OpenCascade 源代码的说明中，我们会给出在实现中实际使用的函数形式。

文献 [2] 中给出了严格的理论基础，证明了使用 thin plate spline 函数插值得到的函数能够将曲面的应变能最小化。用白话说就是基于 thin plate spline 函数的插值方法得到的曲面是所有满足插值条件中最平整光滑的。

这一部分的证明用到了泛函分析，Sobolev 空间，广义函数 (Schwartz-Sobolev 分布理论) 方面的知识，我还没有时间仔细学习研究。

2 OpenCascade 的实现

在实现中实际使用的 thin plate spline 函数的形式是：

$$\phi_i(u, v) = R_i^d log(R_i)$$

以及 phi 函数的各阶偏导数 $\bar{\phi}$: $\phi_u = \frac{\partial}{\partial u}\phi$, $\phi_v = \frac{\partial}{\partial v}\phi$, $\phi_{uv} = \frac{\partial^2}{\partial u \partial v}\phi$ 等等。

完整的插值函数是：

$$f(u, v) = \sum_{i=1}^{N} \alpha_i \phi_i + \sum_{j=1}^{M} \beta_j \bar{\phi} + p_d(u, v)$$

其中 $R_i = (u - u_i)^2 + (v - v_i)^2$，是坐标点到某个插值坐标点的距离的平方。这里的 log 函数里面的变量是距离的平方，和标准形式的距离有所不同，但是因为 log 里面的指数可以直接拿到前面被自由度变量 α_i 吸收进去，所以无妨大碍。

d 是可配置参数，指定次数，默认值是 2. 在 occt 里面实际指定的是 order, 默认值是 3, 等于次数加 1.

$p_d(u, v)$ 是关于 u 和 v 的所有次数小于等于 d 的单项式的和。例如，当 $d = 2$ 时，

$$p_2(u, v) = \gamma_1 + \gamma_2 u + \gamma_3 v + \gamma_4 u^2 + \gamma_5 uv + \gamma_6 v^2$$

加入多项式是因为对于 thin plate spline 函数来说，如果没有多项式项，后面形成的线性方程系数矩阵可能不可逆 [3].

这里和标准的 thin plate spline 插值不同的是，还加入了关于 ϕ 的偏导数的项。这是因为，在指定曲面的约束的时候，有一类约束是关于参数的偏导数的约束，比如我们可以指定需要满足的一条填充曲线的连续性是切向连续。为了满足这类约束，我们需要加入关于 ϕ 的偏导数的项，这个理论基础是文献 [4] 提供的（信息来自于文献 [5]）。

我们提一下如何构造切向连续的偏导约束。要保证切向连续，实际上就是要让在某点处的法向量和相邻曲面的法向量重合，这也称为 G^1 连续。我

们要做的是将当前的偏导数向目标法向量投影，然后再将这个投影变换到当前的法方向上，得到的最终向量就是这个偏导数需要改变的量（需要加一个负号，因为方向反了）。具体的代码实现请参考`Plate_GtoCConstraint`类的构造函数。这里为什么不直接用投影向量作为修改量呢？理论上也是可以的，但可能作者希望变换过后的偏导数能大一些，提高计算稳定性。毕竟，我们关心的不是具体的偏导数的大小，而是做了叉积之后形成的法方向要能够和目标法方向相同。

为什么只把需要改变的量作为约束？因为我们会计算一个平均曲面作为初始曲面，然后用插值计算的实际上是这个初始曲面和目标约束之间的差曲面。在求最终曲面上的点的时候，是用初始曲面加上插值出来的差曲面。所以，所有的约束都是差值约束，也就是初始曲面上的点或者偏导数需要变化的量。

回到主题，全部的自由度是 $N+M+d(d+1)/2$。N 是点约束的个数，M 是偏导约束的个数，$d(d+1)/2$ 是多项式的项数。

$N+M$ 个约束可以给出 $N+M$ 个线性方程，剩下的 $d(d+1)/2$ 个线性方程由以下条件给出：

$$\sum_{i=1}^{N} \alpha_i q(u_i, v_i) = 0$$

其中 $q(u,v)$ 是 p_d 中的每一个单项式，正好是 $d(d+1)/2$ 个项。实际上这个地方把 γ_1 限制为 0 了。

整个线性方程组的系数矩阵是对称的，我们只需要计算其中的下三角部分，再拷贝给上三角即可。

u,v 是参数空间的坐标，要插值的坐标 x,y,z 在世界坐标系中。在实现中是进行了 3 次插值，也就是说分别针对 x,y,z 进行插值。

3 Plate_Plate::SolEm 函数

`Plate_Plate::SolEm`这个函数是整个实现中最难理解的一块。里面有很多 magic constants. 我一开始看到这里的时候完全摸不着头脑。不过当推导出上一节中给出的实际 thin plate spline 函数的形式之后，这个函数的含义也就明确了：就是求 thin plate spline 函数的各阶偏导数。那些常数都是求偏导的结果中产生的常数。

不过这个函数里还有一个 trick 必须要解释一下。

这个函数在计算的时候，把参数 u, v 变换到了一个标准的区间之内 $U = u/du, V = v/dv$，应该是 $[0, 2]$ 区间。后续所有的计算都是在这个参数区间内。也就是说它所有的求导都是关于 U, V 的求导。但是，它最终要计算的导数还是关于原始参数 u, v 的导数。我们来看看这其中的差别。

假设原始的表达式是 u^2，在`SolEm`中，它把这个表达式看成 U^2, 计算出的导数是 $2U = 2u/du$. 但是实际上`SolEm`想计算的是关于 u 的导数，$U^2 = u^2/(du)^2$, 它关于 u 的导数是 $2u/(du)^2$。可以观察到，计算出的导数相对于想计算的导数，少乘了一个 $1/du$.

这也就是为什么在`SolEm`的最后，返回的结果要乘上`ddu`和`ddv`数组里的值。

可以这样理解：occt 所使用的插值函数实际上是对 u, v 进行了 scaling 之后的函数，即给它们乘上了一个 $1/du, 1/dv$ 的系数。

参考文献

[1] Robert L. Harder and Robert N. Desmarais. Interpolation using surface splines. *Journal of Aircraft*, 9, 1972.

[2] Jean Duchon. Splines minimizing rotation-invariant semi-norms in sobolev spaces. In *Constructive Theory of Functions of Several Variables*. 1972.

[3] S. K. Lodha and R. Franke. Scattered data techniques for surfaces. In *Scientific Visualization Conference (dagstuhl '97)*. 1997.

[4] Wu Zongmin. Hermite-Birkhoff interpolation of scattered data by radial basis functions. *Approximation Theory and its Applications*, 8, 1992.

[5] A practical guide to radial basis functions. `http://num.math.uni-goettingen.de/schaback/teaching/sc.pdf`.