数值分析第 7 次上机作业

学号：221840189，姓名：王晨光

§1 问题一

1. 把 `polyfitEx2.py` (Python 代码) 或 `polyfitEx2.m` (Matlab 代码) 的 `TODO` 部分改成可以用下面的广义多项式拟合:

$$f(x) = a_0 + a_1 f_1(x) + a_2 f_2(x) + \cdots + a_n f_n(x).$$

注：这里采用的验证算例是用 $1, x, \cos x, \sin x$ 这四个函数的线性组合来拟合前面的数值算例.
提示：对于 Matlab 代码, 你可能需要查阅 “元胞数组” 的使用方法.

1.1 算法思路

按照广义多项式拟合的方法补全代码：

```
1        # TODO
2        xdatas[i] = f[i-1](xdata)
```

之后使用 polyval 函数获取拟合的广义多项式函数的值，绘制出函数图像. 考虑使用函数族 $f = (1, x, \cos x, \sin x)$ 进行拟合.

Algorithm 1 由系数向量得到拟合函数值

Require: 系数向量 $a = (a_n, a_{n-1}, \cdots, a_1, a_0)$，自变量向量 x，
拟合（广义）多项式族 $f = (f_0, f_1, \cdots, f_{n-1}, f_n)$.
Ensure: 函数值向量 y
function POLYVAL(a, x, f)
翻转系数向量 a
$y_k \leftarrow a_0, \forall i$
for i 从 1 到 n **do**
$y \leftarrow a_i \cdot f_i(x)$
end for
return y
end function

1.2 结果分析

与二次多项式拟合进行对比.

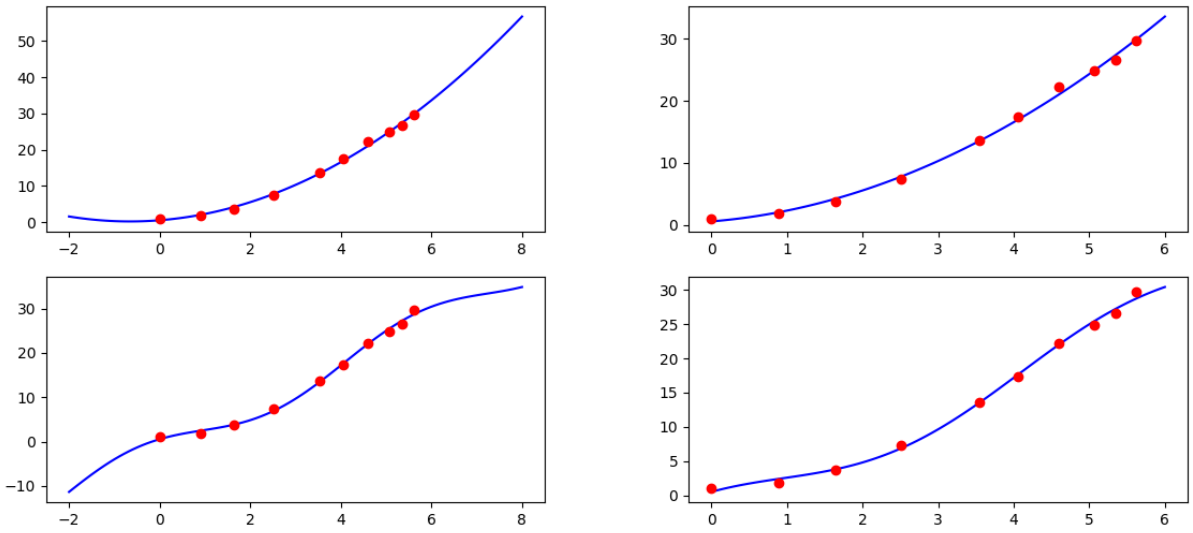


图 1: 二次多项式插值（上）与广义多项式插值（下）

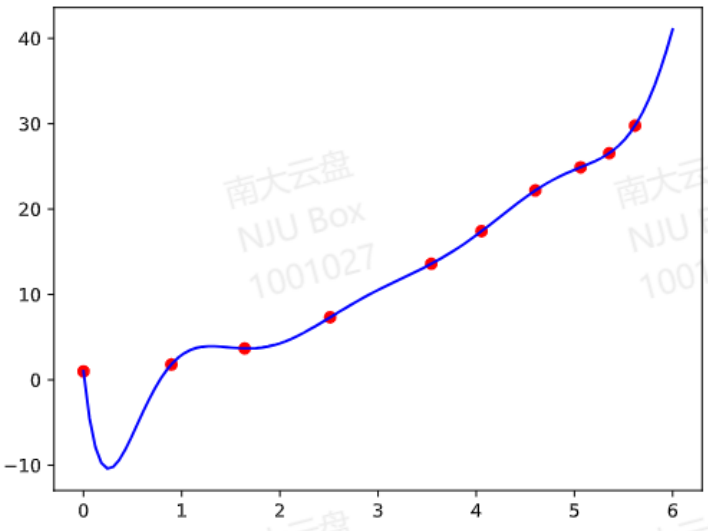
可见多项式拟合与广义多项式在该批数据点上都有较好的拟合，将函数的定义域延展，可以观察到不同的效果，可以根据具体需求和更多的实际细节来选择拟合方式.

§2 问题二

2. 数据

x	0.000	0.895	1.641	2.512	3.542	4.054	4.602	5.063	5.354	5.617
y	1.000	1.803	3.680	7.320	13.59	17.41	22.19	24.89	26.55	29.77

的 9 次多项式拟合可以得到拟合曲线



(1) 得到的均方误差和矩阵 $\boldsymbol{A}$ 的条件数是多少?

提示: 在 Python 中可以用 `numpy.linalg.cond` 来输出条件数; 在 Matlab 中可以用 `cond` 来输出条件数.

(2) 可以证明如果用 9 次多项式拟合, 那么当 f 是 Lagrange 插值多项式时, 均方误差是 0. 但 (1) 中均方误差的结果并不是 0, 可以依此认为在这个例子中 9 次多项式插值和 9 次多项式拟合不是一回事吗?

(3) 前面多项式空间 P^n 采用的基是 $\{1, x, \cdots, x^n\}$. 把多项式空间 P^n 的基改成 Chebyshev 多项式

$$\{T_k(x) : k = 0, 1, \cdots, n\},$$

然后采用 "广义多项式拟合" 一节的方法来推导矩阵 $\boldsymbol{A}$ 并求出拟合函数, 观察均方误差和 $\boldsymbol{A}$ 的条件数会怎么表现.

2.1 算法思路

矩阵 A 的条件数是描述矩阵在数值计算中的稳定性和精度的指标. 它告诉我们矩阵在数值计算中对误差的敏感程度，具体来说，条件数越大，矩阵在计算过程中对误差越敏感.

均方误差的计算公式为

$$\mathrm{MSE} = \frac{1}{m}\sum_{i=1}^{m}(y_i - \hat{y}_i)^2$$

根据 9 次多项式插值与 9 次多项式拟合的计算方法，可以得到它们的理论结果是完全一致的. 但是由于程序运行 9 次多项式拟合的过程中会出现数值计算的舍入误差及其累积，这会导致最终结果与理论结果有出入，所以命题 (2) 并不成立.

Chebyshev 多项式可以通过递推关系得到：

$$\begin{cases} T_0 = 1 \\ T_1 = x \\ T_n = 2xT_{n-1} - T_{n-2}, n \geqslant 2 \end{cases}$$

通过问题一中使用的广义多项式插值方法，同样可以得到使用 Chebyshev 多项式作为基底进行拟合的结果：

$$f(x) = a_0T_0(x) + a_1T_1(x) + \cdots + a_nT_n(x)$$

2.2 结果分析

使用 $\{1, x, x^2, \cdots, x^9\}$ 作为基底进行 9 次多项式拟合时，可以得到矩阵 A 的条件数与拟合函数的离散均方误差如下：

$$\begin{cases} A_{\mathrm{cond}} = 8.578682559932301e + 20 \\ \mathrm{MSE} = 0.0004939324785567218 \end{cases}$$

这说明此时的矩阵 A 具有一定的病态性，但由于选点较少且并不极端，得到的拟合函数结果几乎没有离散误差，有着良好的拟合效果.

使用 Chebyshev 多项式 $\{1, x, T_2(x), \cdots, T_9(x)\}$ 作为基底进行 9 次多项式拟合时，可以得到矩阵 A 的条件数与拟合函数的离散均方误差如下：

$$\begin{cases} A_{\text{cond}} = 5.5388093317210355e + 25 \\ \text{MSE} = 0.00017881211322257203 \end{cases}$$

A 的条件数进一步增大但均方误差进一步减小，这应当与 Chebyshev 多项式的性质有关.

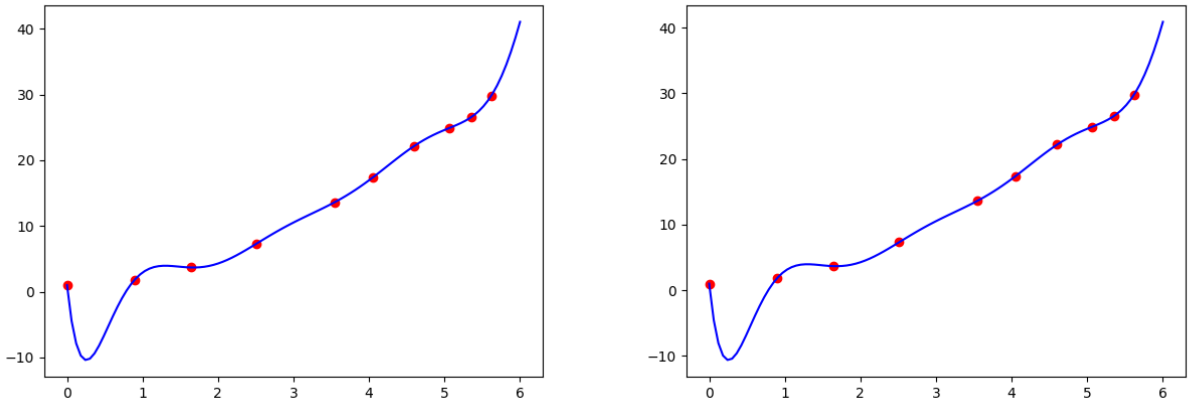


图 2: 9 次多项式插值（左）与 9 次 Chebyshev 多项式插值（右）

§3 问题三

3. 手动实现梯度下降法.

(1) 请补全 `nonlinfitEx2.py` (Python 代码) 或 `nonlinfitEx2.m` (Matlab 代码) 的 `TODO` 部分, 完成梯度下降法优化函数

```
optimize(f,df,x0,eta,eps = 1e-8),
```

输入值分别为函数 `f`、它的梯度 `df`、初值 `x0`、步长 `eta`、终止准则的误差容限是 `eps`. 在程序中, 我们首先判断输入参数的维数是否对应, 如果维数不对应就返回一个错误: Dimension does not match.

(2) 数据

x	-1.5	-1	-0.5	0	0.5	1	1.5
y	0.04	0.17	0.65	1.39	1.85	1.90	1.99

用形如 $y = \dfrac{2}{1 + ae^{bx}}$ 的函数来逼近上面的数据. 先用线性化技巧得到了一个初值, 再用你写的梯度下降法来得到一个更好的值, 并画出拟合函数的图像和损失函数随着迭代步数的变化图像.

3.1 算法分析

首先作一定的变换实现线性化：

$$y = \frac{2}{1+ae^{bx}} \Leftrightarrow \frac{2}{y} = 1 + ae^{bx} \Leftrightarrow \ln\left(\frac{2}{y} - 1\right) = bx + \ln a$$

因此，取 $u(x) = x$, $v(y) = \ln(\frac{2}{y} - 1)$ 即可实现线性化.

课堂上介绍的梯度下降法的具体方法如下：

Algorithm 2 梯度下降法

Require: 多变量单值函数 f, 梯度向量 $\nabla f = (\frac{\partial f}{\partial x_1}, \cdots, \frac{\partial f}{\partial x_n})$, 初始点 $x_0 = (x_0^1, \cdots, x_0^n)$, 学习率 η, 最大收敛误差 eps.

Ensure: 最优值点 $x_* = (x_*^1, \cdots, x_*^n)$

function OPTIMIZE(f, ∇f, x_0, η, eps)

$x_* = x_0$

for $\|\nabla f(x_*)\|_2^2 > eps$ **do**

$x_* \leftarrow x_* - \eta \nabla f(x_*)$

end for

return x_*

end function

根据上述过程补全代码：

```
        # TODO
        while True:
            gradient = np.array([dfi(*x) for dfi in df])
            x -= eta * gradient
            if np.linalg.norm(gradient) < eps:
                break
```

本题中，梯度下降法选取的单值函数 f 为 $loss$ 函数：

$$\mathcal{L}(a,b) = \frac{1}{m}\sum_{i=1}^{m}\left|\frac{2}{1+ae^{bx_i}} - y_i\right|^2$$

可以求得梯度向量 $\nabla f = (\frac{\partial \mathcal{L}}{\partial a}, \frac{\partial \mathcal{L}}{\partial b})$ 如下：

$$\begin{cases} \dfrac{\partial \mathcal{L}}{\partial a} = \dfrac{1}{m}\displaystyle\sum_{i=1}^{m} \frac{-8e^{bx_i}}{(1+ae^{bx_i})^2} \cdot \left(\frac{2}{1+ae^{bx_i}} - y_i\right) \\ \dfrac{\partial \mathcal{L}}{\partial b} = \dfrac{1}{m}\displaystyle\sum_{i=1}^{m} \frac{-8ax_i \cdot e^{bx_i}}{(1+ae^{bx_i})^2} \cdot \left(\frac{2}{1+ae^{bx_i}} - y_i\right) \end{cases}$$

固定 $\eta = 10^{-4}$, $eps = 10^{-8}$ 进行优化.

3.2 结果分析

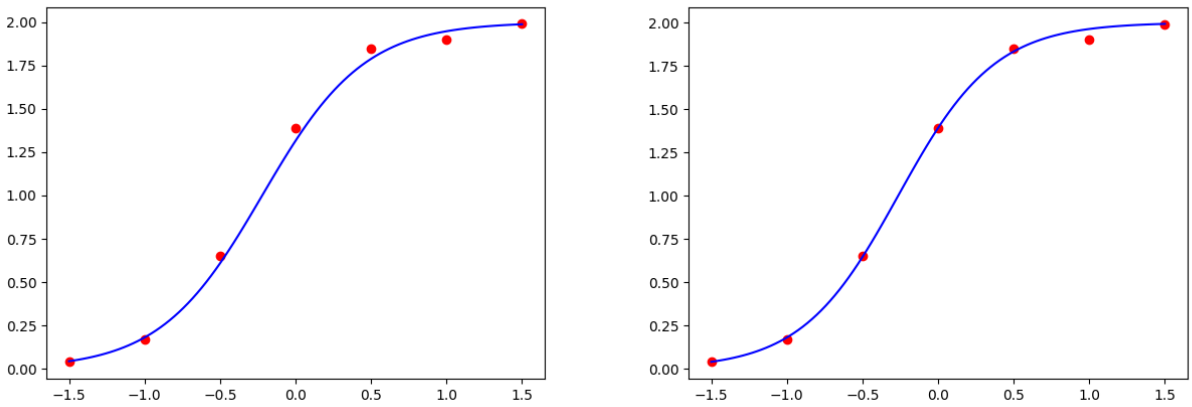


图 3: 梯度下降前（左）后（右）拟合函数图像

使用梯度下降法前的系数为 $\begin{cases} a = 0.5202092596711911 \\ b = -2.9599994340662583 \end{cases}$. 使用梯度下降法后的系数为 $\begin{cases} a = 0.43798576249523435 \\ b = -3.1320556568173634 \end{cases}$ 再观察图像，可以看出梯度下降法有利于得到更好的局部最优解，得到效果更好的拟合函数. 梯度下降法的收敛函数（下图）也印证了这一点：

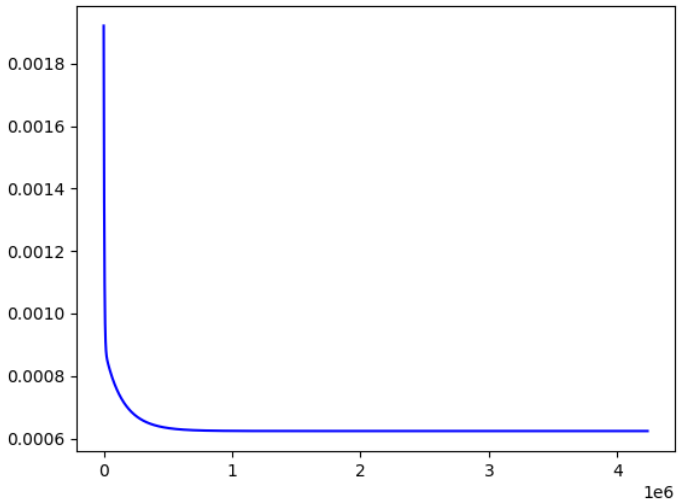


图 4: 梯度下降法 *loss* 值随迭代次数的变化

§4 问题四

4. COVID-19 冠状病毒疫情爆发初期 (2020 年 1 月 16 日至 2020 年 3 月 16 日) 武汉市的确诊人数在附件的 `wuhan2020.csv` 中存储. Logistic 模型

$$\frac{\mathrm{d}P}{\mathrm{d}t} = r\left(1 - \frac{P}{K}\right)P$$

可以用来刻画时间 t 的感染人口数 P. 其中, K 是最大容量, r 是增长率. 若初值为 $P(0) = P_0$, 则这个微分方程的解是

$$P(t) = \frac{K}{1 + (\frac{K}{P_0} - 1)e^{-rt}}. \qquad (*)$$

把 (*) 式作为拟合函数, 根据给定的确诊数据来给出 P_0,K 和 r 这三个参数的预测值.

拐点是满足 $P''(t_0) = 0$ 的点 t_0.

(1) 根据 1 月 16 日至 3 月 16 日的确诊数据, 给出 P_0, K, r 的预测值, 并确定拐点的日期.

(2) 假如今天是 2020 年 2 月 1 日, 仅知道 1 月 16 日至 1 月 31 日这段时间的数据, 用这部分数据预测 P_0, K, r 的值是多少? 根据这些预测值得到的模型, 估计 2 月 1 日至 3 月 16 日的确诊人数和拐点日期, 并与真实值作比较. (拐点可与 (1) 中求出的拐点作比较)

注: 调用 `csv` 文件的方法已在 `nonlinfitEx3.py` 和 `nonlinfitEx3.m` 给出.

4.1 算法分析

首先作一定的变换实现线性化:

$$P(t) = \frac{K}{1 + \left(\frac{K}{P_0} - 1\right)e^{-rt}} \Leftrightarrow \ln[\frac{K}{P(t)} - 1] = \ln(\frac{K}{P_0} - 1) - rt$$

因此, 取 $u(t) = -t$, $v(P) = \ln(\dfrac{K}{P} - 1)$ 即可实现线性化 $v = bu + c$, 并通过逆变换得到:

$$\begin{cases} \dfrac{K}{P_0} = e^c + 1 \\ r = -b \\ P_0 = 45 \quad \text{（观察数据可得）} \end{cases}$$

便可以解出 P_0, K, r 的值. 为求出拐点 t_0 值，考虑对 $P(t)$ 作二阶差分代替 $P^{''}(t)$，即：

$$P^{''}(t) \approx \frac{\Delta^2 P(t)}{(\Delta t)^2} = \frac{\frac{\Delta P(t+1)}{\Delta t} - \frac{\Delta P(t)}{\Delta t}}{\Delta t} = P(t+2) + P(t) - 2 \cdot P(t+1)$$

则可以求出 $t = t_0$ 为 $P^{''}(t)$ 的零点.

使用同样的方法，也可以解决第二问的问题.

4.2 结果分析

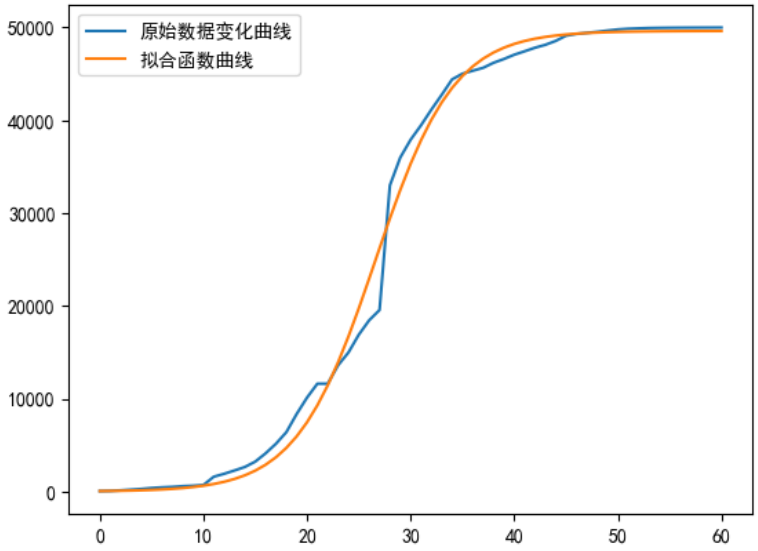


图 5: Logistic 模型拟合函数图像

得到的参数值如下：

$$\begin{cases} K = 4.96346018e+04 \\ r = 2.63615937e-01 \\ P_0 = 45 \end{cases}$$

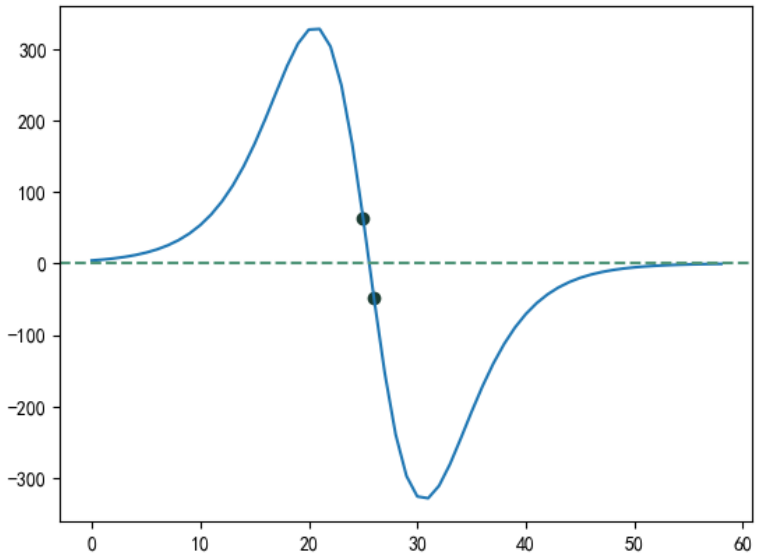


图 6: 拟合函数二阶差分图像

由图像可以得出，拐点的日期在 2020/2/10 与 2020/2/11 之间，即 1 月 16 日之后的 25 天到 26 天之间.

若只考虑 1 月 16 日至 1 月 31 日的数据，则有如下的拟合结果：

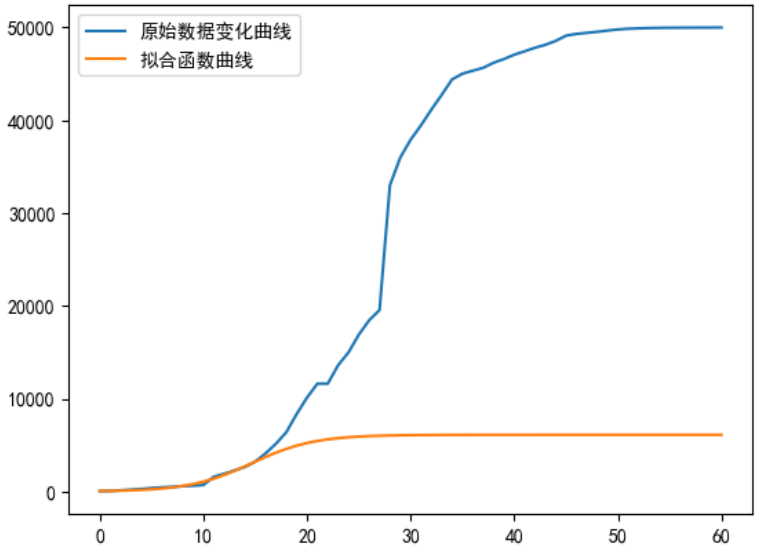


图 7: Logistic 模型拟合函数图像

得到的参数值如下：

$$\begin{cases} K = 6.10376831e+03 \\ r = 3.34055875e-01 \\ P_0 = 45 \end{cases}$$

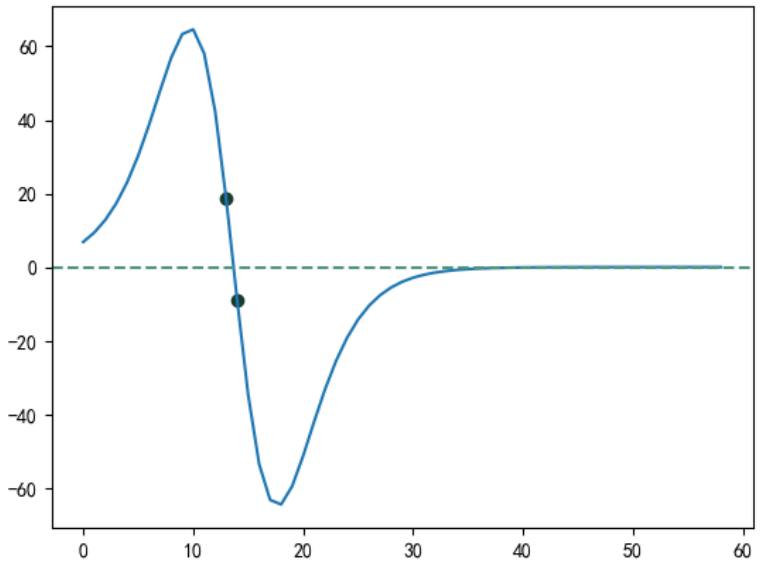


图 8: 拟合函数二阶差分图像

由图像可以得出，拐点的日期在 2020/1/29 与 2020/1/30 之间，即 1 月 16 日之后的 13 天到 14 天之间. 但可以看出，只使用前 16 天的数据，模型拟合的效果很不理想，说明实际的疫情演变需要考虑更多复杂的因素.

§5 结论

函数拟合分为线性拟合与非线性拟合，线性拟合的方法可以推广到多项式拟合与广义多项式拟合，非线性拟合可以通过一定的线性化手段实现将非线性拟合问题转化为一个线性拟合问题. 找到一个合适的参数向量后，可以通过梯度下降法进一步优化参数向量，得到更好的拟合效果.

§6 附录: 问题四完整程序代码

```
import pandas as pd  #需要安装pandas包读取csv文件
# 读取数据
df = pd.read_csv('wuhan2020.csv')
df
df_save = df[df.index<61]
df_save
import numpy as np
t=np.array(df_save.index)
P=np.array(df_save['武汉市'])
print(t)
print(P)
from scipy.optimize import curve_fit
# 定义拟合函数
def func(t, K, r):
    return K / (1 + (K/45 - 1) * np.exp(-1 * r * t))
# 使用curve_fit进行拟合
popt, pcov = curve_fit(func, t, P, maxfev=100000)
# 输出拟合得到的参数值
print("拟合参数:", popt)
import matplotlib.pyplot as plt
plt.rcParams['font.sans-serif'] = ['SimHei']  # 指定默认字体
plt.rcParams['axes.unicode_minus'] = False  # 解决保存图像是负号'-'显示为方块的问题
plt.plot(t,P,label='原始数据变化曲线')
plt.plot(t,func(t,popt[0],popt[1]),label='拟合函数曲线')
plt.legend()
plt.show()
diff2=[]
for i in range(59):
    diff2.append(func(i+2,popt[0],popt[1])+func(i,popt[0],popt[1])-2*func(i+1,popt
print(diff2)
plt.plot(t[:59],diff2)
plt.axhline(y=0, linestyle='—', color='#509579')
plt.scatter(25, diff2[25], color='#1a3b30')
plt.scatter(26, diff2[26], color='#1a3b30')
plt.show()
for i in range(59):
    if diff2[i]*diff2[i+1]<0:
        print('t0␣=',i)
        break
df_save[df_save.index==25]
df_save[df_save.index==26]
df_save[df_save['日期']=='2020/2/1']
df_predict=df_save[df_save.index<16]
```

```
df_predict
t_pre=np.array(df_predict.index)
P_pre=np.array(df_predict['武汉市'])
popt, pcov = curve_fit(func, t_pre, P_pre, maxfev=100000)
print(popt)
plt.plot(t,P,label='原始数据变化曲线')
plt.plot(t,func(t,popt[0],popt[1]),label='拟合函数曲线')
plt.legend()
plt.show()
diff_2=[]
for i in range(59):
    diff_2.append(func(i+2,popt[0],popt[1])+func(i,popt[0],popt[1])-2*func(i+1,pop
plt.plot(t[:59],diff_2)
plt.axhline(y=0, linestyle='—', color='#509579')
plt.scatter(13, diff_2[13], color='#1a3b30')
plt.scatter(14, diff_2[14], color='#1a3b30')
plt.show()
for i in range(59):
    if diff_2[i]*diff_2[i+1]<0:
        print('t0␣=',i)
        break
df_save[df_save.index==13]
df_save[df_save.index==14]
```