哈爾濱工業大學
HARBIN INSTITUTE OF TECHNOLOGY

实验成绩

实验报告

课程名称: 数值逼近

实验项目: Bezier 曲线

所在院系: 信息与计算科学

学生姓名: 葛煜龙

学生学号: 1201200206

授课学期: 22 秋

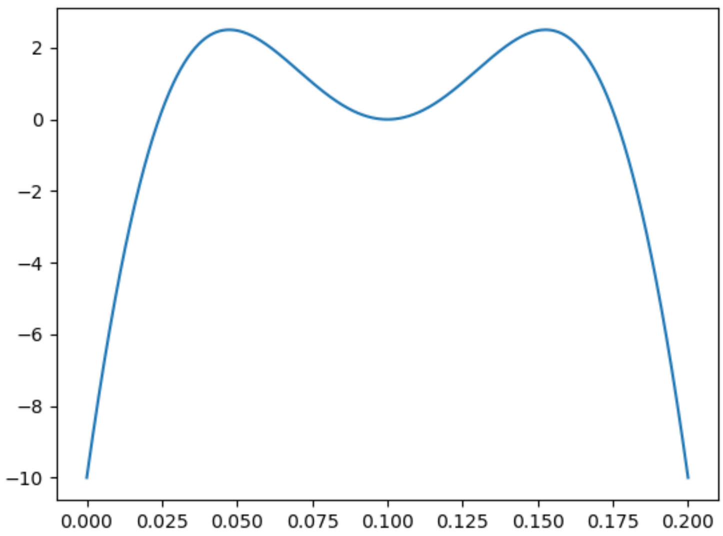
完成时间: 2022.10.16

1 习题一 DrawBezier

下面即为绘制代码。

```
import numpy as np
import matplotlib.pyplot as plt
#输入control points画出bezier曲线(随便画一个形状)

# B = (1-t)*P0+t*P1
def one_bezier_curve(a, b, t):
    return (1 - t) * a + t * b


# 使用de Casteljau算法求解曲线
# Bn(P0,...,Pn)=(1-t)*Pn-1(P0,...,Pn-1)+t*(P1,...,Pn)
def n_bezier_curve(controlPointCoordinate, n, k, t):
    """
    :param controlPointCoordinate: 控制点的x坐标或y坐标
    :param n: n阶bezier曲线
    :param k: 设置控制点下标
    :param t:计算参数为t处的x,y坐标
    :return:参数为t处的x,y坐标
    """
    # 当且仅当为一阶时, 递归结束
    if n == 1:
        return one_bezier_curve(controlPointCoordinate[k],
                                                 controlPointCoordinate[k +
                                                  1], t)
    else:
        return (1 - t) * n_bezier_curve(controlPointCoordinate, n - 1
                                                 , k, t) + t *
                                                 n_bezier_curve(
            controlPointCoordinate, n - 1, k + 1, t)


def bezier_curve(x, y):
    """
    :param x: bezier曲线控制点x坐标数组
    :param y: bezier曲线控制点y坐标数组
    """
    # pyplot作图点
    b_x = []
    b_y = []
    n = len(x) - 1  # n阶bezier曲线
    t_step = 1.0 / 1000
```

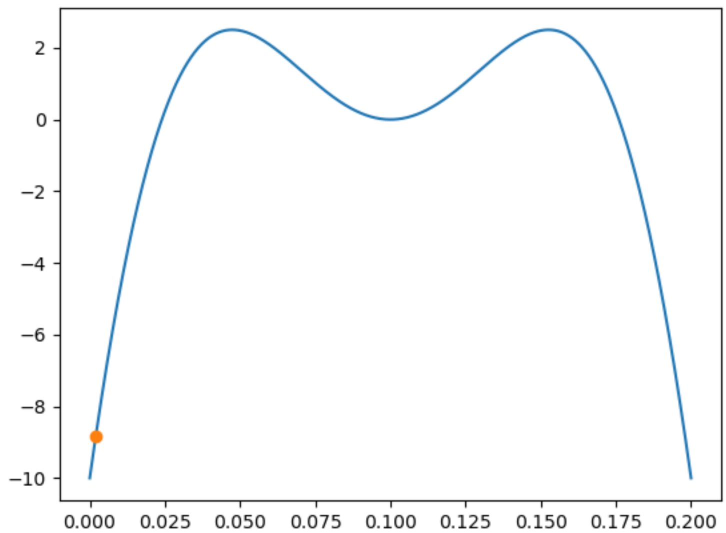
```
    t = np.arange(0.0, 1 + t_step, t_step)
    for each in t:
        b_x.append(n_bezier_curve(x, n, 0, each))
        b_y.append(n_bezier_curve(y, n, 0, each))
    return b_x, b_y


if __name__ == "__main__":
    # x = [int(n) for n in input('x:').split()]
    # y = [int(n) for n in input('y:').split()]
    x = np.linspace(0, 0.2, 7)
    y = [-10, 10, 10, -20, 10, 10, -10]
    # plt.plot(x, y)

    bezier_x, bezier_y = bezier_curve(x, y)
    plt.plot(bezier_x, bezier_y)

    plt.show()
```



2 习题二 求任意输入 t^* 对应点坐标

```
def calculate(t):
    n = len(x) - 1  # n阶bezier曲线
    return n_bezier_curve(x, n, 0, t), n_bezier_curve(y, n, 0, t)


if __name__ == "__main__":
    # x = [int(n) for n in input('x:').split()]
    # y = [int(n) for n in input('y:').split()]
    x = np.linspace(0, 0.2, 7)
    y = [-10, 10, 10, -20, 10, 10, -10]
    # plt.plot(x, y)

    #计算指定t值之后的x,y坐标
    tx, ty = calculate(0.01)
    print(tx, ty)

    bezier_x, bezier_y = bezier_curve(x, y)
    plt.plot(bezier_x, bezier_y)
    plt.plot([tx], [ty], 'o')

    plt.show()
```

假设输入的 t^* 值为 0.01，则在 bezier 曲线上标注如下图所示。



3 习题三 计算细分曲线控制点

下面为计算代码。

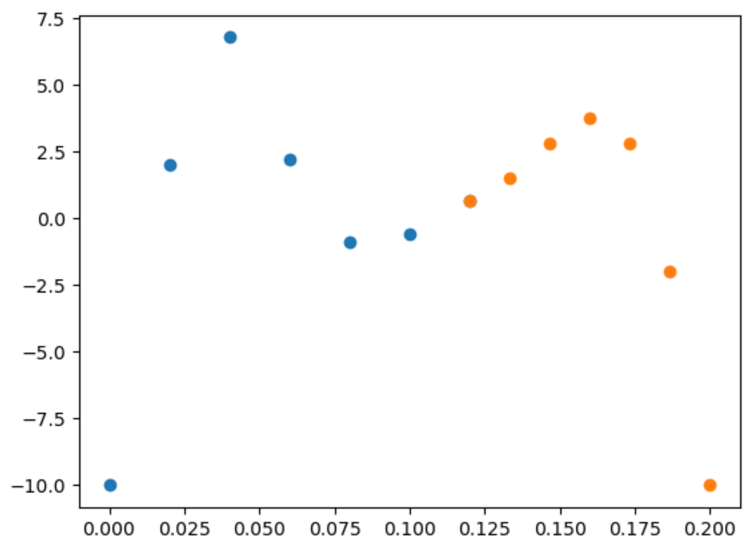
```
def get_subdivision_triangle(t, controlPointCoordinate):
    n = len(controlPointCoordinate)
    num_triangle = np.zeros((n, n))
    for column in range(0, n):
        num_triangle[0][column] = controlPointCoordinate[column]
    for row in range(1, n):
        for column in range(0, n - row):
            num_triangle[row][column] = (1 - t) * num_triangle[row - 1][column] + t * num_triangle[row - 1][column + 1]

    return num_triangle


def get_subdivision_points(subdivision_triangle):
    left_points = subdivision_triangle[:, 0]
    n = len(left_points)
    right_points = []
    for count in range(0, n):
        right_points.append(subdivision_triangle[count, n - count - 1])
    return left_points, right_points


if __name__ == "__main__":
    # x = [int(n) for n in input('x:').split()]
    # y = [int(n) for n in input('y:').split()]
    x = np.linspace(0, 0.2, 7)
    y = [-10, 10, 10, -20, 10, 10, -10]
    # plt.plot(x, y)
    t = 6
    triangle_x = get_subdivision_triangle(0.6, x)
    triangle_y = get_subdivision_triangle(0.6, y)
    left_x_points, right_x_points = get_subdivision_points(triangle_x)
    left_y_points, right_y_points = get_subdivision_points(triangle_y)
    bezier_x, bezier_y = bezier_curve(x, y)
    plt.scatter(left_x_points,left_y_points)
    plt.scatter(right_x_points,right_y_points,)
```

如下为计算所得 x 坐标和 y 坐标数字三角矩阵。

	0	1	2	3	4	5	6
0	0.00000	0.03333	0.06667	0.10000	0.13333	0.16667	0.20000
1	0.02000	0.05333	0.08667	0.12000	0.15333	0.18667	0.00000
2	0.04000	0.07333	0.10667	0.14000	0.17333	0.00000	0.00000
3	0.06000	0.09333	0.12667	0.16000	0.00000	0.00000	0.00000
4	0.08000	0.11333	0.14667	0.00000	0.00000	0.00000	0.00000
5	0.10000	0.13333	0.00000	0.00000	0.00000	0.00000	0.00000
6	0.12000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000

	0	1	2	3	4	5	6
0	-10.00000	10.00000	10.00000	-20.00000	10.00000	10.00000	-10.00000
1	2.00000	10.00000	-8.00000	-2.00000	10.00000	-2.00000	0.00000
2	6.80000	-0.80000	-4.40000	5.20000	2.80000	0.00000	0.00000
3	2.24000	-2.96000	1.36000	3.76000	0.00000	0.00000	0.00000
4	-0.88000	-0.36800	2.80000	0.00000	0.00000	0.00000	0.00000
5	-0.57280	1.53280	0.00000	0.00000	0.00000	0.00000	0.00000
6	0.69056	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000

下图为计算所得的左曲线控制点和右曲线控制点。



4 习题四 升阶算法

升阶后散点图如下: