数值分析上机报告
第四章

院 (系) 名 称：　微电子学院

学 生 姓 名：　周玉乾

学　　号：　220205764

二〇二〇年十二月

第四章

4.1 3 次样条插值函数

1. 问题

(1) 编制求第一型 3 次样条插值函数的通用程序;

(2) 已知汽车门曲线型值点的数据如下:

i	0	1	2	3	4	5	6	7	8	9	10
x_i	0	1	2	3	4	5	6	7	8	9	10
y_i	2.51	3.30	4.04	4.70	5.22	5.54	5.78	5.40	5.57	5.70	5.80

端点条件为 $y_0' = 0.8$，$y_{10}' = 0.2$，用所编程序求车门的三次样条插值函数 $S(x)$，并打印出 $S(i+0.5), i = 0, 1, \cdots, 9$。

2. 分析

根据 3 次样条插值的定义，插值函数 $S(x)$ 满足：1. $S(x)$ 在每一个小区间上式 3 次多项式，2. $S(x)$ 在区间 $[a, b]$ 上具有连续 2 阶导数，3. $S(x)$ 经过每一个给定的插值节点。

在区间 $[a, b]$ 上有 $n+1$ 个插值节点，因此可以设每个小区间上的插值函数 $S_i(x)$ 为:

$$S_i(x) = a_i x^3 + b_i x^2 + c_i x + d_i \tag{4.1}$$

在 n 个区间上有 n 个插值函数，每个插值函数有 4 个参数，因此需要 $4n$ 个不相关的方程来求解:

(1) 每个插值函数都会经过它所在小区间上的插值节点，即小区间的两个端点，有 $2n$ 个方程:

$$S_i(x_i) = y_i, \quad i \in [0, n-1] \tag{4.2}$$

$$S_i(x_{i+1}) = y_{i+1}, \quad i \in [0, n-1] \tag{4.3}$$

(2) 一阶导数连续，即两个相邻的插值函数连接点处的一阶导数相等，有 $n-1$ 个方程:

$$S_i'(x_{i+1}) = S_{i+1}'(x_{i+1}), \quad i \in [0, n-2] \tag{4.4}$$

(3) 二阶导数连续，即两个相邻的插值函数连接点处的二阶导数相等，有 $n-1$ 个方程：

$$S_i''(x_{i+1}) = S_{i+1}''(x_{i+1}), \quad i \in [0, n-2] \tag{4.5}$$

(4) 第一型边界条件，已知两个端点处的一阶导数值，有 2 个方程：

$$S_0'(x_0) = y_0' \tag{4.6}$$

$$S_{n-1}'(x_n) = y_n' \tag{4.7}$$

$4n$ 个方程，使用列主元高斯法求解方程，得到最终的插值函数。

3. 程序

```python
import numpy as np
import matplotlib.pyplot as plt
from pylab import mpl
import sys, os

'''
DESCRIPTION:
PARAM {*} x N+1 个插值点
PARAM {*} y N+1 个插值点
RETURN {*} N
'''
def Prejudgment(x, y):
    n1 = len(x)
    n2 = len(y)
    if n1 != n2:
        print('x 与 y 长度不相等')
        sys.exit()

    n = n1-1
    return n

'''
DESCRIPTION: 求三次样条差值的 4N 个方程
PARAM: {x[0,N], y[0,N]} N+1 个插值点
PARAM: Type 三次样条边界条件 1 OR 2 OR 3
RETURN {A, B} [A0 B0 C0 D0 A1 B1 C1 D1 ... A(N-1) B(N-1) C(N-1) D(N-1)] = [B] 形式的方程组
'''
def calculateEquationParameters(x, y, Type=1, dy0=0, dyn=0):
    n = Prejudgment(x, y)

    parameterA = []
    parameterB = []
```

```
33
34          # S_I(x_I) = y_I
35          # S_I(x_{I+1}) = y_{I+1}
36          # 0 <= I <= N-1
37          for i in range(0, n):
38              # S_I(x_I) = y_I
39              data = np.zeros(n*4)
40              data[i*4] = pow(x[i], 3)
41              data[i*4+1] = pow(x[i], 2)
42              data[i*4+2] = x[i]
43              data[i*4+3] = 1
44              parameterA.append(data.tolist())
45              parameterB.append(y[i])
46
47              # S_I(x_{I+1}) = y_{I+1}
48              data1 = np.zeros(n*4)
49              data1[i*4] = pow(x[(i+1)], 3)
50              data1[i*4+1] = pow(x[(i+1)], 2)
51              data1[i*4+2] = x[(i+1)]
52              data1[i*4+3] = 1
53              parameterA.append(data1.tolist())
54              parameterB.append(y[i+1])
55
56          # S'_I(x_{I+1}) = S'_{I+1}(x_{I+1})
57          # 0 <= I <= N-2
58          for i in range(0, n-1):
59              data = np.zeros(n*4)
60
61              data[i*4] = 3 * pow(x[i+1], 2)
62              data[i*4+1] = 2 * x[i+1]
63              data[i*4+2] = 1
64
65              data[(i+1)*4] = -3 * pow(x[i+1], 2)
66              data[(i+1)*4+1] = -2 * x[i+1]
67              data[(i+1)*4+2] = -1
68
69              parameterA.append(data.tolist())
70              parameterB.append(0)
71
72          # S''_I(x_{I+1}) = S''_{I+1}(x_{I+1})
73          # 0 <= I <= N-2
74          for i in range(0, n-1):
75              data = np.zeros(n*4)
76
77              data[i*4] = 6 * x[i+1]
78              data[i*4+1] = 2
79
80              data[(i+1)*4] = -6 * x[i+1]
```

```
            data[(i+1)*4+1] = -2

            parameterA.append(data.tolist())
            parameterB.append(0)

        if Type == 1:
            # S'_0(x_0) = Y'_0
            data = np.zeros(n*4)
            data[0] = 3 * pow(x[0], 2)
            data[1] = 2 * x[0]
            data[2] = 1
            parameterA.append(data.tolist())
            parameterB.append(dy0)

            # S'_{N-1}(x_N) = Y'_N
            data = np.zeros(n*4)
            data[(n-1)*4] = 3 * pow(x[n], 2)
            data[(n-1)*4+1] = 2 * x[n]
            data[(n-1)*4+2] = 1

            parameterA.append(data.tolist())
            parameterB.append(dyn)
        elif Type == 2:
            # S''(A) = S''(B) = 0

            # S''_0(x_0) = 0
            data = np.zeros(n*4)
            data[0] = 6 * x[0]
            data[1] = 2
            parameterA.append(data.tolist())
            parameterB.append(0)

            # S''_{N-1}(x_N) = 0
            data = np.zeros(n*4)
            data[(n-1)*4] = 6 * x[n]
            data[(n-1)*4+1] = 2
            parameterA.append(data.tolist())
            parameterB.append(0)

        elif Type == 3:
            # S'(A) = S'(B) AND # S''(A) = S''(B)
            pass
        else:
            print('Error! Unknown "Type" Value!')

        return parameterA, parameterB

    """
```

```
129     功能：根据所给参数，计算三次函数的函数值：
130     参数：OriginalInterval为原始x的区间，parameters为二次函数的系数，x为自变量
131     返回值：为函数的因变量
132     """
133     def calculate(OriginalInterval, paremeters, x):
134         n = int(len(paremeters)/4)
135         result=[]
136         for data_x in x:
137 
138             Interval = 0
139             if data_x <= OriginalInterval[0]:
140                 Interval = 0
141             elif data_x >= OriginalInterval[-1]:
142                 Interval = n-1
143             else:
144                 for i in range(0,n):
145                     if data_x >= OriginalInterval[i] and data_x < OriginalInterval[i+1]:
146                         Interval = i
147                         break
148 
149             result.append(paremeters[Interval*4+0]*data_x*data_x*data_x+paremeters[
                    Interval*4+1]*data_x*data_x+paremeters[Interval*4+2]*data_x+paremeters[
                    Interval*4+3])
150         return result
151 
152     """
153     功能：将函数绘制成图像
154     参数：data_x,data_y为离散的点.new_data_x,new_data_y为由拉格朗日插值函数计算的值。x为
            函数的预测值。
155     返回值：空
156     """
157     def Draw(data_x,data_y,new_data_x,new_data_y, title):
158             plt.plot(new_data_x, new_data_y, label="拟合曲线", color="black")
159             plt.scatter(data_x,data_y, label="离散数据",color="red")
160             mpl.rcParams['font.sans-serif'] = ['SimHei']
161             mpl.rcParams['axes.unicode_minus'] = False
162             plt.title("三次样条函数")
163             plt.legend(loc="upper left")
164             plt.savefig(os.path.join(os.path.dirname(os.path.abspath(__file__)), title+
                    '.png'), dpi=300)
165             plt.show()
166 
167     def PrintS(parameterX):
168         n = int(len(parameterX)/4)
169         print('S(x) = ')
170         for i in range(0,n):
171             print("%.6g & %.6g & %.6g & %.6g \\\\" % (parameterX[i*4], parameterX[i
                    *4+1], parameterX[i*4+2], parameterX[i*4+3]))
```

```
172         print('\n\n')
173 
174     def main():
175         x = [0,   1,    2,    3,    4,    5,    6,    7,    8,    9,    10]
176         y = [2.51, 3.30, 4.04, 4.70, 5.22, 5.54, 5.78, 5.40, 5.57, 5.70, 5.80]
177         dy0 = 0.8
178         dyn = 0.2
179         parameterA, parameterB = calculateEquationParameters(x, y, 1, dy0, dyn)
180         parameterX = MGauss_Caculate(parameterA, parameterB)
181 
182         PrintS(parameterX)
183 
184         # 画图
185         new_data_x = np.arange(x[0]-0.5, x[-1]+0.6, 0.1)
186         new_data_y = calculate(x, parameterX, new_data_x)
187         Draw(x, y, new_data_x, new_data_y, '三次样条插值')
188 
189         # 打印
190         new_data_x = np.arange(0.5, 10.5, 1)
191         new_data_y = calculate(x, parameterX, new_data_x)
192         # F4_5 = CALCULATE(PARAMETERX[8:12], [4.5])
193         print(new_data_x)
194         for i,data in enumerate(new_data_y):
195             print("%.6g & " % data)
196 
197     if __name__ == "__main__":
198 
199         # 获取当前文件路径
200         current_path = os.path.abspath(__file__)
201         sys.path.append(os.path.abspath(os.path.join(os.path.dirname(current_path), '
                ../')))
202         # PRINT(SYS.PATH)
203         # 调用 CHAPTER3 中的列主元高斯消去法
204         from chapter3.q3 import MGauss_Caculate
205 
206         main()
```

4. 算例

对题目中的数据进行三次样条插值：

i	0	1	2	3	4	5	6	7	8	9	10
x_i	0	1	2	3	4	5	6	7	8	9	10
y_i	2.51	3.30	4.04	4.70	5.22	5.54	5.78	5.40	5.57	5.70	5.80

得到的插值函数系数为：

$$\begin{bmatrix} -0.00851404 & -0.00148596 & 0.8 & 2.51 \\ -0.00445789 & -0.0136544 & 0.812168 & 2.50594 \\ -0.00365441 & -0.0184753 & 0.82181 & 2.49952 \\ -0.0409245 & 0.316955 & -0.184482 & 3.50581 \\ 0.107352 & -1.46237 & 6.93281 & -5.98391 \\ -0.268485 & 4.17519 & -21.255 & 40.9958 \\ 0.426587 & -8.33611 & 53.8128 & -109.14 \\ -0.267865 & 6.24739 & -48.2717 & 129.057 \\ 0.0548723 & -1.4983 & 13.6939 & -36.1842 \\ 0.0583759 & -1.5929 & 14.5453 & -38.7383 \end{bmatrix}$$

题目中要求的 $S(i+0.5), i = 0, 1, \cdots, 9$ 值如表4.1所示，插值函数的图像如图4.1所示。

表 4.1 $S(i+0.5), i = 0, 1, \cdots, 9$

i	0	1	2	3	4
x_i	0.5	1.5	2.5	3.5	4.5
$S(x)$	2.90856	3.67843	4.38147	4.98819	5.38328
i	5	6	7	8	9
x_i	5.5	6.5	7.5	8.5	9.5
$S(x)$	5.7237	5.59441	5.42989	5.65977	5.7323

5. 结论

(1) 编写程序实现了第一型边界条件和第二型边界条件下的三次样条插值计算；
(2) 三次样条插值得到的函数二阶导数连续，因此其光滑性较好；
(3) 目前算法使用的是待定系数法求解，当插值节点较多时，计算量较大，应加以改进。

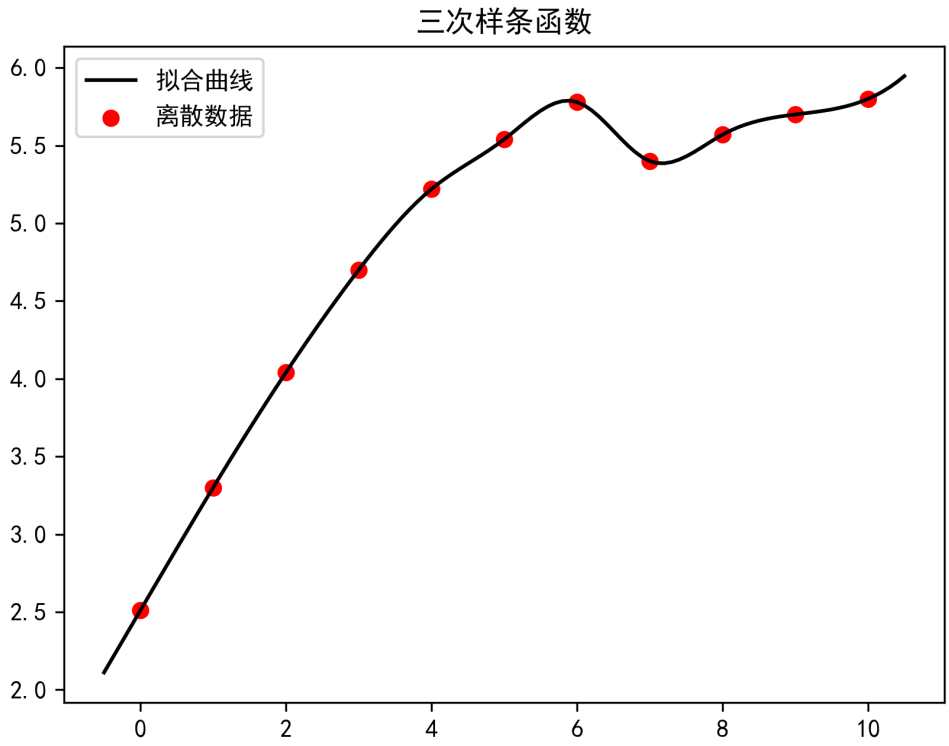

图 4.1　三次样条插值图像

4.2 离散数据的最佳平方逼近