第八次作业报告

PB21010362 汪兆辰

2023 年 5 月 12 日

1 算法介绍

给定 n 个控制点 $P_1^0, P_2^0 \dots$ 生成 Bézier 曲线的算法是由相邻两个点插值得到一个点：

$$tP_i^m + (1-t)P_{i+1}^m = P_i^{m+1} \tag{1}$$

对于 $t \in [0,1]$, 所有控制点经 (n-1) 次 (1) 式操作后可以得到 f(t)，从而可以得到 Bézier 曲线的表达式.

为了得到上述的表达式，可以采用 De Casteljau 算法或利用 Bernstein 基函数，在这里我们利用 Bernstein 基函数得到 Bézier 曲线的表达式.

Bernstein 基函数定义为：

$$B_i^{(n)}(t) = \binom{n}{i} t^i (1-t)^{n-i} \tag{2}$$

利用 Bernstein 基函数的线性组合可以写出 Bézier 曲线表达式：

$$f(t) = \sum_{i=1}^{n} B_i^n \mathbf{p}_i, t \in [0,1] \tag{3}$$

2 算法实现

由 (3) 式给出的表达式需要 Bernstein 基函数，为了计算方便起见没有使用 matlab 内置的 bernstein 函数，而自定义了函数返回 n 阶 Bernstein 基函数在 t 处值组成的向量：

```
function B=Bernstein(n,t)
    coe=diag(flipud(pascal(n+1)));
    num=(t'.^(0:n)).*((1-t').^(n:-1:0));
    B=coe.*num';
end
```

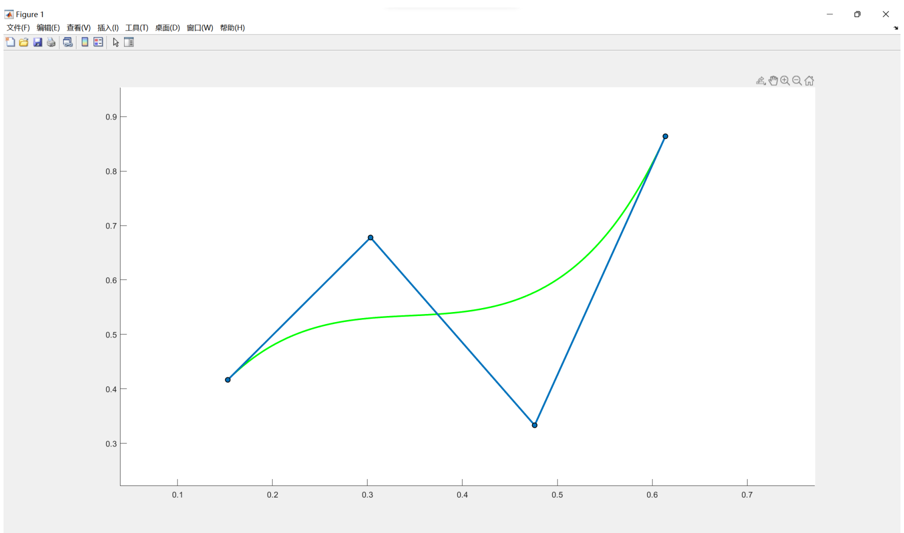
其中为了得到 n 阶二项式系数向量，利用了 n+1 阶 Pascal 矩阵的反对角线.

为了实现交互式编辑，使得拖动控制点可以实时得到新的 Bézier 曲线，对控制点列 p 注册监听器并调用回调函数：

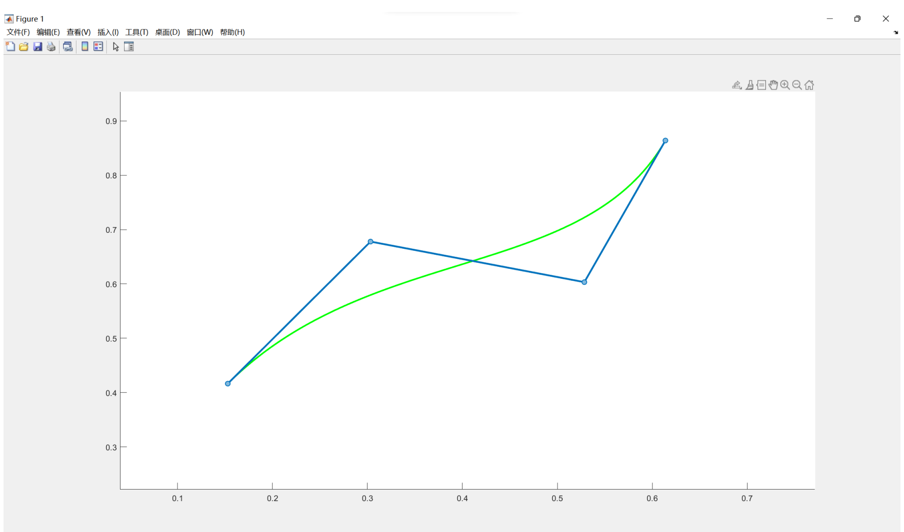
```
h.addlistener('MovingROI',@(h,evt)bezier...
(evt.CurrentPosition,t,hcurve));
```

其中 bezier 将控制点代入 (3) 式计算结果.

3 实现效果



拖动控制点后得到新的 Bézier 曲线:

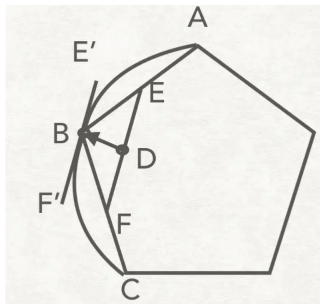


4 Bonus: 用 Bézier 样条实现曲线插值

上述的算法得到的曲线是由控制点拟合而来的，但并不经过所有的控制点，为了能够使生成的曲线经过控制点，考虑利用 Bézier 样条实现曲线插值.

我们知道，Bézier 曲线只经过两个端点而不经过中间的控制点，如果将定义的每个控制点都看作一段 Bézier 曲线的端点，可以由多段 Bézier 曲线生成一整段连续的曲线. 但是注意到，直接简单连接无法保证在曲线段相接处的光滑性，这样的结果并不是我们想要的，因而我们对结果提出 C^2 连续的要求.

为了满足连续性要求，需要用 3 阶的 Bézier 曲线连接相邻两点，而这需要我们自行计算出其余两个控制点. 对于非端点的控制点，可以由如下算法得到，以顶点 B 为例：



1、取 AB 和 BC 的中点 E,F, 并连接 E,F

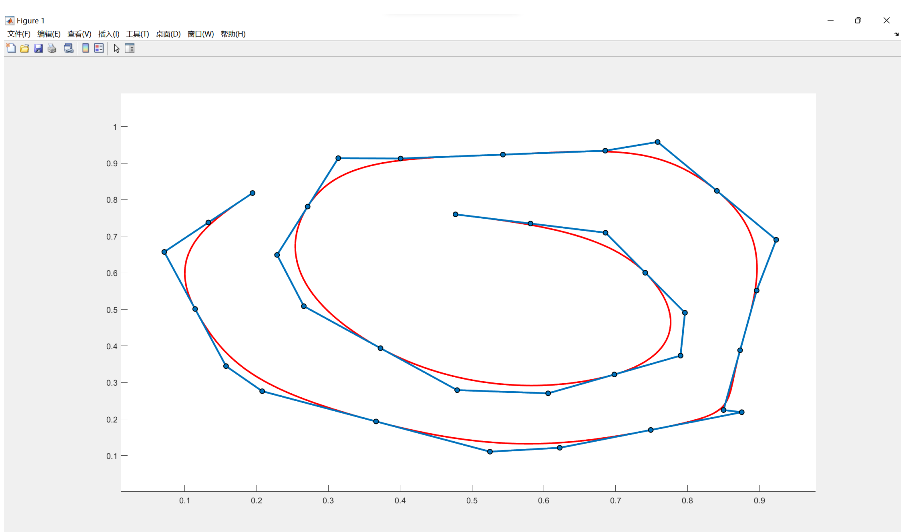
2、在 EF 上取中点 D

3、将直线 EF 按照矢量 DB 平移到通过 B 点，并且使得平移后的 D 和 B 点重合

4、得到 E' 与 F' 点用作贝塞尔曲线的控制点。

而对于端点，利用 Natural end condition，可以取其控制点为端点与相邻控制点的中点. 这样就得到了每段 Bézier 曲线需要的四个控制点，分别生成 Bézier 曲线就可以得到经过所有顶点的曲线.

算法实现的想法主要是先由用户定义所需经过的顶点，由顶点根据上述算法容易计算出所需要的控制点，由得到的控制点再调用已经实现的生成 Bézier 曲线算法即可.



参考文献

[1] https://pages.mtu.edu/~shene/COURSES/cs3621/NOTES/spline/Bezier/de-casteljau.html

[2] https://blog.csdn.net/u014084081/article/details/79798231