arXiv:2301.01318v1 [cs.CG] 3 Jan 2023

双二次贝塞尔三角形与四边形曲面的隐式化

预印本

Jackson Borchardt 和 Saul Kato魏尔神经科学研究所，神经内科，加利福尼亚大学旧金山分校，旧金山，CA 94107
jackson.borchardt2 -at- ucsf.edu | saul.kato -at -ucsf.edu

2023年1月5日

摘要

我们为一般的双二次（阶数为 2×2）贝塞尔三角形和四边形曲面片推导隐式方程，并提供函数求值代码，利用现代计算资源发挥传统代数构造技术的优势。

关键词 隐式化 · 计算几何 · 曲面

1 引言

在计算机图形学、计算机辅助设计和物理仿真领域，通常将复杂曲面建模为较简单曲面片的分段组合。贝塞尔曲面是一类用于此类分段组合的参数曲面，它是贝塞尔曲线的二维扩展，是由空间中固定数量（取决于曲面阶数）的控制点进行参数化的多项式曲线。

对于任何参数定义的曲面，都存在一个等价的隐式方程表示 $F(x,\ y,\ z) = 0$，具有多种用途。隐式方程可用于直接判断空间中的任意点位于给定曲面上、曲面的一侧还是另一侧，这对于物理仿真中的命中/碰撞检测以及物体的体素化很有用。通过直接代入，隐式方程还可用于计算一个曲面与另一个参数曲面的交线，这对于光线追踪以及由实体基元并集构成的物体的线框计算很有用。多项式曲面的隐式化过程已被研究逾百年；然而，由于所得隐式方程规模庞大，尚未将其应用于一般的多项式曲面情形，甚至连阶数很低的曲面也不例外。自动符号计算的出现以及计算资源的爆发式增长，促使人们重新审视代数消元理论，以便为生成式建模应用生成具有表现力的数学曲面的实用闭式隐式表达式。

我们推导了具有四边形和三角形输入域的一般双二次（阶数为 2×2）贝塞尔曲面的隐式方程，并提供了 Python 求值代码。

2 背景

具有三角形输入域的双二次 Bézier 曲面通常称为 Bézier 三角形，由六个三维控制点（即 18 个自由参数）参数化。这些控制点标记为 b_{200}、b_{020}、b_{002}、b_{110}、b_{101} 和 b_{011}。b_{200}、b_{020} 和 b_{002} 是构成输入域的三角形的顶点。对于该三角形上的任意点 P，该点的重心坐标（u, v）通过 $\Re^2 \to \Re^3$ 函数映射到曲面上

$$P(u,v) = b_{200}u^2 + b_{020}v^2 + b_{002}w^2 + 2b_{110}uv + 2b_{101}uw + 2b_{011}vw \tag{1}$$

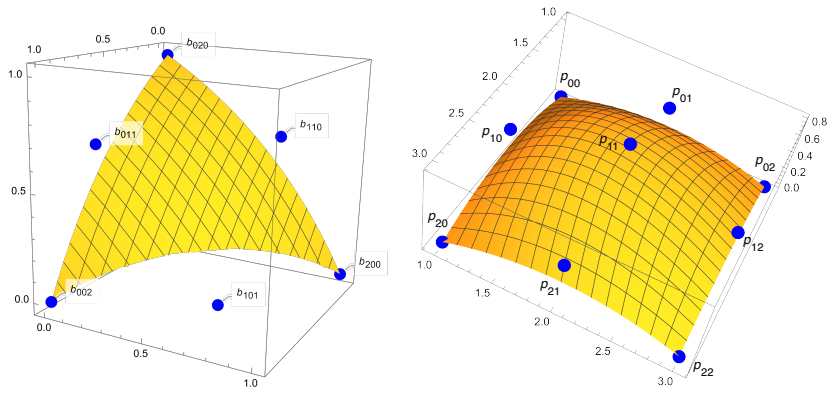

图 1：双二次贝塞尔三角形和双二次贝塞尔四边形曲面片。

根据此函数，我们通过分别减去 x、y 和 z，为每个输出维度（x, y, z）生成一个隐式方程，得到

$$f_x(u,v) = b_{200_x}u^2 + b_{020_x}v^2 + b_{002_x}w^2 + 2b_{110_x}uv + 2b_{101_x}uw + 2b_{011_x}vw - x \tag{2}$$

$$f_y(u,v) = b_{200_y}u^2 + b_{020_y}v^2 + b_{002_y}w^2 + 2b_{110_y}uv + 2b_{101_y}uw + 2b_{011_y}vw - y \tag{3}$$

$$f_z(u,v) = b_{200_z}u^2 + b_{020_z}v^2 + b_{002_z}w^2 + 2b_{110_z}uv + 2b_{101_z}uw + 2b_{011_z}vw - z \tag{4}$$

由于这些方程同时成立时会得到曲面上的一个有效点，因此，该 3 个关于 u 和 v 的多项式组成的系统的结式将是贝塞尔三角形的隐式方程。Sederberg 等人 [1984] 提出了一种使用 Dixon [1909] 所发展的方法对参数多项式曲面进行隐式化的方法，该方法将 Cayley 经典的两个多项式结式构造扩展到了三个方程。

通过引入两个辅助变量 α 和 β，可以构造出一个增广多项式

$$\delta = \frac{1}{(u-\alpha)(v-\beta)} \det \begin{bmatrix} f_x(u,v) & f_y(u,v) & f_z(u,v) \\ f_x(u,\beta) & f_y(u,\beta) & f_z(u,\beta) \\ f_x(\alpha,\beta) & f_y(\alpha,\beta) & f_z(\alpha,\beta) \end{bmatrix} \tag{5}$$

当且仅当 u 和 v 取满足原始隐式方程的值时，对于 α 和 β 的任意取值，该多项式才会消失。该多项式可以写成三个矩阵的乘积

$$\delta = \begin{bmatrix} 1 \\ \alpha \\ \beta \\ \dots \\ \alpha\beta^3 \end{bmatrix} \begin{bmatrix} \dots & \dots & & \dots & \dots \\ \dots & \dots & & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & & \dots & \dots \\ \dots & \dots & & \dots & \dots \end{bmatrix} \begin{bmatrix} 1 \\ u \\ v \\ \dots \\ u^3v \end{bmatrix} \tag{6}$$

使中心矩阵（我们称之为 Cayley 矩阵，对于双二次多项式，其最大阶数为 2*2*2 = 8）的元素，是针对 u、v、α 和 β 的每种非零系数组合的系数（这些系数将是关于 x、y 和 z 的线性多项式）。如果 Cayley 矩阵的行列式不恒为零 [1]，那么它就是三个初始隐式方程的结式，称为 Dixon 结式 [Kapur 等人，1994]，；然而，行列式的阶数可能使这项计算难以实现。

此方法也可用于计算输入域为正方形的双二次 Bézier 曲面的隐式方程，这种曲面称为片。该曲面由一个 3×3 网格中的九个控制点参数化，这些控制点标记为 p_{ij}，其中 i 和 j 为网格索引：

$$P(u,v) = \sum_{i=0}^{2}\sum_{j=0}^{2} \frac{2}{i!(2-i)!}u^i(1-u)^{2-i}\frac{2}{j!(2-j)!}v^j(1-v)^{2-j}p_{ij} \tag{7}$$

[1]某些多项式系统可能需要更复杂的扩充方式，例如加入一行包含诸如 $\lambda f_x(u,\ \beta) + \mu f_x(\alpha,\ v)$ 这样的项，以避免行列式为零，但这里不需要这样做。

对于单位正方形上的任意点（u， v）。

遵循 上述针对该映射所阐述的相同方法，可得到三个隐式子方程

$$\begin{aligned}0 = {} & p_{00_x}(1-u)^2(1-v)^2 + 2p_{01_x}v(1-u)^2(1-v) + p_{02_x}v^2(1-u)^2 + 2p_{10_x}u(1-u)(1-v)^2 \\ & + 4p_{11_x}uv(1-u)(1-v) + 2p_{12_x}uv^2(1-u) + p_{20_x}u^2(1-v)^2 + 2p_{21_x}u^2v(1-v) + p_{22_x}u^2v^2 - x\end{aligned} \tag{8}$$

$$\begin{aligned}0 = {} & p_{00_y}(1-u)^2(1-v)^2 + 2p_{01_y}v(1-u)^2(1-v) + p_{02_y}v^2(1-u)^2 + 2p_{10_y}u(1-u)(1-v)^2 \\ & + 4p_{11_y}uv(1-u)(1-v) + 2p_{12_y}uv^2(1-u) + p_{20_y}u^2(1-v)^2 + 2p_{21_y}u^2v(1-v) + p_{22_y}u^2v^2 - y\end{aligned} \tag{9}$$

$$\begin{aligned}0 = {} & p_{00_z}(1-u)^2(1-v)^2 + 2p_{01_z}v(1-u)^2(1-v) + p_{02_z}v^2(1-u)^2 + 2p_{10_z}u(1-u)(1-v)^2 \\ & + 4p_{11_z}uv(1-u)(1-v) + 2p_{12_z}uv^2(1-u) + p_{20_z}u^2(1-v)^2 + 2p_{21_z}u^2v(1-v) + p_{22_z}u^2v^2 - z\end{aligned} \tag{10}$$

这些方程的结式同样可以使用 Dixon 方法计算。

3 结果

我们使用符号计算库 SymPy [Meurer 等人， 2017] 利用参数化隐式方程，为一般双二次贝塞尔三角形和四边形构造凯莱矩阵。由于 SymPy 计算大型符号行列式的速度较慢，我们使用 Mathematica 计算行列式展开式，从而得到一般双二次贝塞尔三角形和四边形的隐式方程。双二次三角形的凯莱矩阵为 5x5，因为项 u^3、 u^3v 和 u^2v 不会出现在 δ 多项式中，且 δ 的形式为

$$\delta = \begin{bmatrix}1\\ \alpha\\ \beta\\ \beta^2\\ \alpha\beta\end{bmatrix}\begin{bmatrix}m_{00} & m_{10} & m_{20} & m_{30} & m_{40}\\ m_{01} & m_{11} & m_{21} & m_{31} & m_{41}\\ m_{02} & m_{12} & m_{22} & m_{32} & m_{42}\\ m_{03} & m_{13} & m_{23} & m_{33} & 0\\ m_{04} & m_{14} & 0 & m_{34} & m_{44}\end{bmatrix}\begin{bmatrix}1\\ u\\ v\\ u^2\\ uv\end{bmatrix} \tag{11}$$

其中 $m_{ij} = A_{ij}x + B_{ij}y + C_{ij}z + D_{ij}$和 A、 B、 C、 D是控制点分量b_{***}的多重线性多项式。

采用相同的构造方法，我们为双二次四边形构造了一个凯莱矩阵。由于多项式中包含所有项 $u^{(0\ldots3)}v^{(0..1)}$，该矩阵为 8x8。

隐式方程过长（纯文本形式下，三角形为 353 kB，四边形为 430 MB），因此无法在此处列出，但已作为文本文件和独立的 Python 函数提供，网址为 https://github.com/focolab/biquadratic- implicitization。三角形的隐式方程包含关于 x、y 和 z 的最高 5 次项，需要 12,986 次乘法运算和 269 次加法运算；四边形的隐式方程包含关于 x、 y 和 z 的最高 8 次项，需要 16,621,762 次乘法运算和 138,561 次加法运算。尽管这些方程是精确的，但它们由许多乘积项相加构成，因此使用有限精度变量求值时可能会出现舍入误差或数值溢出错误。

由于贝塞尔四边形的隐式方程远大于贝塞尔三角形的隐式方程，四边形的求值速度慢得多（我们的测试中每次求值耗时 2.983 秒），而三角形每次求值仅需 0.0024 秒。不过，我们可以通过计算 Cayley 矩阵的数值行列式来近似求值，而且这种近似计算相对较快（我们的测试中，四边形每次求值耗时 0.0024 秒，三角形每次求值耗时 0.0003 秒）。尽管隐式函数和数值行列式计算在某些情况下都容易受到舍入误差影响，例如控制点之间的距离相对于控制点的整体幅值非常小时（图 2），非正式测试表明，在典型情况下，对 Cayley 矩阵行列式进行数值求值，可能足以近似完整隐式函数的求值结果。

3.1 坐标变换减少计算量

在不失一般性的前提下，可对控制点集进行坐标变换，使控制点分量固定为零或一，然后再计算结式，从而为三角形和四边形生成复杂度较低的等价隐式方程。通过分别将三角形上的点 b_{200}、 b_{020} 和 b_{002}，以及四边形上的点 p_{00}、p_{02} 和 p_{20} 映射到点 (0，0，0) 和 (1，0，0)，并分别映射到平面 $z = 0$，这相当于

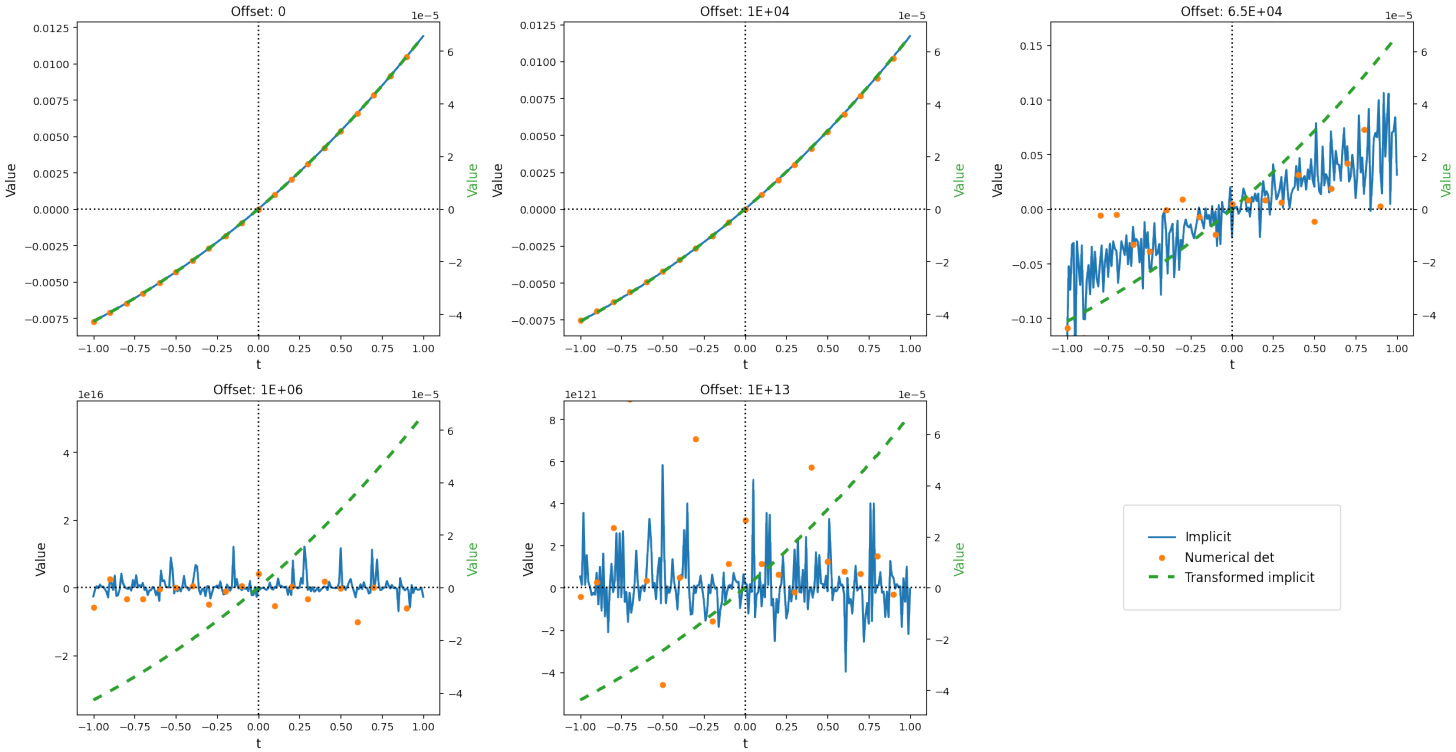

图 2: 通过使用坐标预变换，可以避免某些有限精度失效情形 .

有限精度表示会导致双二次贝塞尔三角形的隐式函数及其数值行列式近似出现误差。我们通过给图 1 中三角形的控制点添加大小为 $\{0, 1E4, 6.5E4, 1E6, 1E13\}$ 的偏移量，构造了五组输入案例。对于每组案例，我们在三角形曲面 ($u = .33,\ v = .33$) 上确定一个点 $\vec{p}$。随后，我们计算一个单位向量 $\hat{v} = \frac{\vec{v}}{\|\vec{v}\|}$，并沿线段 $\vec{p} + t\hat{v},\ -1 \le t \le 1$ 评估隐式函数（蓝色）和数值行列式近似（橙色点）；它们应在 $t = 0$ 处穿过零点。初始控制点为 $b_{200} = (1,\ 0,\ 0)$、$b_{020} = (0,\ 1,\ 0)$、$b_{002} = (0,\ 0,\ 1)$、$b_{110} = (.65,\ .65,\ 0)$、$b_{101} = (.65,\ 0,\ .65)$ 和 $b_{011} = (0,\ .65,\ .65)$，所有计算均在 Python 中使用 64 位浮点数完成。

平移、旋转和均匀缩放，可将用于计算 Dixon 结式的参数方程组中的 18 个系数中的 6 个置零（并消除 7 个自由度）。该变换如下：

$$f(\vec{p}) = \mathbf{R}_2(g(\vec{p}) - g(\vec{c_1})), \qquad g(\vec{q}) = \mathbf{R}_1 \frac{\vec{q}}{\|\vec{c_1} - \vec{c_2}\|} \tag{12}$$

其中，$\vec{c_1}$、$\vec{c_2}$ 和 $\vec{c_3}$ 分别是贝塞尔三角形的控制点 b_{200}、b_{020} 和 b_{002}，或贝塞尔四边形的控制点 p_{00}、p_{02} 和 p_{20}；$\mathbf{R}_1$ 是旋转矩阵，用于将法向量 $(\vec{c_1} - \vec{c_2}) \times (\vec{c_3} - \vec{c_2})$ 对齐到向量 $(0, 0, 1)$；$\mathbf{R}_2$ 是绕 z 轴的旋转矩阵，用于将 $(g(\vec{c_2}) - g(\vec{c_1}))$ 放置在 x 轴上。

采用这种坐标变换技巧，计算三角形隐式函数所需的乘法运算次数从 12,986 次减少到 5,278 次，计算四边形隐式函数所需的乘法运算次数则从 16,621,762 次减少到 4,719,155 次，而所需的加法运算次数保持不变。此外，由于控制点分量的量级，以及由此产生的中间单项式乘积的量级，都经过重新缩放而接近 1（病态细长三角形和曲面片除外），坐标变换方法对有限精度效应的鲁棒性更高（图 2）。

4 讨论

我们生成了双二次贝塞尔三角形和四边形的一般隐式方程，并证明这些方程可用于判定空间中任意一点相对于给定双二次贝塞尔曲面的位置关系（在曲面上、曲面内或曲面外）。双二次曲面能够表现各种平滑的曲面，但

它们提供的自由度不足，难以拼接成任意复合曲面，同时确保接缝处保持 C^1连续性。因此，灵活的对象建模系统通常改用双三次曲面。原则上，此处用于计算双二次情形隐式方程的方法也可应用于更高阶的贝塞尔曲面；但在一般情况下，结式多项式的项数会急剧膨胀，超出实际计算约束。若没有闭式符号函数，对高阶凯莱矩阵的行列式进行数值求值，或许能在许多应用中为其隐式函数提供满意的近似。这些方法应能很好地映射到 GPU架构，因为它们不需要分支或迭代。

5 致谢

J.B. 和 S.K. 参与了构思、理论研究、实现和写作。本研究由威尔神经科学研究所神经计算资助、威尔神经枢纽支柱计划以及美国国立卫生研究院下属美国国立普通医学科学研究所（R35GM124735）资助。

参考文献

T.W. Sederberg、D.C. Anderson 和 R.N. Goldman。参数曲线与曲面的隐式表示。《计算机视觉、图形学与图像处理》，28(1):72–84，1984年。ISSN 0734-189X。doi:https://doi.org/10.1016/0734- 189X(84)90140-3。URL https://www.sciencedirect.com/science/article/pii/0734189X84901403。A.L. Dixon。两个独立变量中三个量子的消元式：（第二篇论文）。《伦敦数学会会刊》，2(1):473–492，1909年。Deepak Kapur、Tushar Saxena 和 Lu Yang。使用迪克森结式进行代数与几何推理。载于国际符号与代数计算研讨会论文集，ISSAC '94，第99–107页，美国纽约州纽约市，1994年。美国计算机协会。ISBN 0897916387。doi:10.1145/190347.190372。URL https://doi.org/10.1145/190347.190372。ˇ Aaron Meurer、Christopher P. Smith、Mateusz Paprocki、Ondˇrej Certík、Sergey B. Kirpichev、Matthew Rocklin、AMiT Kumar、Sergiu Ivanov、Jason K. Moore、Sartaj Singh、Thilina Rathnayake、Sean Vig、Brian E. Granger、Richard P. Muller、Francesco Bonazzi、Harsh Gupta、Shivam Vats、Fredrik Johansson、Fabian Pedregosa、Matthew J. Curry、Andy R. Terrel、Štˇepán Rouˇcka、Ashutosh Saboo、Isuru Fernando、Sumith Kulal、Robert Cimrman 和 Anthony Scopatz。Sympy: Python 中的符号计算。《*PeerJ*计算机科学》，3:e103，2017年1月。ISSN 2376-5992。doi:10.7717/peerj-cs.103。URL https://doi.org/10.7717/peerj-cs.103。