



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 $S(i + 0.5), i = 0, 1, \dots, 9$ 。

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:

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

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:

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

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

- (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:

$$S'_i(x_{i+1}) = S'_{i+1}(x_{i+1}), \quad i \in [0, n - 2] \quad (4.4)$$

- (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：

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

- (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：

$$S_0'(x_0) = y_0' \quad (4.6)$$

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

$4n$ 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

```

1  import numpy as np
2  import matplotlib.pyplot as plt
3  from pylab import mpl
4  import sys, os
5
6  '''
7  DESCRIPTION:
8  PARAM {*} X N+1 个插值点
9  PARAM {*} Y N+1 个插值点
10 RETURN {*} N
11 '''
12 def Prejudgment(x, y):
13     n1 = len(x)
14     n2 = len(y)
15     if n1 != n2:
16         print('x 与 y 长度不相等')
17         sys.exit()
18
19     n = n1-1
20     return n
21
22 '''
23 DESCRIPTION: 求三次样条差值的 4N 个方程
24 PARAM: {x[0,N], y[0,N]} N+1 个插值点
25 PARAM: TYPE 三次样条边界条件 1 OR 2 OR 3
26 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
27 '''
28 def calculateEquationParameters(x, y, Type=1, dy0=0, dyn=0):
29     n = Prejudgment(x, y)
30
31     parameterA = []
32     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]

```

```

81         data[(i+1)*4+1] = -2
82
83         parameterA.append(data.tolist())
84         parameterB.append(0)
85
86     if Type == 1:
87         # S'_0(x_0) = y'_0
88         data = np.zeros(n*4)
89         data[0] = 3 * pow(x[0], 2)
90         data[1] = 2 * x[0]
91         data[2] = 1
92         parameterA.append(data.tolist())
93         parameterB.append(dy0)
94
95         # S'_{N-1}(x_N) = y'_N
96         data = np.zeros(n*4)
97         data[(n-1)*4] = 3 * pow(x[n], 2)
98         data[(n-1)*4+1] = 2 * x[n]
99         data[(n-1)*4+2] = 1
100
101         parameterA.append(data.tolist())
102         parameterB.append(dyn)
103     elif Type == 2:
104         # S''(A) = S''(B) = 0
105
106         # S''_0(x_0) = 0
107         data = np.zeros(n*4)
108         data[0] = 6 * x[0]
109         data[1] = 2
110         parameterA.append(data.tolist())
111         parameterB.append(0)
112
113         # S''_{N-1}(x_N) = 0
114         data = np.zeros(n*4)
115         data[(n-1)*4] = 6 * x[n]
116         data[(n-1)*4+1] = 2
117         parameterA.append(data.tolist())
118         parameterB.append(0)
119
120     elif Type == 3:
121         # S'(A) = S'(B) AND # S''(A) = S''(B)
122         pass
123     else:
124         print('Error! Unknown "Type" Value!')
125
126     return parameterA, parameterB
127
128 """

```

```

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, parameters, x):
134     n = int(len(parameters)/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(parameters[Interval*4+0]*data_x*data_x*data_x+parameters[
150                 Interval*4+1]*data_x*data_x+parameters[Interval*4+2]*data_x+parameters[
151                     Interval*4+3])
152
153     return result
154
155 """
156 功能：将函数绘制成图像
157 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
158 函数的预测值。
159 返回值：空
160 """
161 def Draw(data_x,data_y,new_data_x,new_data_y, title):
162     plt.plot(new_data_x, new_data_y, label="拟合曲线", color="black")
163     plt.scatter(data_x,data_y, label="离散数据",color="red")
164     mpl.rcParams['font.sans-serif'] = ['SimHei']
165     mpl.rcParams['axes.unicode_minus'] = False
166     plt.title("三次样条函数")
167     plt.legend(loc="upper left")
168     plt.savefig(os.path.join(os.path.dirname(os.path.abspath(__file__)), title+
169         '.png'), dpi=300)
170     plt.show()
171
172 def PrintS(parameterX):
173     n = int(len(parameterX)/4)
174     print('S(x) = ')
175     for i in range(0,n):
176         print("%.6g & %.6g & %.6g & %.6g \\\\" % (parameterX[i*4], parameterX[i
177             *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         ../')))
203     # PRINT(SYS.PATH)
204     # 调用 CHAPTER3 中的列主元高斯消去法
205     from chapter3.q3 import MGauss_Caculate
206
207     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 $S(i + 0.5), i = 0, 1, \dots, 9$ 值如表4.1所示，插值函数的图像如图4.1所示。

表 4.1 $S(i + 0.5), i = 0, 1, \dots, 9$

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。

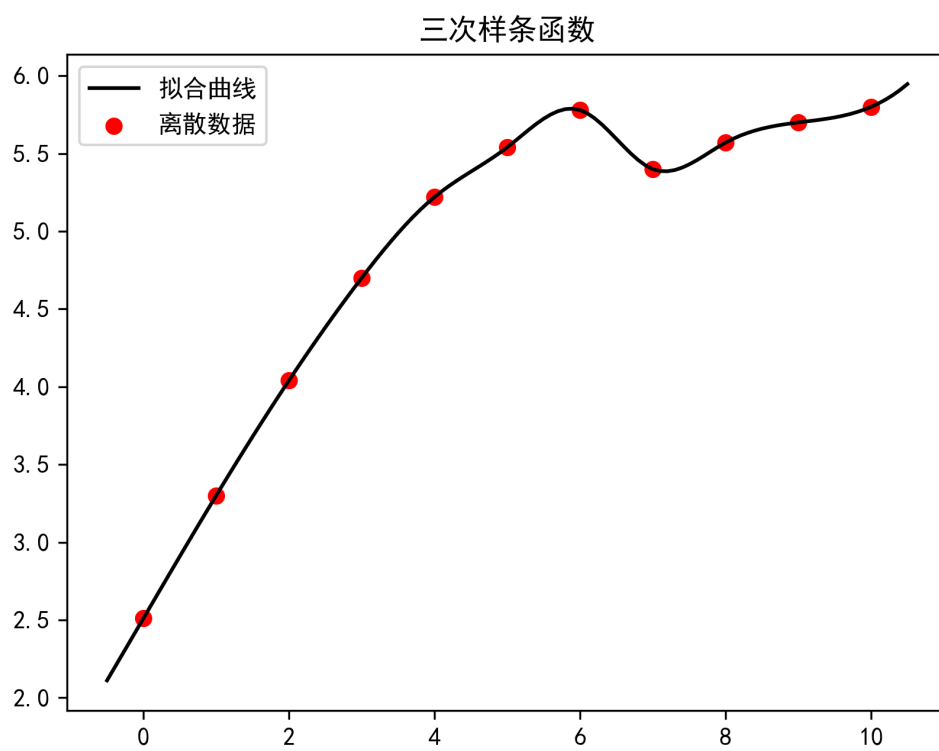


图 4.1 三次样条插值图像

4.2 离散数据的最佳平方逼近

1. 问题

一种商品的需求量和其价格有一定关系，先对一定时期内的商品价格 x 与需求量 y 进行观察，取得如下的样本数据，如表4.2所示。

表 4.2 样本数据

价格	2	3	4	5	6	7	8	9	10	11
需求量	58	50	44	38	34	30	29	26	25	24

- (1) 对上述数据，分别求出其 2, 3, 4 次最佳平方逼近多项式。画出图形，并比较拟合误差：

$$Q = \sum_{k=1}^n (q(x_k) - y_k)^2 \quad (4.8)$$

- (2) 假设拟合函数分别为：

$$a + \frac{b}{x}, \quad a + b \ln x, \quad a e^{bx}, \quad \frac{1}{a + bx} \quad (4.9)$$

分别求出 a, b 的值。绘图，计算误差，并比较优劣。

2. 分析

本题是一个离散数据的最佳平方逼近问题，对于这类问题，首先要将拟合函数化为标准形式，即 $p(x) = \sum_{k=1}^n (c_i \varphi_i(x))$ ，然后求出正规方程，通过求解正规方程得到拟合函数系数 c_i ，如有需要再将标准形式的拟合函数化为需要的形式。

在第二问中，四个拟合函数都不是标准形式。

1. 对于 $y = a + b/x$ ，令 $t = 1/x$ ，则 $y = a + bt$ ；
2. 对于 $y = a + b \ln x$ ，令 $t = \ln x$ ，则 $y = a + bt$ ；
3. 对于 $y = a e^{bx}$ ，两边取对数，得到 $\ln y = \ln a + bx$ ；
4. 对于 $y = 1/(a + bx)$ ，令 $t = 1/y$ ，则 $t = a + bx$ 。

3. 程序

```
1 # 离散数据的最佳平方逼近
2 import numpy as np
3 import sys, os
4 import matplotlib.pyplot as plt
5 from pylab import mpl
6 import math
```

```

7
8     '''
9     DESCRIPTION:
10    PARAM {*} X N+1 个插值点
11    PARAM {*} Y N+1 个插值点
12    RETURN {*} N
13    '''
14    def PrejudgmentShape(x, y):
15        n1 = len(x)
16        n2 = len(y)
17        if n1 != n2:
18            print('x 与 y 长度不相等')
19            sys.exit()
20
21        n = n1-1
22        return n
23
24    '''
25    DESCRIPTION: 求最佳平方逼近的正规方程
26    PARAM {*} M M次最佳平方逼近
27    PARAM {*} X N个自变量x
28    PARAM {*} Y N个因变量Y
29    RETURN {*} (A, B)
30    '''
31    def GetNormalEquation(m, x, y):
32        A = []
33        b = []
34        PrejudgmentShape(x, y)
35        for j in range(0,m+1):
36            AAline = []
37            for i in range(0, m+1):
38                tmp = np.dot(np.power(x, j), np.power(x, i))
39                AAline.append(tmp)
40            A.append(AAline)
41            tmp = np.dot(y, np.power(x, j))
42            b.append(tmp)
43        return A, b
44
45    '''
46    DESCRIPTION: 获取最佳平方逼近的多项式
47    PARAM {*} A 正规方程系数A
48    PARAM {*} B 正规方程系数B
49    RETURN {*} C 多项式系数,  $q(x) = c_0 + c_1x + c_2x^2 + \dots + c_Mx^M$ 
50    '''
51    def GetExpression(A, b):
52        c = MGauss_Caculate(A, b)
53        return c
54

```

```

55     '''
56     DESCRIPTION: 计算最佳平方逼近多项式
57     PARAM {*} M M次最佳平方逼近
58     PARAM {*} X N个自变量x
59     PARAM {*} Y N个因变量y
60     RETURN {*} c 最佳平方逼近多项式系数,  $q(x) = c_0 + c_1x + c_2x^2 + \dots + c_Mx^M$ 
61     '''
62     def GetLeastSquaresApproximationExpression(m, x, y):
63         A, b = GetNormalEquation(m, x, y)
64         c = GetExpression(A, b)
65         return c
66
67     '''
68     DESCRIPTION: 计算多项式,  $q(x) = c_0 + c_1x + c_2x^2 + \dots + c_Mx^M$ 
69     PARAM {*} c 最佳平方逼近多项式系数
70     RETURN {*} 多项式结果
71     '''
72     def CaculateExpression(c, x):
73         res = 0
74         for i, ci in enumerate(c):
75             res = res + ci * pow(x, i)
76         return res
77
78     '''
79     DESCRIPTION: 计算多项式, x 从 x0 到 xn,  $q(x) = c_0 + c_1x + c_2x^2 + \dots + c_Mx^M$ 
80     PARAM {*} c 最佳平方逼近多项式系数
81     PARAM {LIST} xn
82     RETURN {*}
83     '''
84     def CaculateInterval(c, xn):
85         res = []
86         for x in xn:
87             res.append(CaculateExpression(c, x))
88         return res
89
90     """
91     功能: 将函数绘制成图像
92     参数: DATA_X, DATA_Y 为原始值, NEW_DATA_X, NEW_DATA_Y 为预测计算的值。
93     返回值: 空
94     """
95     def Draw(data_x,data_y,new_data_x,new_data_y, title):
96         plt.plot(new_data_x, new_data_y, label="拟合曲线", color="black")
97         plt.scatter(data_x,data_y, label="离散数据",color="red")
98         mpl.rcParams['font.sans-serif'] = ['SimHei']
99         mpl.rcParams['axes.unicode_minus'] = False
100         plt.title(title)
101         plt.legend(loc="upper right")
102         plt.savefig(os.path.join(os.path.dirname(os.path.abspath(__file__)), title+'.

```

```

        png'), dpi=300)
103     plt.show()
104
105     '''
106     DESCRIPTION: 计算拟合误差  $Q = \sum_{k=1}^N (q(x_k) - y_k)^2$ 
107     PARAM {*} c 最佳平方逼近多项式系数,  $q(x) = c_0 + c_1x + c_2x^2 + \dots + c_mx^m$ 
108     PARAM {*} x
109     PARAM {*} y
110     RETURN {*} Q
111     '''
112     def GetFittingError(c, x, y):
113         Err = 0
114         for n, xn in enumerate(x):
115             qx = CaculateExpression(c, xn)
116             Err += pow((qx - y[n]), 2)
117         return Err
118
119     '''
120     DESCRIPTION: 别求出其 2, 3, 4 次最佳平方逼近多项式. 画出图形, 并比较拟合误差
121     PARAM {*}
122     RETURN {*}
123     '''
124     def Q1():
125         x = [2, 3, 4, 5, 6, 7, 8, 9, 10, 11]
126         y = [58, 50, 44, 38, 34, 30, 29, 26, 25, 24]
127
128         result_file = os.path.join(os.path.dirname(os.path.abspath(__file__)), '最佳平方逼近_result.txt')
129
130         if os.path.exists(result_file):
131             os.remove(result_file)
132
133         # 最佳平方逼近
134         for m in range(2, 5):
135             c = GetLeastSquaresApproximationExpression(m, x, y)
136             # 画图
137             new_x = np.arange(x[0]-0.5, x[-1]+0.6, 0.1)
138             new_y = CaculateInterval(c, new_x)
139             Draw(x, y, new_x, new_y, "{0}次最佳平方逼近".format(m))
140             # 拟合误差
141             FittingErr = GetFittingError(c, x, y)
142             # 写入文件
143             with open(result_file, "a+") as fo:
144                 fo.write("\n
145                     \\*****\\n")
146                 fo.write("{0} 次最佳平方逼近, m = {1}\n".format(m, m))
147                 fo.write("x = ")
148                 for xi in x:

```

```

148         fo.write(str(xi))
149         fo.write("\t")
150     fo.write("\n")
151     fo.write("y = ")
152     for yi in y:
153         fo.write(str(yi))
154         fo.write("\t")
155     fo.write("\n")
156     fo.write("近似函数: q(x) = c0 + c1*x + c2*x^2 + ... + cm*x^m \n")
157     fo.write("c = ")
158     for ci in c:
159         fo.write(str(format(ci, '.6g')))
160         fo.write("\t")
161     fo.write("\n")
162     fo.write("拟合误差: {0}\n".format(format(FittingErr, '.8g')))
163     fo.write("\n\*****\n")
164
165     # A+B÷x近似
166     def Q2_1():
167         x = [2, 3, 4, 5, 6, 7, 8, 9, 10, 11]
168         y = [58, 50, 44, 38, 34, 30, 29, 26, 25, 24]
169
170         result_file = os.path.join(os.path.dirname(os.path.abspath(__file__)), 'a+b÷x近
171             似_result.txt')
172
173         # A + B/x
174         x1 = [1.0/xi for xi in x]
175         c = GetLeastSquaresApproximationExpression(1, x1, y)
176         # 画图
177         new_x = np.arange(x[0]-0.5, x[-1]+0.6, 0.1)
178         new_x1 = [1.0/xi for xi in new_x]
179         new_y = CaculateInterval(c, new_x1)
180         Draw(x, y, new_x, new_y, "a+b÷x近似")
181         # 拟合误差
182         FittingErr = GetFittingError(c, x1, y)
183         # 写入文件
184         with open(result_file, "w+") as fo:
185             fo.write("\n\*****\n")
186             fo.write("a+b÷x近似\n")
187             fo.write("x = ")
188             for xi in x:
189                 fo.write(str(xi))
190                 fo.write("\t")
191             fo.write("\n")
192             fo.write("y = ")
193             for yi in y:

```

```

193         fo.write(str(yi))
194         fo.write("\t")
195     fo.write("\n")
196     fo.write("近似函数: q(x) = {} + {} / x \n".format(c[0], c[1]))
197     fo.write("拟合误差: {0}\n".format(FittingErr))
198     fo.write("\n\*****\n")
199
200     # A+BLNX近似
201     def Q2_2():
202         x = [2, 3, 4, 5, 6, 7, 8, 9, 10, 11]
203         y = [58, 50, 44, 38, 34, 30, 29, 26, 25, 24]
204
205         result_file = os.path.join(os.path.dirname(os.path.abspath(__file__)), 'a+blnx
                近似_result.txt')
206
207         # A + BLNX
208         x1 = [math.log(xi) for xi in x]
209         c = GetLeastSquaresApproximationExpression(1, x1, y)
210         # 画图
211         new_x = np.arange(x[0]-0.5, x[-1]+0.6, 0.1)
212         new_x1 = [math.log(xi) for xi in new_x]
213         new_y = CaculateInterval(c, new_x1)
214         Draw(x, y, new_x, new_y, "a+blnx近似")
215         # 拟合误差
216         FittingErr = GetFittingError(c, x1, y)
217         # 写入文件
218         with open(result_file, "w+") as fo:
219             fo.write("\n\*****\n")
220             fo.write("a+blnx近似\n")
221             fo.write("x = ")
222             for xi in x:
223                 fo.write(str(xi))
224                 fo.write("\t")
225             fo.write("\n")
226             fo.write("y = ")
227             for yi in y:
228                 fo.write(str(yi))
229                 fo.write("\t")
230             fo.write("\n")
231             fo.write("近似函数: q(x) = {} + {} ln(x) \n".format(c[0], c[1]))
232             fo.write("拟合误差: {0}\n".format(FittingErr))
233             fo.write("\n\*****\n")
234
235     '''
236     DESCRIPTION: 计算拟合误差  $Q = \sum_{k=1}^N (q(x_k) - y_k)^2$ 
237     PARAM {*} C 最佳平方逼近多项式系数,  $q(x) = A \cdot \exp(Bx)$ 
238     PARAM {*} X

```

```

239     PARAM {*} Y
240     RETURN {*} Q
241     '''
242     def GetFittingError_q2_3(c, x, y):
243         Err = 0
244         for n, xn in enumerate(x):
245             qx1 = CaculateExpression(c, xn)
246             qx = math.exp(qx1)
247             Err += pow((qx - y[n]), 2)
248         return Err
249
250     # A*EXP(BX)近似
251     def Q2_3():
252         x = [2, 3, 4, 5, 6, 7, 8, 9, 10, 11]
253         y = [58, 50, 44, 38, 34, 30, 29, 26, 25, 24]
254
255         result_file = os.path.join(os.path.dirname(os.path.abspath(__file__)), 'a*exp(
                bx)近似_result.txt')
256
257         # A + BLNX
258         y1 = [math.log(yi) for yi in y]
259         c = GetLeastSquaresApproximationExpression(1, x, y1)
260         # 画图
261         new_x = np.arange(x[0]-0.5, x[-1]+0.6, 0.1)
262         new_y1 = CaculateInterval(c, new_x)
263         new_y = [math.exp(yi) for yi in new_y1]
264         Draw(x, y, new_x, new_y, "a*exp(bx)近似")
265         # 拟合误差
266         FittingErr = GetFittingError_q2_3(c, x, y)
267         # 写入文件
268         with open(result_file, "w+") as fo:
269             fo.write("\n\\*****\\n")
270             fo.write("a*exp(bx)近似\n")
271             fo.write("x = ")
272             for xi in x:
273                 fo.write(str(xi))
274                 fo.write("\t")
275             fo.write("\n")
276             fo.write("y = ")
277             for yi in y:
278                 fo.write(str(yi))
279                 fo.write("\t")
280             fo.write("\n")
281             fo.write("近似函数: q(x) = {0} * exp({1} * x) \n".format(math.exp(c[0]), c
                [1]))
282             fo.write("拟合误差: {0}\n".format(FittingErr))
283             fo.write("\\*****\\n")

```



```

284
285     '''
286     DESCRIPTION: 计算拟合误差  $Q = \sum_{k=1}^N (Q(x_k) - y_k)^2$ 
287     PARAM {*} c 最佳平方逼近多项式系数,  $Q(x) = 1/(A+Bx)$ 
288     PARAM {*} x
289     PARAM {*} y
290     RETURN {*} Q
291     '''
292     def GetFittingError_q2_4(c, x, y):
293         Err = 0
294         for n, xn in enumerate(x):
295             qx1 = CaculateExpression(c, xn)
296             qx = 1.0/qx1
297             Err += pow((qx - y[n]), 2)
298         return Err
299
300     #  $1/(A+Bx)$ 近似
301     def Q2_4():
302         x = [2, 3, 4, 5, 6, 7, 8, 9, 10, 11]
303         y = [58, 50, 44, 38, 34, 30, 29, 26, 25, 24]
304
305         result_file = os.path.join(os.path.dirname(os.path.abspath(__file__)), '1/(a+
306             bx)近似_result.txt')
307
308         #  $A + BLNX$ 
309         y1 = [1.0/yi for yi in y]
310         c = GetLeastSquaresApproximationExpression(1, x, y1)
311         # 画图
312         new_x = np.arange(x[0]-0.5, x[-1]+0.6, 0.1)
313         new_y1 = CaculateInterval(c, new_x)
314         new_y = [1.0/yi for yi in new_y1]
315         Draw(x, y, new_x, new_y, "1/(a+bx)近似")
316         # 拟合误差
317         FittingErr = GetFittingError_q2_4(c, x, y)
318         # 写入文件
319         with open(result_file, "w+") as fo:
320             fo.write("\n\*****\n\n")
321             fo.write("1/(a+bx)近似\n")
322             fo.write("x = ")
323             for xi in x:
324                 fo.write(str(xi))
325                 fo.write("\t")
326             fo.write("\n")
327             fo.write("y = ")
328             for yi in y:
329                 fo.write(str(yi))
330                 fo.write("\t")

```

```

330         fo.write("\n")
331         fo.write("近似函数: q(x) = 1 ÷ ( {0} + {1} * x) \n".format(math.exp(c[0]),
332                                c[1]))
332         fo.write("拟合误差: {0}\n".format(FittingErr))
333         fo.write("\n\*****\n\n")
334
335     def main():
336         Q1()
337         Q2_1()
338         Q2_2()
339         Q2_3()
340         Q2_4()
341
342     if __name__ == "__main__":
343
344         # 获取当前文件路径
345         current_path = os.path.abspath(__file__)
346         sys.path.append(os.path.abspath(os.path.join(os.path.dirname(current_path), '
347             ../')))
348         # PRINT(SYS.PATH)
349         # 调用 CHAPTER3 中的列主元高斯消去法
350         from chapter3.q3 import MGauss_Caculate
351
352     main()

```

4. 算例

对于题目中的第一问，分别求出 2, 3, 4 次最佳平方逼近多项式的拟合函数如下，拟合误差分别为 3.2166667, 1.5102564, 1.4679487，拟合函数的图像如图4.2所示。

$$p(x) = 74.3258 - 9.31136 * x + 0.435606 * x^2 \quad (4.10)$$

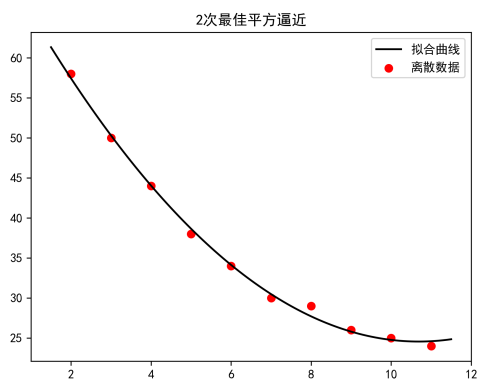
$$p(x) = 78.5424 - 11.9462 * x + 0.893939 * x^2 - 0.0235043 * x^3 \quad (4.11)$$

$$p(x) = 76.9924 - 10.6129 * x + 0.520542 * x^2 + 0.0181624 * x^3 - 0.00160256 * x^4 \quad (4.12)$$

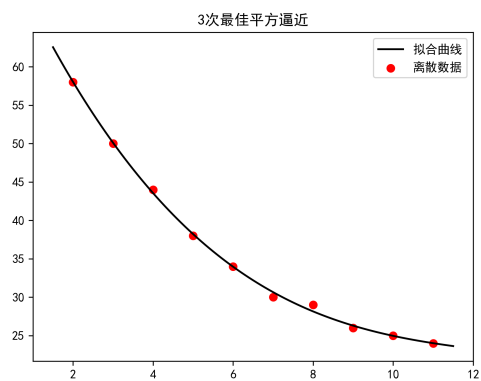
对于题目中的第二问，拟合函数如下，拟合的图像如图4.3所示，拟合误差分别为 57.302270, 8.7788712, 41.844596, 7.7324292。

$$q(x) = 18.160410958809152 + \frac{87.330000932616}{x} \quad (4.13)$$

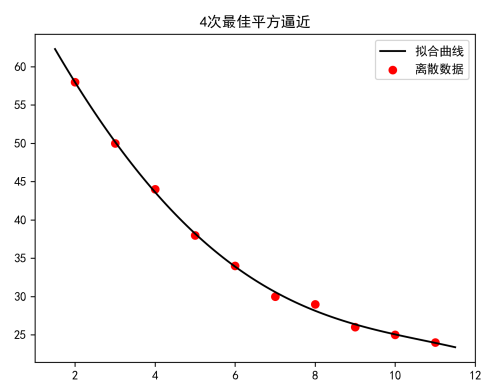
$$q(x) = 72.13901063630205 - 20.762410852503 \ln(x) \quad (4.14)$$



(a) 2 次最佳平方逼近



(b) 3 次最佳平方逼近

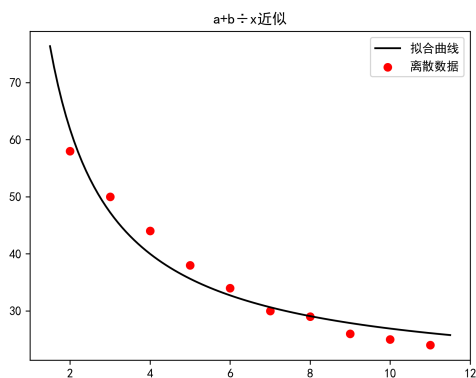


(c) 4 次最佳平方逼近

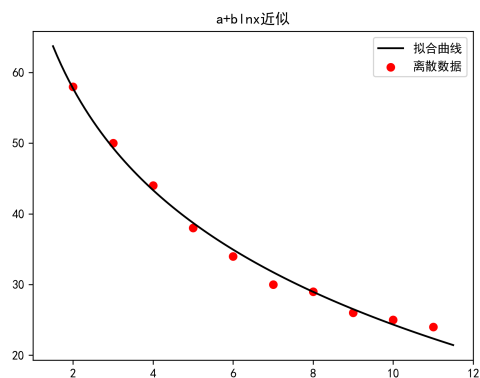
图 4.2 最佳平方逼近

$$q(x) = 65.29387947714481 * e^{-0.09915171210523323*x} \quad (4.15)$$

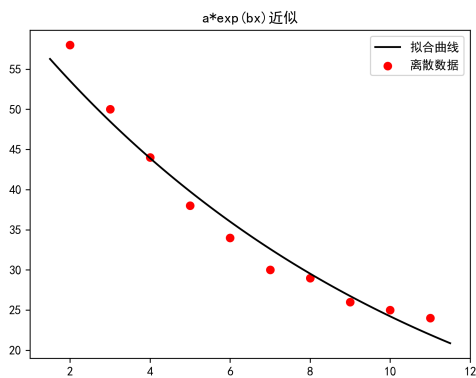
$$q(x) = \frac{1}{1.012042106641784 + 0.0028298266117981353 * x} \quad (4.16)$$



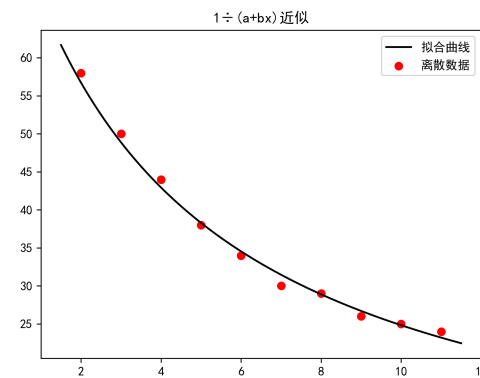
(a) $y = a + b/x$



(b) $y = a + b \ln x$



(c) $y = ae^{bx}$



(d) $y = 1/(a + bx)$

图 4.3 最佳平方逼近变形

5. 结论

- (1) 7 种不同的拟合函数的拟合误差对比如表 4.3 所示。通过对比可以看到 4 次最佳平方逼近的拟合误差最小，但也是计算量最大的一个。
- (2) 4 次最佳平方逼近的拟合误差相比于 3 次最佳平方逼近的拟合误差没有太多提升，反而加大了计算量。
- (3) 在 4 种变形的拟合函数里， $y = a + b \ln x$ 和 $y = 1/(a + bx)$ 的拟合误差也比较小，但是都比 2 次最佳平方逼近的误差大。
- (4) 整体上来说，相比于插值函数，次数不太高的最佳平方逼近的计算量不大。

表 4.3 拟合误差

拟合函数	2 次最佳平方逼近	3 次最佳平方逼近	4 次最佳平方逼近	
拟合误差	3.2166667	1.5102564	1.4679487	
拟合函数	$y = a + b/x$	$y = a + b\ln x$	$y = ae^{bx}$	$y = 1/(a + bx)$
拟合误差	57.302270	8.7788712	41.844596	7.7324292