浙江大學

《数值分析方法Project》

上课时间：	2022学年秋冬学期
授课教师：	余官定
姓　　名：	[redacted]
学　　号：	[redacted]
日　　期：	2022/11/13

1 背景介绍

曲线拟合是一种构建一个函数曲线，使之最佳地吻合现有数据点的过程，该过程可能附加若干条件限制。在实际生活中，我们常常无法直接获得连续的的函数关系，只能的通过取样得到离散的点，通过这些散点得到我们所需的目标函数的过程，就是函数拟合。然而，随着国外软件的封禁，我们急需研发属于我们自己的数学工具。

在秋冬学期，我们学习了函数拟合的基本知识，对拉格朗日内插多项式、三次样条插值、 最小二乘法有了系统的了解，所以在这个project中，我们将借助python和相应的外部库实现基本函数拟合的工具箱

2 设计思路

2.1 设计目的

综合考虑我们日常使用中的使用场景，以及在工程中我们对函数拟合的需求，我们在设计之前，对函数拟合工具箱提出了以下基本的要求：

- 能够使用拉格朗日内插多项式、三次样条插值、 多项式拟合（可以通过输入自定义阶数）来实现函数拟合。
- 考虑实际使用场景，能够用手动输入与文件导入两种手段进行数据输入。
- 为了使函数拟合更直观，需要在提供函数表达式的同时，提供做出函数图像。
- 图形化具有一定的交互性，能给出错误输入的报错等功能.
- ...

2.2 设计流程

考虑到以上这些需求，我们采用功能较为丰富、外部库较多的python作为我们的设计语言。在设计过程中，我们遵循算法设计、图形化界面设计、交互功能设计、代码封装的流程进行。

在总的设计过程中，我们遵循模块化、分而治之的总体思路，这种思路主要体现在功能构建以及检验，不仅使功能设计的思路更为清晰，并且降低了debug的难度。

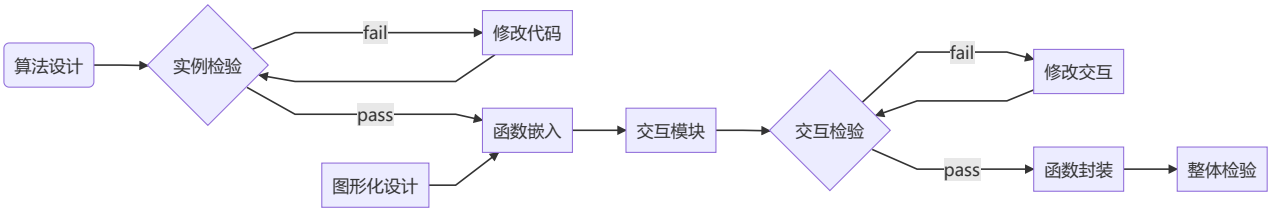

图 思路流程图表示

2.3 设计环境

在实际设计过程中，我们使用的代码环境为*Python*3.10.5，使用的编辑器为*vscode*

同时为了上述目标中的特定功能，如绘图、图形化界面的实现，我们在python中导入了特定的函数库。

```
<---外部库--->
matplotlib                  3.6.2 # 绘图
numpy                       1.23.1 # 矩阵计算
openpyxl                    3.0.10 # 读入文件
pyinstaller                 5.6.2 # 函数封装
sympy                       1.11.1 # 数学公式符号化
```

3 算法描述

3.1 拉格朗日插值法

我们给定拉格朗日插值多项式的一般表达式

$$\begin{aligned} L_{n,k}(x) &= \frac{(x-x_0)\cdots(x-x_{k-1})(x-x_{k+1})\cdots(x-x_n)}{(x_k-x_0)\cdots(x_k-x_{k-1})(x_k-x_{k+1})\cdots(x_k-x_n)} \\ &= \prod_{i=0,i\neq k}^{n} \frac{x-x_i}{x_k-x_i} \end{aligned}$$

并给出伪代码:

```
Input:x_dataset,y_dataset
step1: for xi in x_dataset except xj:
         L_j = L_j * (x - xi) / (xj - xi)
step2: for xi in x_dataset:
         Poly = Sum (y_dataset[j] * L_j)
Output:Poly
```

3.2 三次样条插值

对于任意一个区间j，函数都满足以下表达式

$$S_j(x) = a_j + b_j(x-x_j) + c_j(x-x_j)^2 + d_j(x-x_j)^3$$

3.2.1 自然样条

给出关键代码片段

```
# 循环得到系数矩阵及中间变量和系数的关系
for i in range(listNum):
    hi = x_dataset[i+1] - x_dataset[i]
    if(i != 0):
```

```
        alpha[i] = 3 / hi * (a_coefficient[i+1] - \
a_coefficient[i]) - 3 / last_hi *
(a_coefficient[i] - a_coefficient[i-1])
        l[i] = 2 * (x_dataset[i+1] - x_dataset[i-1]) -
last_hi *u[i-1]
        u[i] = hi/l[i]
        z[i] = (alpha[i] - last_hi * z[i-1]) / l[i]
                last_hi = hi
# 利用中间变量得到各个系数列表
for j in range(listNum-1, -1, -1):
    hj = x_dataset[j+1] - x_dataset[j]
    c_coefficient[j] = z[j] - u[j] * c_coefficient[j+1]
    b_coefficient[j] = (a_coefficient[j+1] - a_coefficient[j])
/ hj\
    -hj * (c_coefficient[j+1]+2 * c_coefficient[j]) / 3
    d_coefficient[j] = (c_coefficient[j+1] - c_coefficient[j])
/ (3 * hj)
```

3.2.2 紧压样条

与自然样条相比，紧压样条给定了左右两侧的导数值。

下给出关键代码片段

```
# 循环得到系数矩阵与中间变量的关系
for i in range(listNum):
    hi = x_dataset[i+1] - x_dataset[i]
    if(i != 0):
        alpha[i] = 3 / hi * (a_coefficient[i+1] - \
a_coefficient[i])- 3 / last_hi *\
(a_coefficient[i] - a_coefficient[i-1])
        l[i] = 2 * (x_dataset[i+1] - x_dataset[i-1]) - last_hi
 \*u[i-1]
        u[i] = hi/l[i]
        z[i] = (alpha[i] - last_hi * z[i-1]) / l[i]
        last_hi = hi
        # 单独考虑循环到最后一次的规律
        if(i == listNum-1):
            alpha[listNum] =  3 * FPN - 3 *
(a_coefficient[listNum] -\
a_coefficient[listNum-1]) /hi
            l[listNum] = hi * (2 - u[listNum-1])
            z[listNum] = (alpha[listNum] - hi * z[listNum-
1])/l[listNum]
            c_coefficient[listNum] = z[listNum]
# 根据系数关系得到系数列表
for j in range(listNum-1, -1, -1):
    hj = x_dataset[j+1] - x_dataset[j]
```

```
    c_coefficient[j] = z[j] - u[j] * c_coefficient[j+1]
                    b_coefficient[j] = (a_coefficient[j+1] -\
a_coefficient[j]) / hj-hj *\
(c_coefficient[j+1] + 2 * c_coefficient[j])\
/ 3
    d_coefficient[j] = (c_coefficient[j+1] - c_coefficient[j])\
/ (3 * hj)
```

3.3 最小二乘法

相比于前两种方法，在给定的散点上没有误差，最小二乘法考虑整个函数的准确性，降低错误采样的带来的的不利影响。

下给出最小二乘法的公式：

$$P_n(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0$$

$$E = \sum_{i=1}^{m} \left(y_i - P_n\left(x_i\right)\right)^2$$

满足的条件为：

$$\frac{\partial E}{\partial a_j} = 0$$

给出关键代码：

```
def pow_sum(list:list, order, ylist = []):# ylist is for
another pow and sum
    ret = 0.0 # 初值为0
    listNum = len(list)
    # 对ylist做判断，默认函数参量为空，此时都赋值为1，不影响list的乘幂
    if(ylist == []):
        ylist = [1] * listNum
    # 循环实现乘幂与累和
    for i in range(listNum):
        ret = ret + ylist[i] * list[i] ** order
        return ret
# 计算系数矩阵
for i in range(order + 1):
    # 计算常数向量
    const_arr[i] = pow_sum(x_dataset, i, y_dataset)
    for j in range(order + 1):
        coeff_mat[i][j] = pow_sum(x_dataset, (i+j))
# 引入线性代数库求解线性方程组
if(linalg.det(coeff_mat) == 0): # 如果系数矩阵不满秩，即没有唯一解
   return "there is no unique solution!"
a_coefficient_solution = linalg.solve(coeff_mat, const_arr)
```

4 应用示例

在文件夹中我们给出了data1.xlsx，data2.xlsx，data3.xlsx三个数据集，分别模拟小、中、大三种数据集

data1：

x	y
1	2.5
2	3.6
3	7.2

data2：

x	y
1	2.5
2	3.4
3	9.5
4	5.1
5	3.89
6	6.97
7	3.48
8	11.53
9	9.12
10	3.5

data3：

x	y	x	y	x	y
2	1026	26	10117	43	63386
3	654	27	19742	44	65301
6	1522	29	21257	45	50045
8	1995	31	20030	47	67226
12	6187	33	21977	48	46293
13	5596	35	25420	51	72076
14	5270	36	33686	52	79533
17	9729	39	35037	55	80120
18	6017	40	21062	56	46029
21	17501	41	62802	60	130683
24	16205	42	52877	61	94542
25	18657	43	63386	62	104950

4.1 拉格朗日插值法

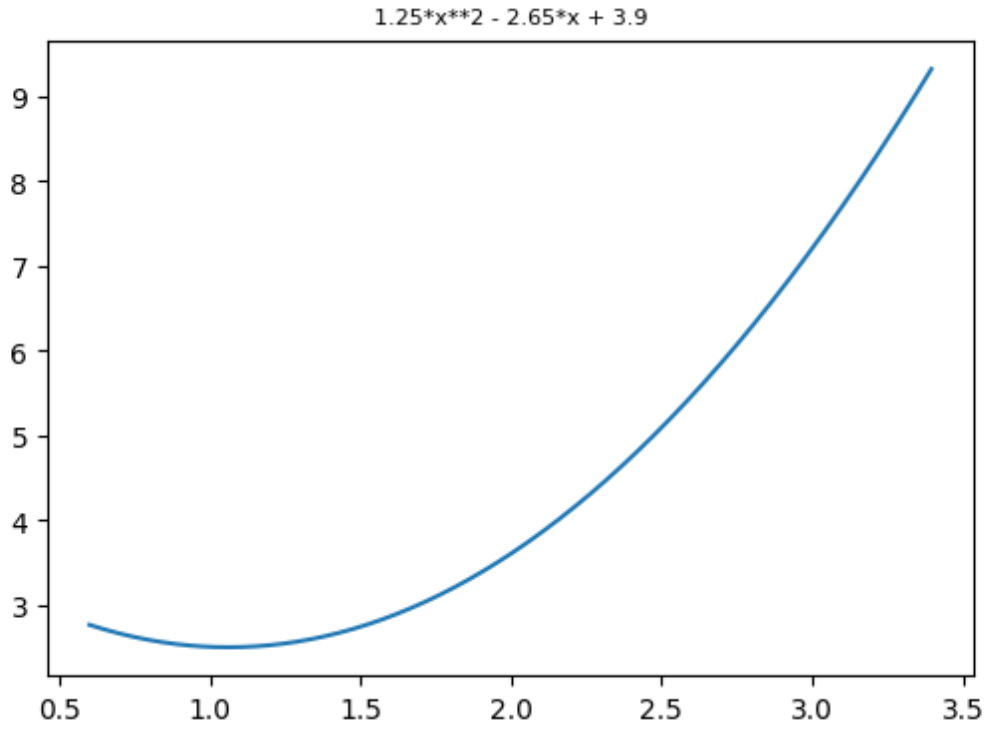

图 2 data1拉格朗日插值法

运行时间较快。

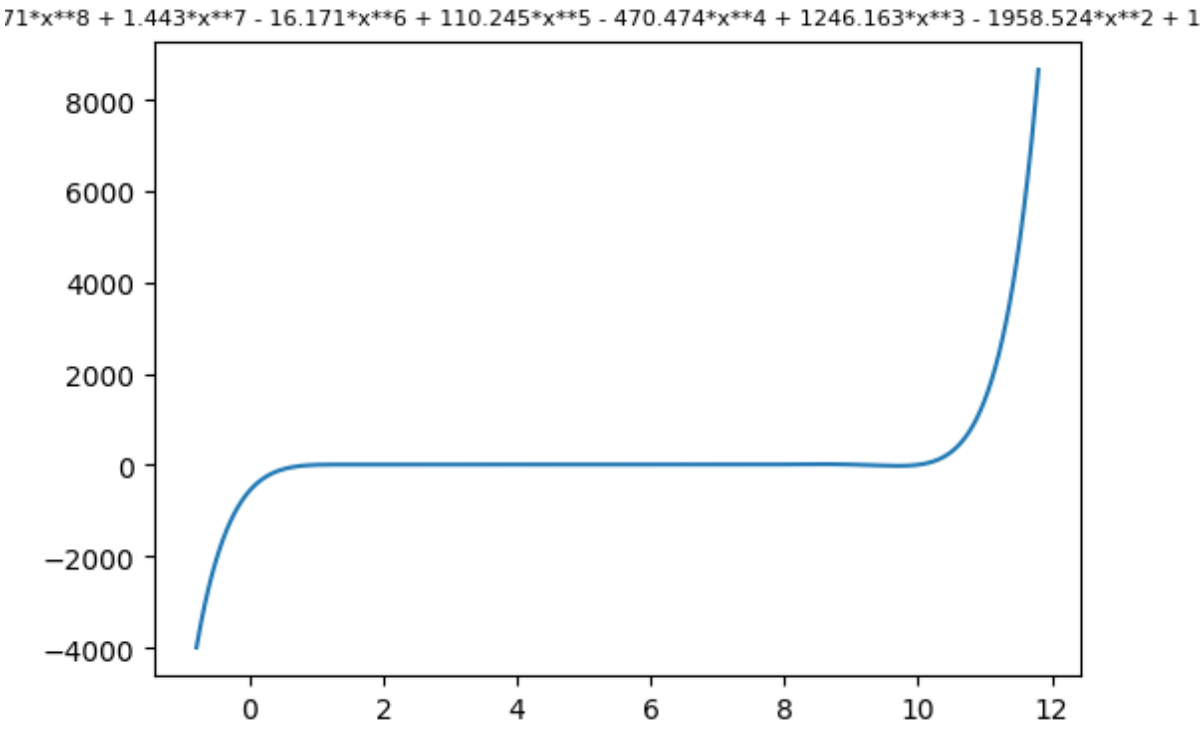

图 data2拉格朗日插值法

运行时间在4~5秒之间。在小于0，大于10，即给定散点范围之外拟合的不好

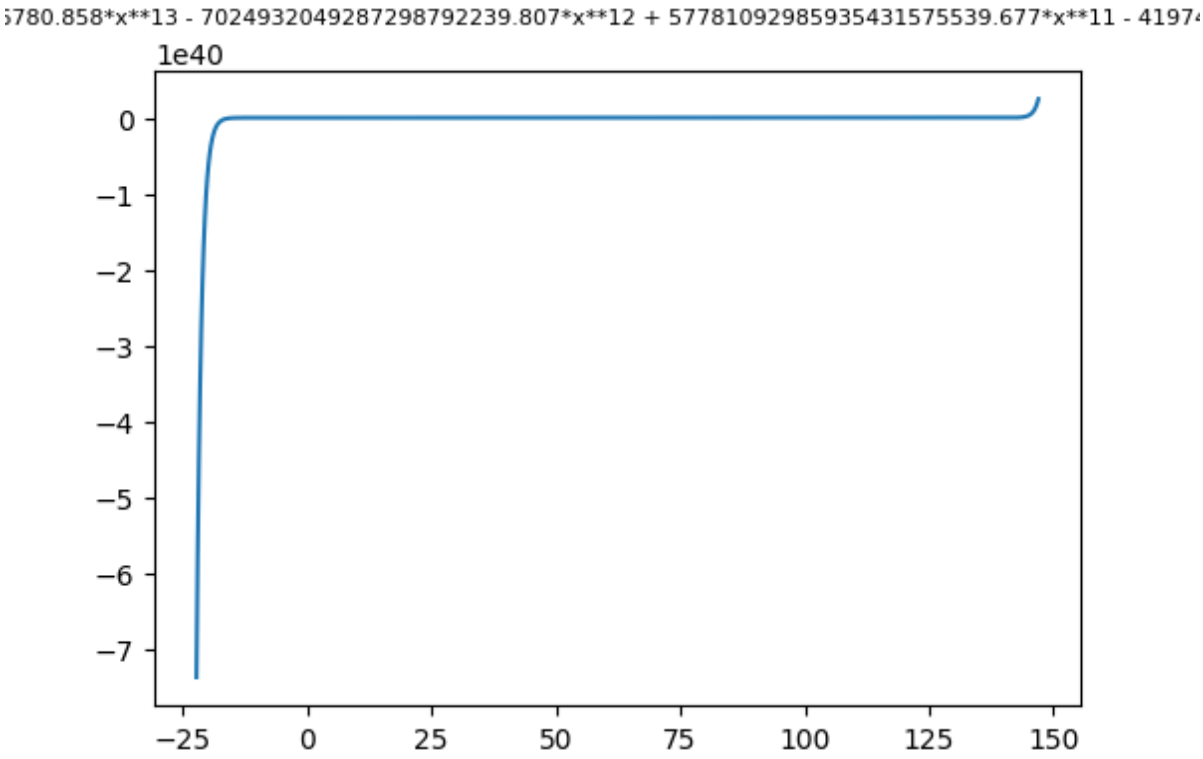

图 2 data3拉格朗日插值法

运行时间在1分10秒左右，在0

4.2 最小二乘法（以五阶为例）

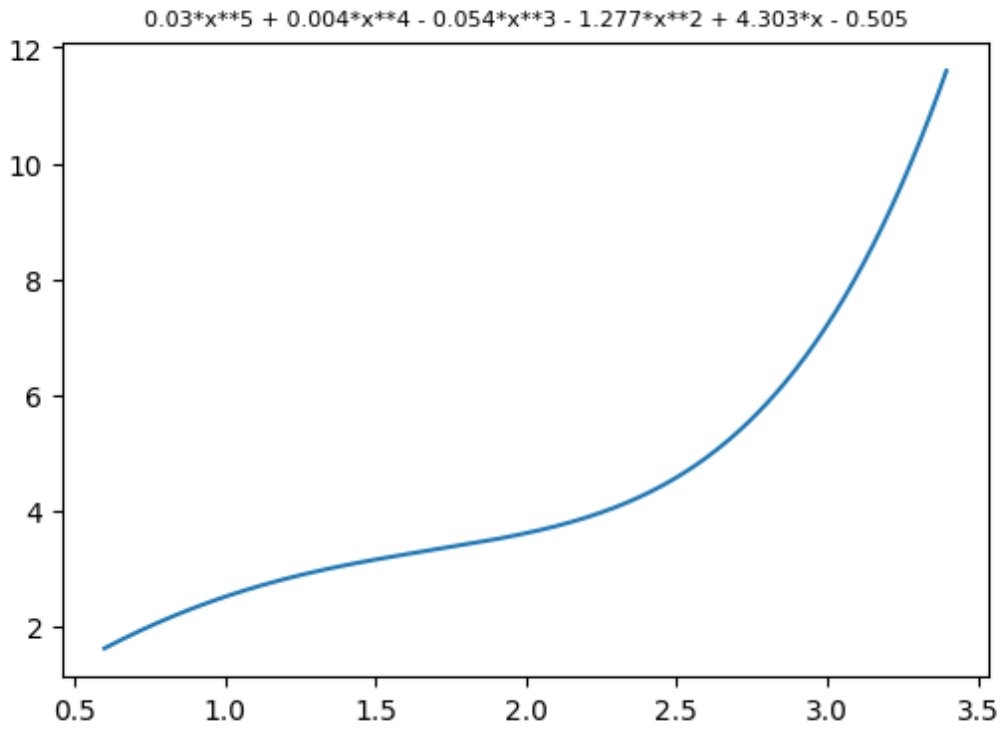

图 data1五阶多项式拟合

运行时间较快。

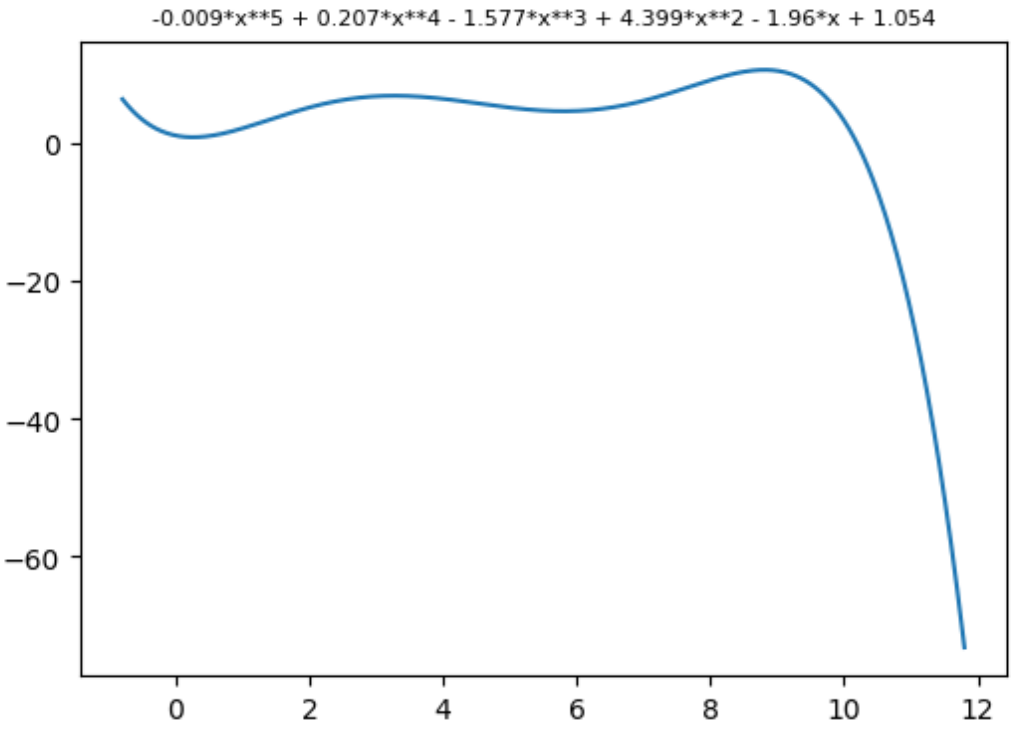

图 data2五阶多项式拟合

运行时间小于2s。

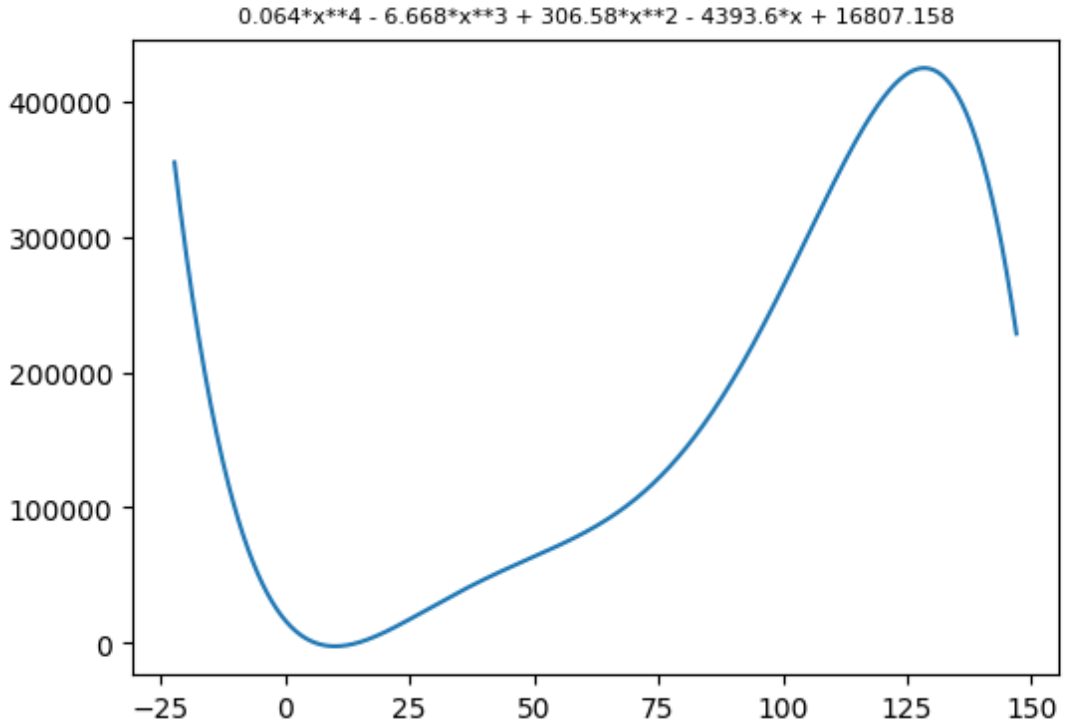

图 data3五阶多项式拟合

时间小于2s。同样出现了，在给定散点范围之外拟合的不好

不妨再对高阶进行测试。

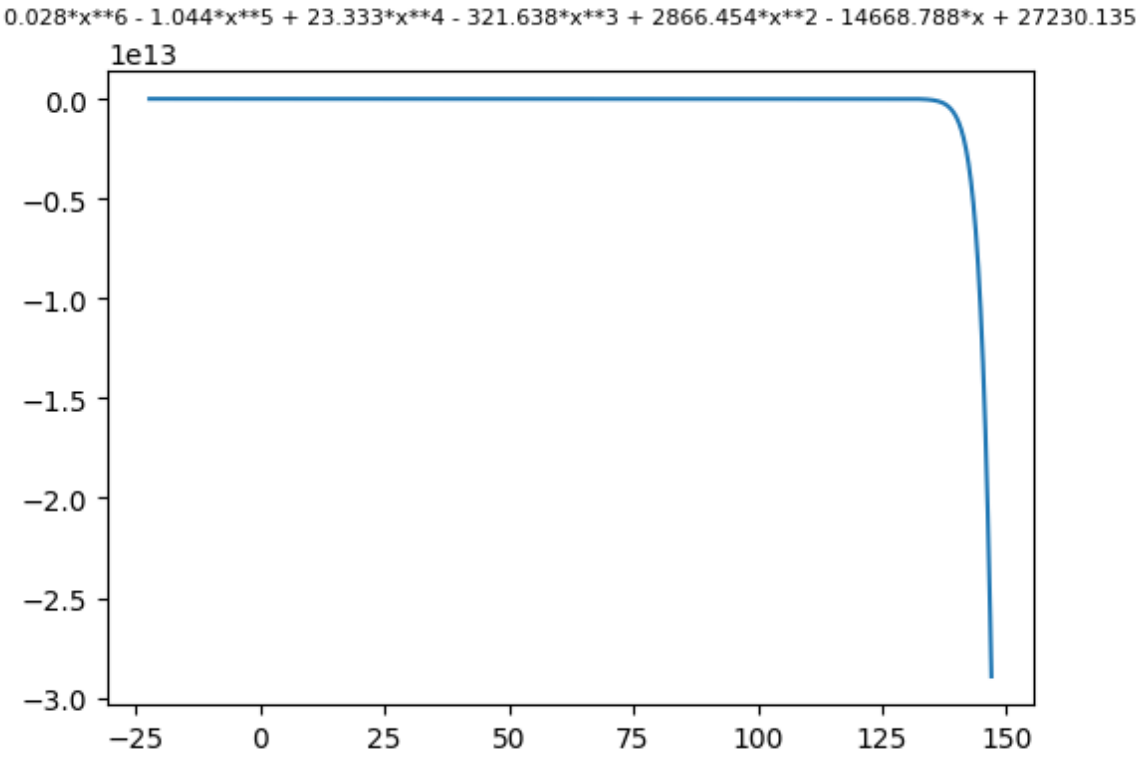

图 data3五十阶多项式拟合

我们发现在图像已经明显不符合准确性。

4.3 三次样条（自然，第0段）

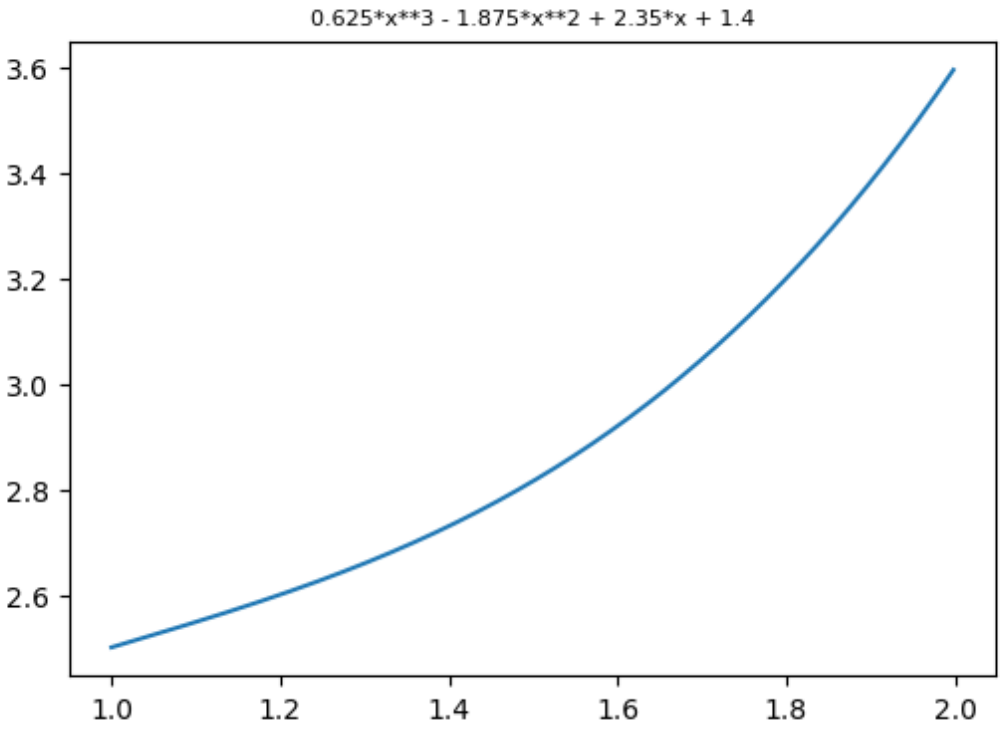

图 data1三次自然样条区间0拟合

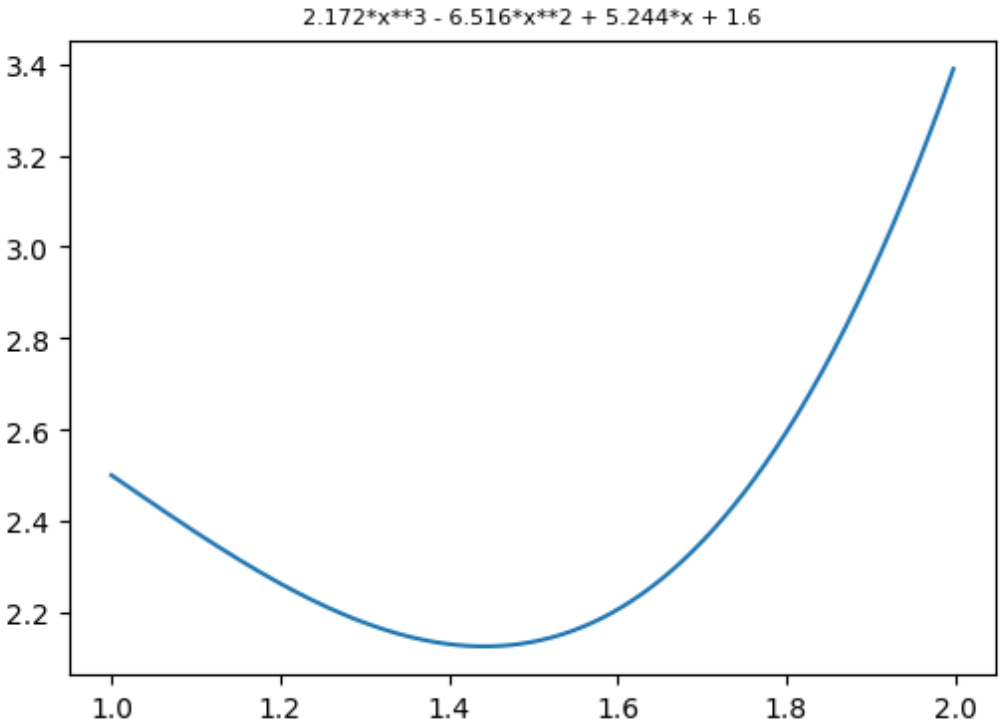

图 data2三次自然样条区间0拟合

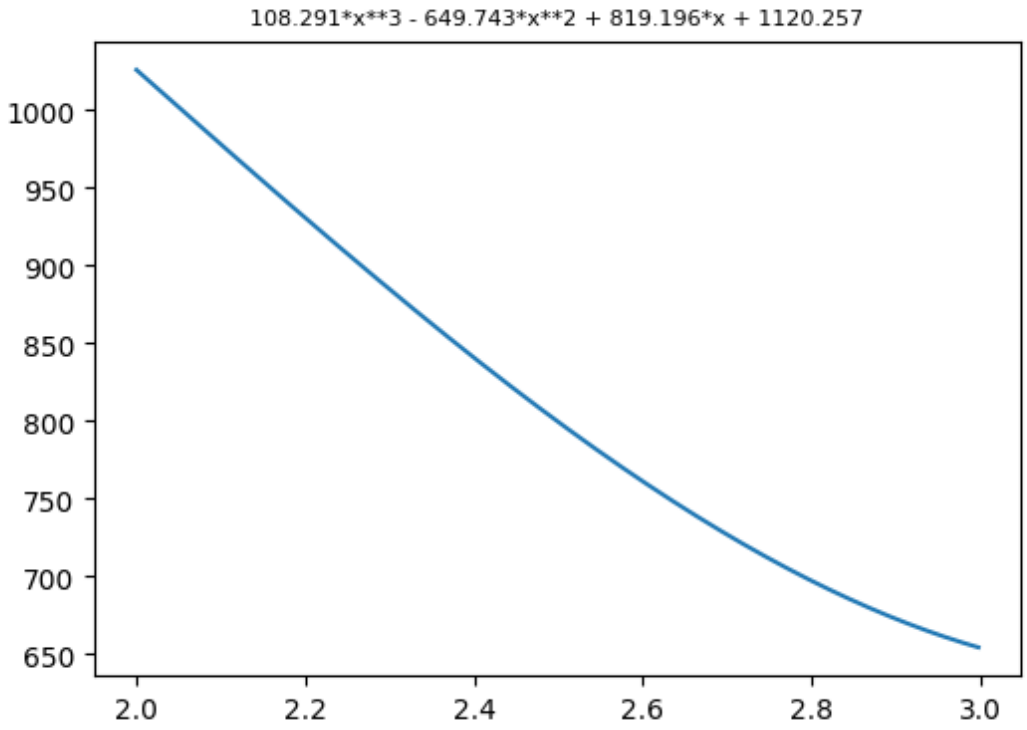

图 data3三次自然样条区间0拟合

ps.在应用实例中计算的时间对研究时间复杂度并没有太大意义，因为在实际操作过程，我们发现运行时间的主要贡献是来自画图。

5 时间复杂度分析

拉格朗日内内插多项式：L_j为$O(N)$，总的为O（N^2）.

三次样条实则为线性方程组求解，时间复杂度为$O(N)$

多项式拟合，设计系数矩阵生成，时间复杂度为$O(N^3)$

6 软件使用说明

在提交的压缩中，给出了.exe文件与命名为source_code的文件夹，可以选择直接运行.exe文件或文件夹中运行Curve_Fitting_Toolbox.py文件，具有同样效果。建议使用的是Curve_Fitting_Toolbox.py文件,Curve_Fitting_Toolbox.py文件运行速度要比.exe要快，图像往往显示更慢，而具体函数能够较快在终端中得到。并且在终端中，可以看到最小二乘法的error。

软件运行界面如下：

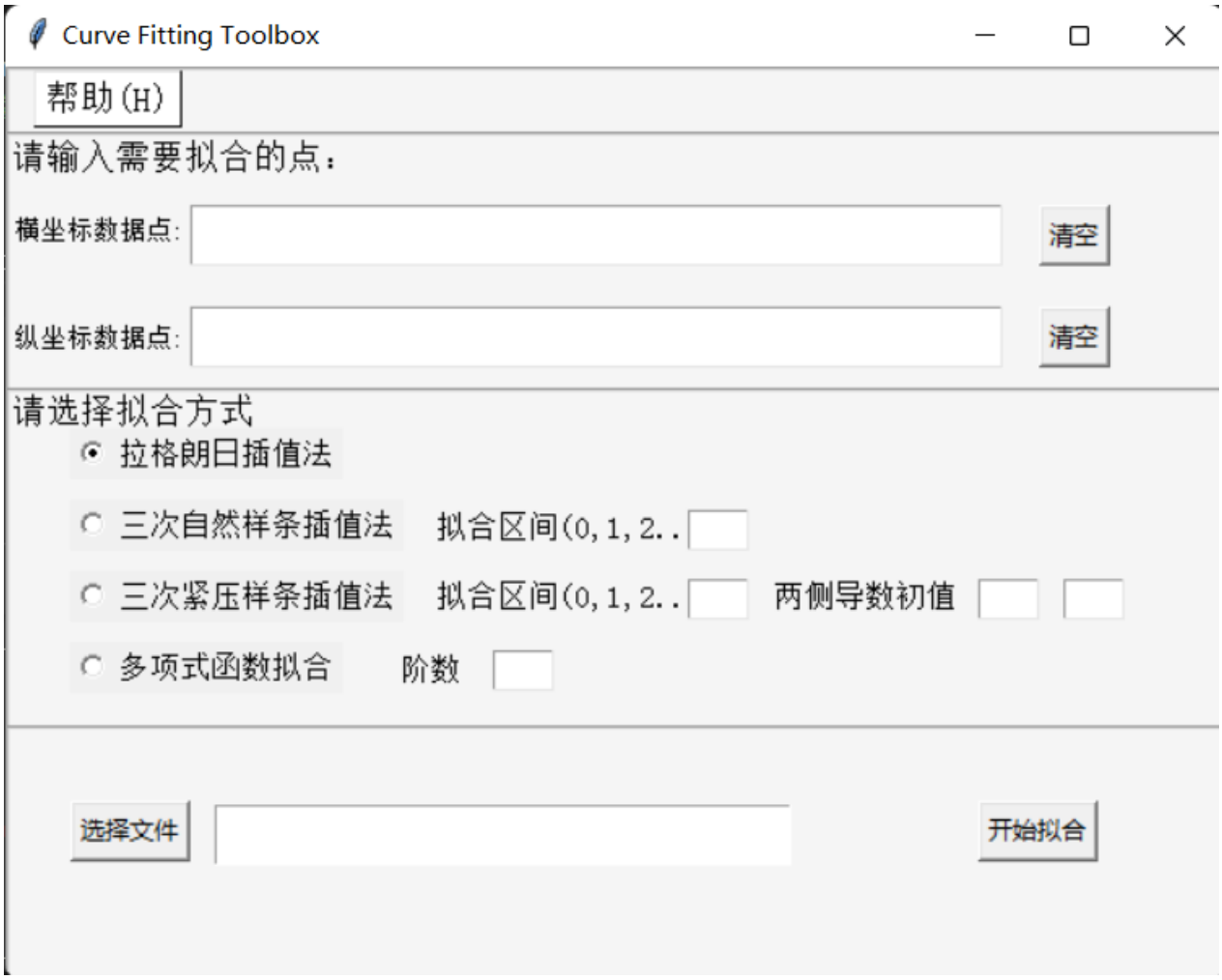

图 2 软件运行界面图

您有手动输入与文件导入两种选择，report中给定的实例已经打包放入对应的文件夹中了。注意，如果您要编写自己的excel文件导入，请务必注意导入的路径中不要存在中文，否则极大可能会寄。同时，您可以点击“帮助”获取关于输入等操作的细节指导。

7 优化与改进

在测试程序的过程中，我们发现了以下一些问题：

- 性能方面：由于算法主要都是靠循环和四则运算，故时间复杂度会比较高，优化的方向是利用numpy进行向量化计算。
- 功能方面
    1. 每一次拟合都要重新打开软件，改进思路：将函数图像放在新的窗口上，每次拟合完销毁该窗口。
    2. 三次样条插值无法整体做出图像，改进思路：将多个图同时显示在一张figure上。