插值和拟合在数学建模中的应用

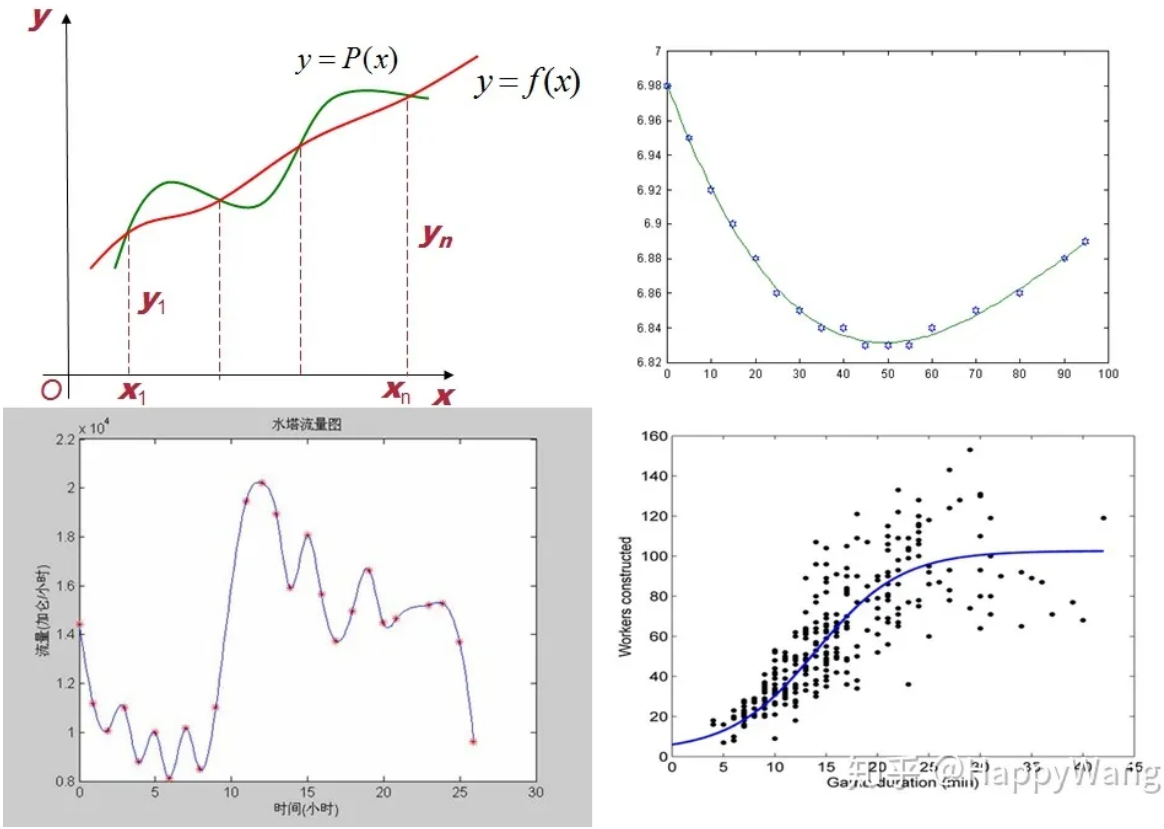
插值问题：已知函数$f(x)$在某区间$[a,b]$上一系列点上的值x_i的函数值$y_i = f(x_i), i = 0, 1, \cdots, n$

x	x_0	x_1	$\cdots$	x_n
$y = f(x)$	y_0	y_1	$\cdots$	y_n

函数一定经过原始数据点。插值就是用较简单、满足一定条件的函数$\varphi(x)$去代替$f(x)$，插值函数满足条件

$$\varphi(x_i) = y_i, i = 0, 1, \ldots, n \tag{1}$$

拟合问题：用一个函数去近似原函数，不要求过已知数据点，只要求在某种意义下它在这些点上的总偏差最小。



图片取自知乎

1 插值在数学建模中的应用

1.1 一维插值的Python实现

1.1.1 分段多项式插值：`interp1d`

在Scipy里可以用`scipy.interpolate`模块下的`interp1d`函数实现分段多项式插值，特点是插值凸点多，不够光滑。

```
from scipy.interpolate import interp1d # scipy一元插值
f_interp = interp1d(xdata, ydata, kind) #利用scipy进行插值
y_list = f_interp(x_list)   #利用插值多项式计算新点的函数值
```

> 分段插值方式有：kind='linear', 'zero', 'slinear', 'quadratic'(2次), 'cubic'(3次), 4, 5等。

例：根据下表数据插值计算$f(175)$的近似值。

x	144	169	196
$y=f(x)$	12	13	14

1.1.2 样条插值：`splev`，`splrep`

样条插值法是一种以可变样条来作出一条经过一系列点的光滑曲线的数学方法。插值样条是由一些多项式组成的，每一个多项式都是由相邻的两个数据点决定的，这样，任意两个相邻的多项式以及它们的导数在内部连接点处都是**连续的**。

`scipy.interpolate`包里提供了两个函数`splev`和`splrep`共同完成(B-样条: 贝兹曲线(又称贝塞尔曲线))插值，和之前一元插值`interp1d`一步就能完成不同，样条插值需要两步完成，第一步先用`splrep`计算出B样条曲线的参数tck，第二步在第一步的基础上用`splev`计算出各取样点的插值结果。

```
from scipy.interpolate import splev, splrep #样条插值
spl = splrep(xdata, ydata) #样条插值节点和系数
'''
splrep返回的tck：A tuple (t,c,k) containing the vector of knots, the B-spline coefficients, and the degree of the spline.
'''
y_list = splev(x_list, spl) #利用样条插值函数计算新点的函数值
```

1.1.3 一维插值案例：机翼轮廓线的加工设计

问题的提出：下表给出的数据位于机翼端面的轮廓线上，Y1和Y2分别对应轮廓的上下线。假设需要得到x每改变0.1时的数据点，

(1)试完成加工所需数据，画出曲线；

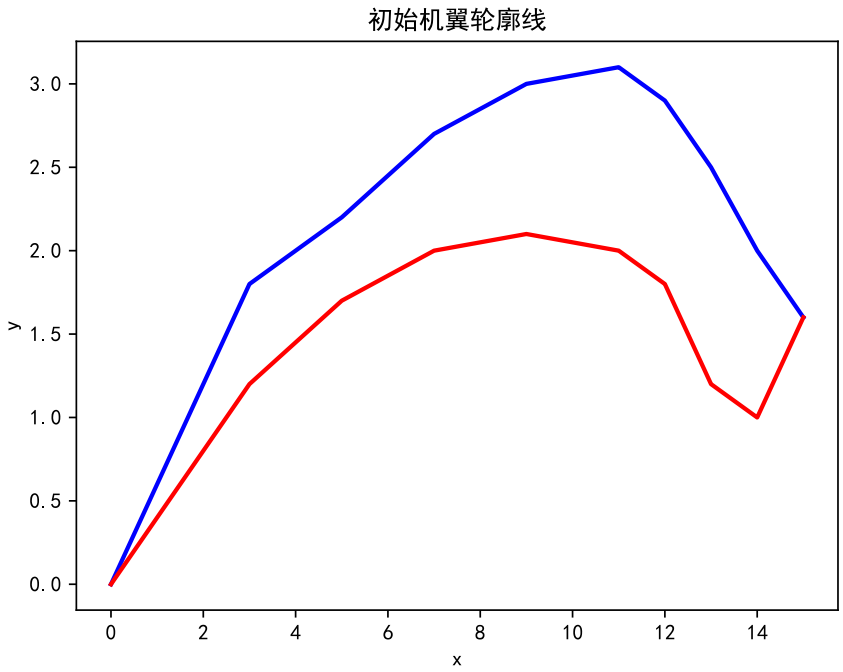
(2)求加工端面的面积。

x	0	3	5	7	9	11	12	13	14	15
Y1	0	1.8	2.2	2.7	3.0	3.1	2.9	2.5	2.0	1.6
Y2	0	1.2	1.7	2.0	2.1	2.0	1.8	1.2	1.0	1.6

（1）绘制初始图

```
#绘图显示机翼轮廓线原始数据
import matplotlib.pyplot as plt
import numpy as np

plt.rcParams["font.sans-serif"]=["SimHei"] #设置字体
plt.rcParams["axes.unicode_minus"]=False #该语句解决图像中的“-”负号的乱码问题

data = np.loadtxt(r'..\data\PlaneContour.txt') #读取数据
```

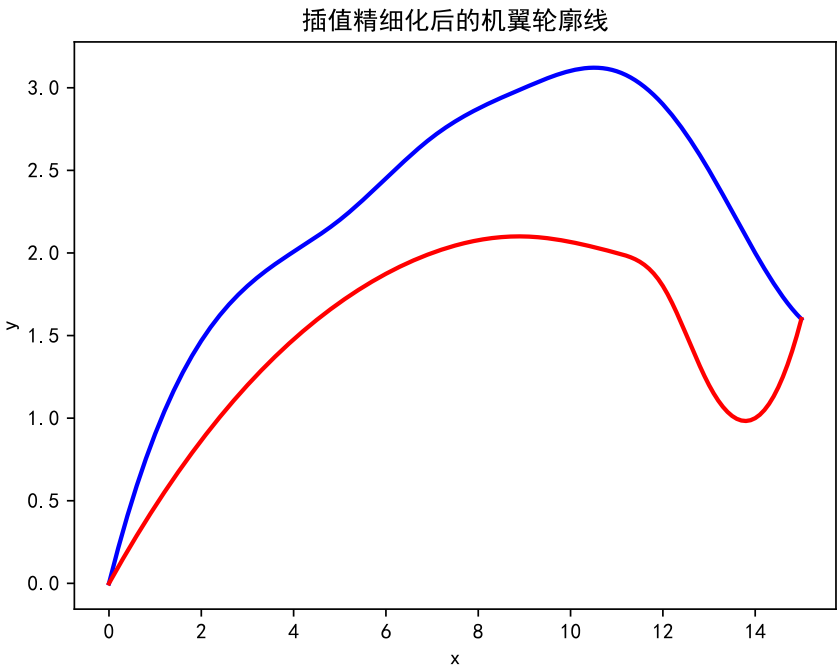
```
xdata = data[0]
ydata1 = data[1]
ydata2 = data[2]
plt.plot(xdata, ydata1, color='blue', linewidth=2)
plt.plot(xdata, ydata2, color='red',  linewidth=2)
plt.xlabel('x')
plt.ylabel('y')
plt.title('初始机翼轮廓线')
plt.savefig('初始机翼轮廓线.svg', dpi=500)
# plt.show()
```



(2) 加密后的效果图

```
# 插值得到精细的机翼轮廓线
import matplotlib.pyplot as plt
import numpy as np
from scipy.interpolate import interp1d # scipy一元插值
plt.rcParams["font.sans-serif"]=["SimHei"] #设置字体
plt.rcParams["axes.unicode_minus"]=False #该语句解决图像中的“-”负号的乱码问题

data = np.loadtxt(r'..\data\PlaneContour.txt') #读取数据
xdata = data[0]
ydata1 = data[1]
ydata2 = data[2]

x_new = np.arange(min(xdata), max(xdata)+0.01, 0.1) #不包括最后一个点

f1_interp = interp1d(xdata, ydata1, kind='cubic') #利用scipy进行插值
y1_new = f1_interp(x_new)   #利用插值多项式计算新点的函数值

f2_interp = interp1d(xdata, ydata2, kind='cubic') #利用scipy进行插值
y2_new = f2_interp(x_new)   #利用插值多项式计算新点的函数值
plt.plot(x_new, y1_new, color='blue', linewidth=2)
plt.plot(x_new, y2_new, color='red',  linewidth=2)
plt.xlabel('x')
```

```
plt.ylabel('y')
plt.title('插值精细化后的机翼轮廓线')
plt.savefig('插值精细化后的机翼轮廓线.svg', dpi=500)
# plt.show()
```



(3) 近似计算面积

数值积分

在SciPy中，提供了`scipy.intergrate`模块进行积分计算。通过散列的点进行数值积分计算主要有梯形法、复合梯形法、Simpson法、Romberg法。如下将展示复合梯形法与Simpson法的计算过程。

对于样本点间隔随机的情况，通常在梯形法和Simpson法中选择。梯形法和Simpson法分别使用一阶和二阶的Newton-Coates公式进行积分。梯形法将函数近似为相邻点间的直线进行计算，Simpson法将函数近似为3个相邻点之间的抛物线进行计算。**对于等距的奇数个样本，如果函数是三阶或更少阶的多项式，Simpson法是准确的。如果样本不等距，则只有当函数是二阶或更少阶才是准确的**

如下为使用梯形法、复合梯形法、Simpson法的示例代码。

```
import numpy as np
from scipy import integrate

def f1(x):
    return x ** 2

x = np.array([1, 3, 4])
y = f1(x)

# Simpson法
result = integrate.simps(y, x)
# 梯形法
result1 = integrate.trapz(y, x)
# 复合梯形法
result2 = integrate.cumtrapz(y, x)
print(result, result1, result2) # 21.0 22.5 22.5
```

```
# Calculate the area
Area = integrate.trapz(y1_new, x_new)-trapz(y2_new, x_new)
```

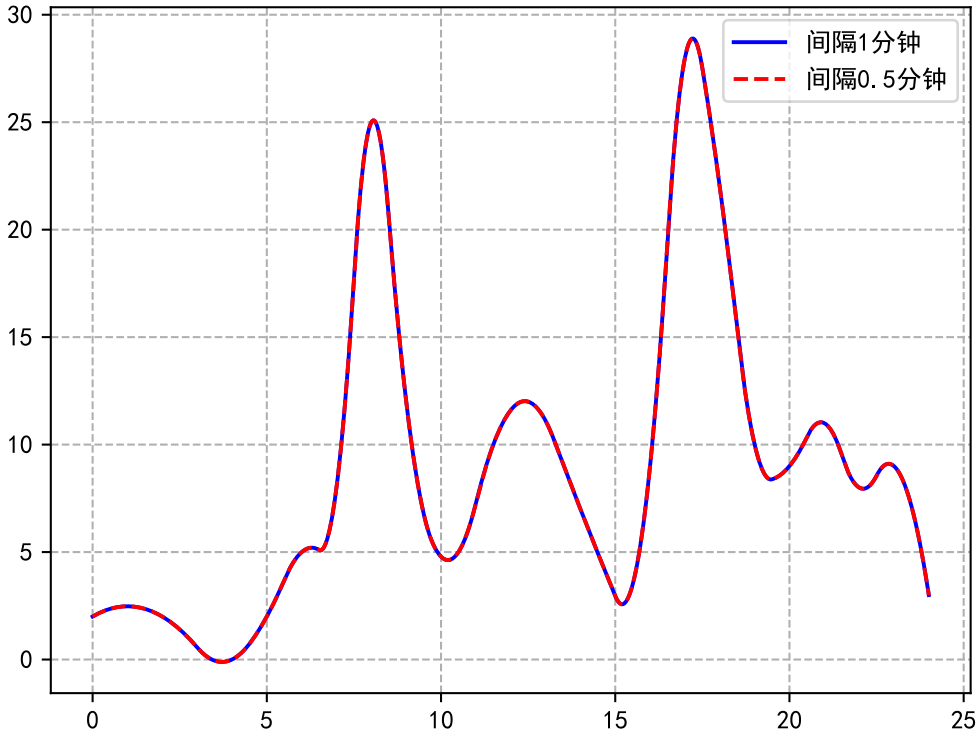
练习：交通管理部门为了掌握一座桥梁的通行情况，在桥梁的一端每隔一段不等的时间连续记录1 min内通过桥梁的车辆数量，连续观测一天24h的通过车辆车辆数据如下表所示。试建立模型分析估计这一天中总共有多少车辆通过这座桥梁。

24h 通过桥梁的车辆统计数据

时间	0:00	2:00	4:00	5:00	6:00	7:00	8:00
车辆数	2	2	0	2	5	8	25
时间	9:00	10:30	11:30	12:30	14:00	16:00	17:00
车辆数	12	5	10	12	7	9	28
时间	18:00	19:00	20:00	21:00	22:00	23:00	24:00
车辆数	22	10	9	11	8	9	3

提示：每分钟通过的车辆数为变化率（单位时间的车流量），插值得到更密的变化率，进行数值积分可得总的车辆数。

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.interpolate import interp1d
from scipy.integrate import trapz, simps #复合梯形和复合辛普森公式
plt.rcParams["font.sans-serif"]=["SimHei"] #设置字体
plt.rcParams["axes.unicode_minus"]=False #该语句解决图像中的“-”负号的乱码问题

X = np.array([
    0, 2, 4, 5, 6, 7, 8, 9, 10.5, 11.5, 12.5, 14, 16, 17, 18, 19, 20, 21, 22,
    23, 24
])
Y = np.array(
    [2, 2, 0, 2, 5, 8, 25, 12, 5, 10, 12, 7, 9, 28, 22, 10, 9, 11, 8, 9, 3])
xnew = np.linspace(0, 24, 24 * 60)
# 引入插值函数，这里的点较散乱，所以我选取高阶的样条插值
f = interp1d(X, Y, 'quadratic')
ynew = f(xnew)
# 计算一天的车辆总数
count = sum(ynew)
print(f'一天的车辆总数为:{np.round(count)}')
#利用数值积分求车辆数
x_new2 = np.linspace(0, 24, 24 * 60 * 2)  #得到每隔半分钟的数据
y_new2 = f(x_new2)
# Simpson法
count2 = trapz(y_new2, x=x_new2) * 60 #1小时的车流量转化为1分钟的求和
print(f'数值积分求得一天的车辆总数为:{np.round(count2)}')

plt.plot(xnew, ynew, c='b', label='间隔1分钟')
plt.plot(x_new2, y_new2, c='r', linestyle='--', label='间隔0.5分钟')
plt.grid(linestyle='--')  #设置网格线---这里是虚线
plt.legend()
plt.show()
```



1.2 二维插值的Python实现

1.2.1 scipy.interpolate.interp2d二维插值

```
from scipy.interpolate import interp2d
x1 = np.arange(a, b, step1)
y1 = np.arange(c, d, step2)
X1,Y1 = np.meshgrid(x1,y1) #二维数据生成
f1 = interp2d(X,Y,Z,kind)
z1 = f1(x1,y1)
```

> kind类型设置：
>
> - 线性插值: kind='linear'
> - 三次式插值: kind='cubic'
> - 五次式插值: kind='quintic'

1.2.2 插值案例2：逢山开路山貌复原（沙盘制作问题）

逢山开路问题：

```
# 逢山开路山貌精细化
import matplotlib.pyplot as plt
import numpy as np
from scipy.interpolate import interp2d

fig = plt.figure()
ax = fig.add_subplot(projection='3d')

x = np.arange(0, 5600 + 10, 400) #arange 不包括终点
```

```
y = np.arange(0, 4800 + 10, 400)

X, Y = np.meshgrid(x, y)
Z = np.loadtxt(r'../data/Kailu.txt') #读取数据
Z = Z[-1::-1, :] #调整原点坐标到矩阵的左上角

x_new = np.arange(0, 5600 + 10, 100) #arange 不包括终点
y_new = np.arange(0, 4800 + 10, 100)
f1 = interp2d(X, Y, Z, kind='cubic')
Z_new = f1(x_new, y_new)

X_new, Y_new = np.meshgrid(x_new, y_new) #绘图需要网格点矩阵

ax.plot_surface(X_new, Y_new, Z_new, cmap='rainbow')
ax.set_xlabel('x')
ax.set_ylabel('y')
ax.set_zlabel('z')
plt.show()
```

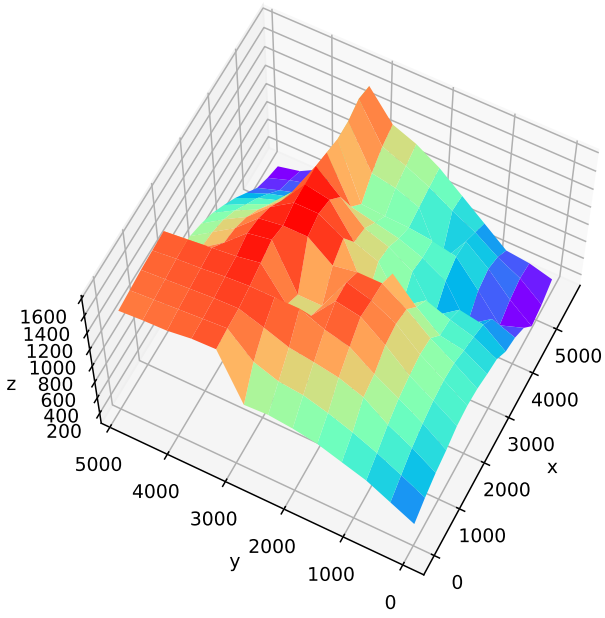


图1: 原始山貌图

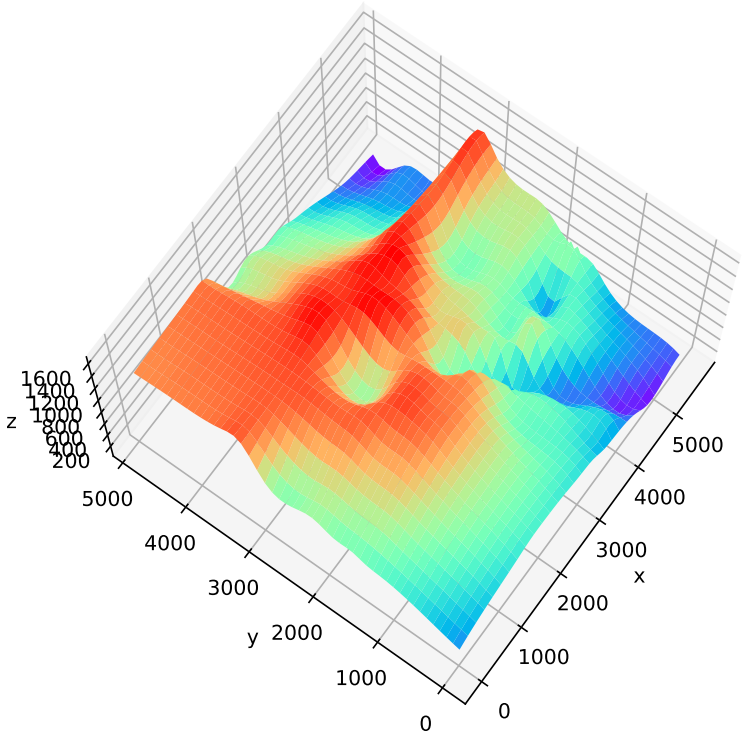
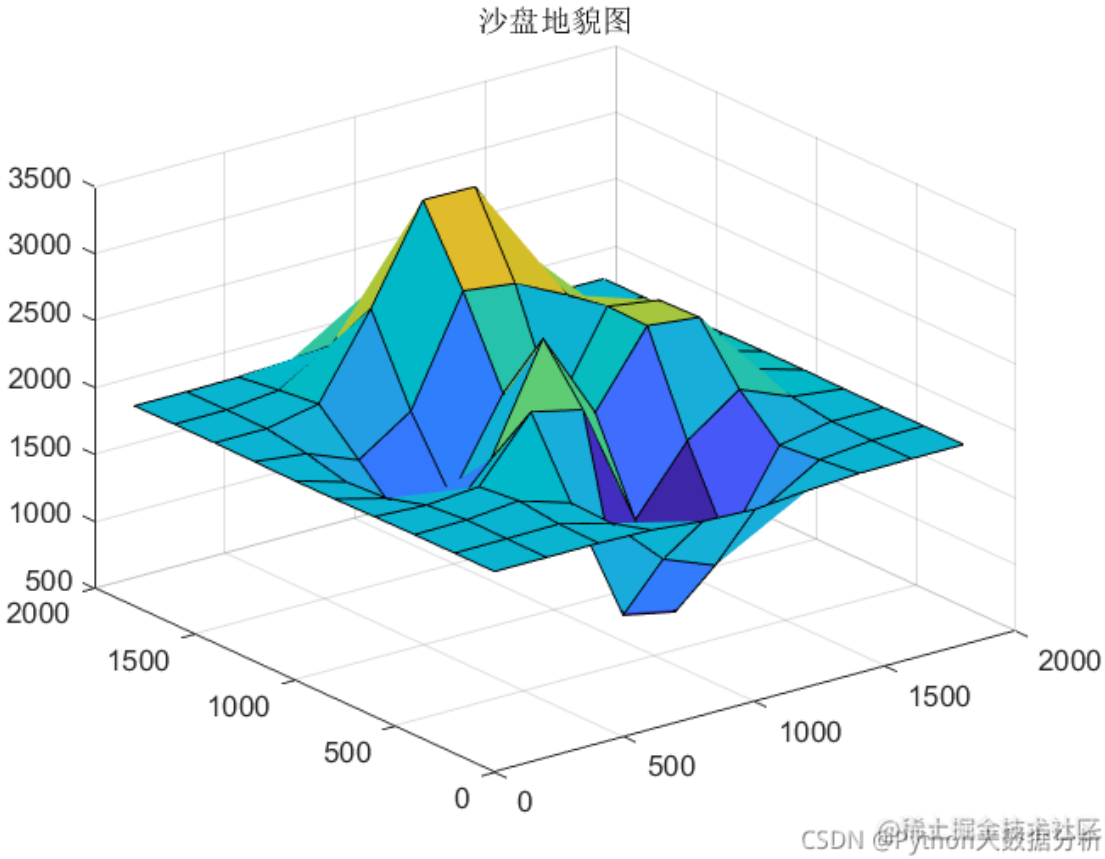
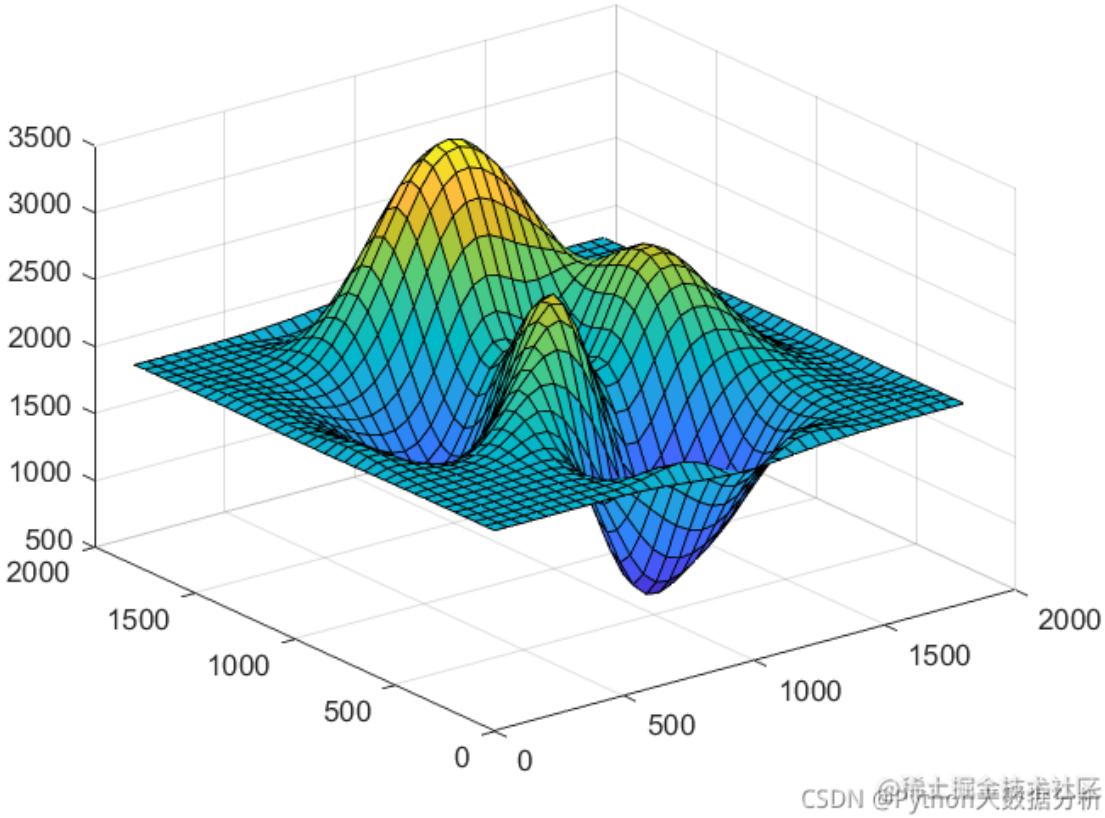
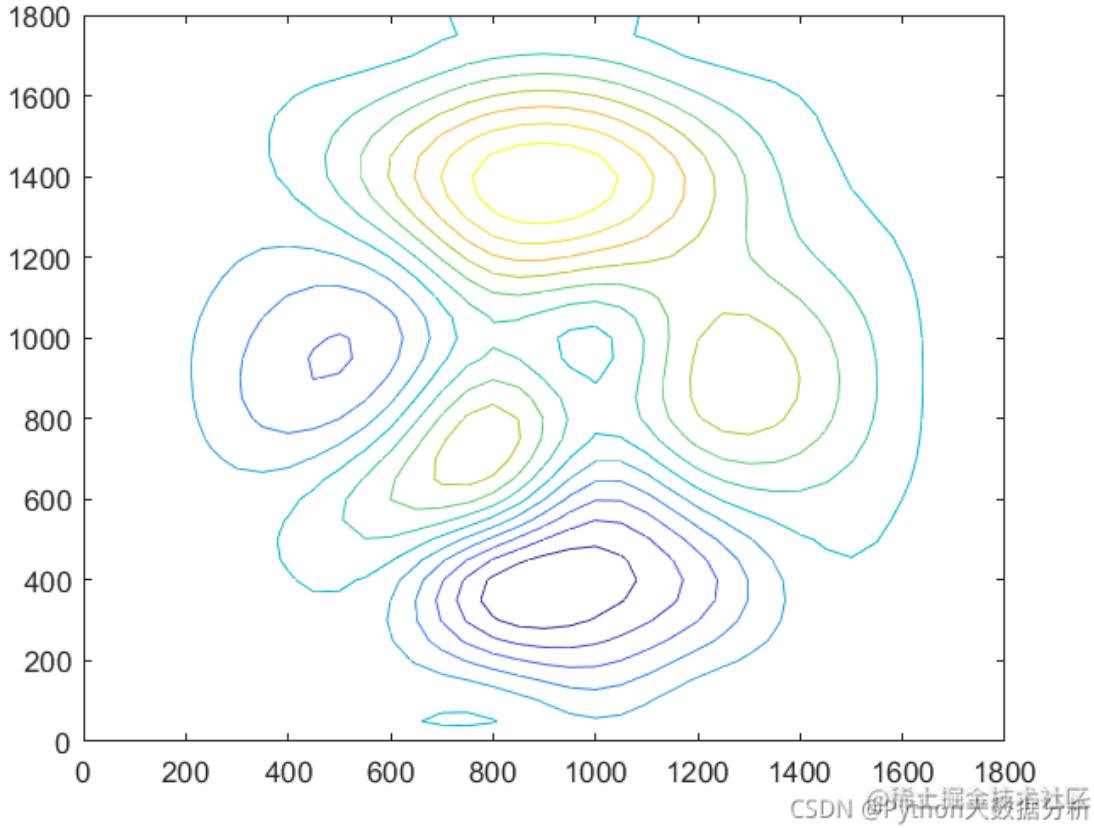


图2: 插值精细化后的山貌图

某地面部队分成红蓝两方在指定的陌生区域(平面区域[0,2000*[0,2000]内，单位:m) 进行作战演习。在演习过程中，红方侦查单位已经测得一些地点的高程如下表所示。①根据表中数据，制作军事沙盘。②在演习范围内，占领最大高地的一方将获得居高临下的优势。请问红方应第一时间抢 占哪块区域。







2 拟合在数学建模中的应用

拟合：用一个函数去近似原函数，不要求过已知数据点，只要求在某种意义下它在这些点上的总偏差最小。

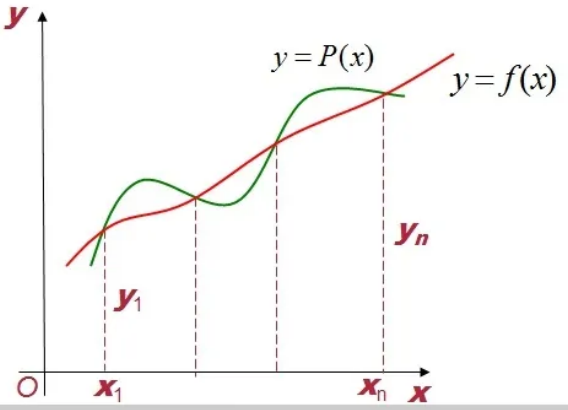
最小二乘拟合：已知函数$y=f(x;\alpha)$在一系列点上的值x_i的函数值$y_k=f(x_k;\alpha)$，$k=1,2,\cdots,m$，其中α为待定的拟合参数。

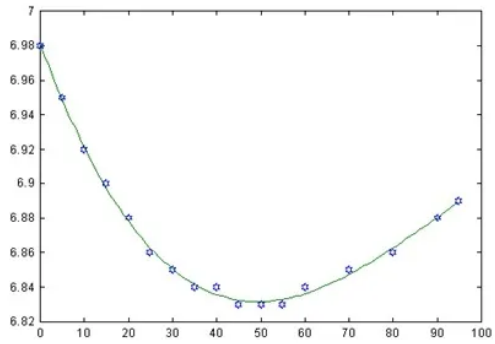
已知函数$f(x)$

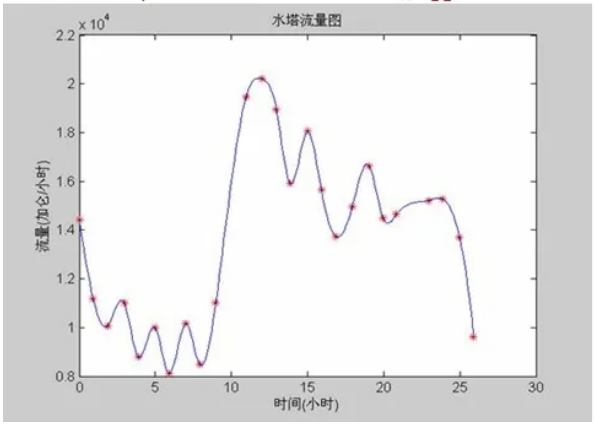
x	x_1	x_2	$\cdots$	x_m
$y=f(x;\alpha)$	y_1	y_2	$\cdots$	y_m

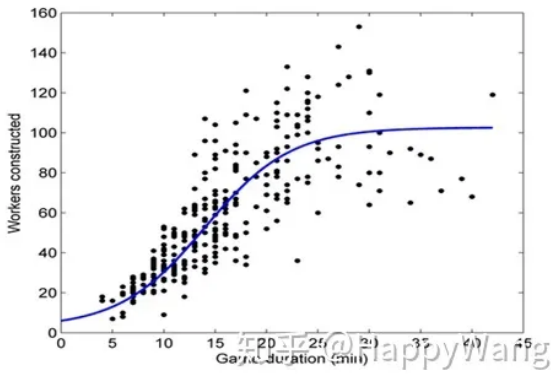
最小二乘拟合原理：转化为关于拟合参数α的最优化问题

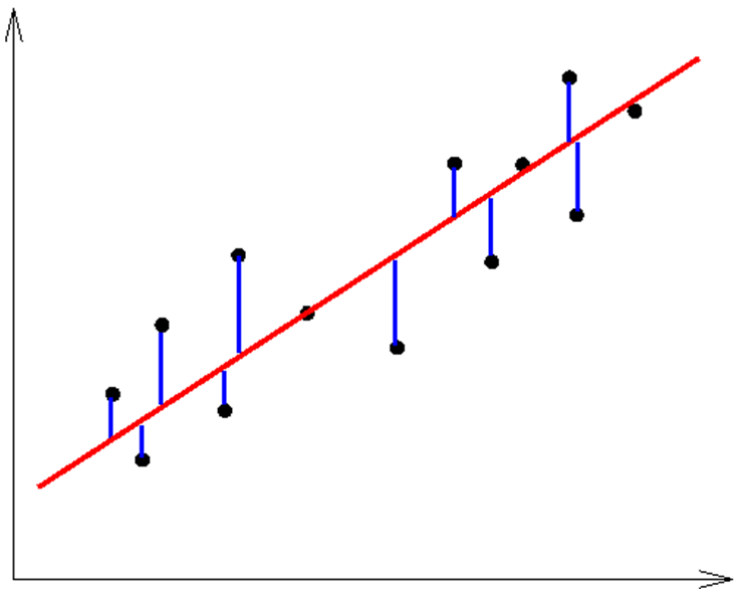
$$\min F(\alpha)=\sum_{k=1}^{m}\left[f(x_k;\alpha)-y_k\right]^2 \tag{2}$$











2.1 最小二乘拟合原理和算例

在某化学反应里，测得生成物的质量浓度$y(10^{-3}\mathrm{g/cm}^3)$与时间$t(\mathrm{min})$的关系如下表。为了研究该化学反应的性质，如反应速度等，试求y与t之间的连续函数关系式$y=f(t)$.

t	1	2	3	4	5	6	7	8	9	10
y	4.00	6.41	8.01	8.79	9.53	9.86	10.33	10.42	10.53	10.61

1．用线性模型$\varphi(t)=a+bt+ct^2$拟合上表数据，求拟合函数。

2．若选取非线性模型$\varphi(t)=ae^{\frac{b}{t}}$拟合上述数据，求$a,b$.

3．若选取线性模型$\varphi(t)=\dfrac{t}{at+b}$拟合上述数据，求$a,b$.

4．试分别计算上述三种拟合方法的平方误差，并进行比较，做出上面三种拟合效果图。

实验原理:

已知函数值表

x	x_0	x_1	$\cdots$	x_m
$y=f(x)$	y_0	y_1	$\cdots$	y_m

求函数$S(x;a_0,a_1,\cdots,a_n)=a_0\varphi_0(x)+a_1\varphi_1(x)+\cdots+a_n\varphi_n(x)$按最小二乘原则拟合函数$f(x)$，即

$$\min F(a_0,\cdots,a_n)=\min\sum_{k=0}^{m}\left[a_0\varphi_0(x_k)+a_1\varphi_1(x_k)+\cdots+a_n\varphi_n(x_k)-y_k\right]^2 \tag{3}$$

根据多元函数求极值的方法，$a_0,a_1,\cdots,a_n$需满足

$$\begin{cases}\dfrac{\partial F}{\partial a_0}=0\\ \dfrac{\partial F}{\partial a_1}=0\\ \cdots\\ \dfrac{\partial F}{\partial a_n}=0\end{cases} \tag{4}$$

整理，得

$$\begin{pmatrix}\sum\limits_{k=0}^{m}\varphi_0(x_k)\varphi_0(x_k) & \cdots & \sum\limits_{k=0}^{m}\varphi_0(x_k)\varphi_n(x_k)\\ \cdots & \ddots & \cdots\\ \sum\limits_{k=0}^{m}\varphi_n(x_k)\varphi_0(x_k) & \cdots & \sum\limits_{k=0}^{m}\varphi_n(x_k)\varphi_n(x_k)\end{pmatrix}\begin{pmatrix}a_0\\ \cdots\\ a_n\end{pmatrix}=\begin{pmatrix}\sum\limits_{k=0}^{m}y_k\varphi_0(x_k)\\ \cdots\\ \sum\limits_{k=0}^{m}y_k\varphi_n(x_k)\end{pmatrix} \tag{5}$$

引入离散内积

$$(\varphi_i,\varphi_j)=\sum_{k=0}^{m}\varphi_i(x_k)\varphi_j(x_k),\ i,j=0,1,\cdots,n \tag{6}$$

这样，上述方程组可以化为

$$\begin{pmatrix} (\varphi_0,\varphi_0) & (\varphi_0,\varphi_1) & \cdots & (\varphi_0,\varphi_n) \\ \cdots & \cdots & \ddots & \cdots \\ (\varphi_n,\varphi_0) & (\varphi_n,\varphi_1) & \cdots & (\varphi_n,\varphi_n) \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \cdots \\ a_n \end{pmatrix} = \begin{pmatrix} (f,\varphi_0) \\ \cdots \\ (f,\varphi_n) \end{pmatrix}$$

该方程组通常称为法方程组。引入矩阵

$$A_{(m+1)\times(n+1)} = \begin{pmatrix} \varphi_0(x_0) & \varphi_1(x_0) & \cdots & \varphi_n(x_0) \\ \varphi_0(x_1) & \varphi_1(x_1) & \cdots & \varphi_n(x_1) \\ \cdots & \cdots & \ddots & \cdots \\ \varphi_0(x_m) & \varphi_1(x_m) & \cdots & \varphi_n(x_m) \end{pmatrix} \tag{7}$$

这样利用矩阵乘法可以把法方程组写为

$$A^T A\alpha = A^T Y \tag{8}$$

其中$\alpha = (a_0, a_1, \cdots, a_n)^T, Y = (y_0, y_1, \cdots, y_n)^T.$

实验过程：1. 用线性模型$\varphi(t) = a + bt + ct^2$拟合上表数据时，取$\varphi_0(t) = 1, \varphi_1(t) = t, \varphi_2(t) = t^2$, 求解法方程组即可得拟合系数$a, b, c$。

2. 用非线性模型$\varphi(t) = ae^{\frac{b}{t}}$拟合上述数据时，两边取对数

$$\ln\varphi(t) = \ln a + \frac{b}{t} \tag{9}$$

令$\Psi = \ln\varphi(t), a' = \ln a$, 取$\varphi_0(t) = 1, \varphi_1(t) = \dfrac{1}{t}$, 则

$$\Psi = a'\varphi_0(t) + b\varphi_1(t) \tag{10}$$

需要准备好拟合数据(t_i, Ψ_i)，然后转化为第1种情形。

3. 类似第2种情形。

2.2 基于scipy的`leastsq`函数的最小二乘拟合

利用scipy提供的最小二乘工具来验证上面的拟合结果的准确性。

`scipy.optimize`包里提供了最小二乘法的功能函数`leastsq()`，最小二乘法是一种数学优化技术，最小二乘法还可用于曲线拟合。它通过最小化误差的平方和寻找数据的最佳函数匹配。具体拟合步骤可以参考下面的示例。

```
'''
利用scipy.optimize模块的最小二乘拟合函数leastsq()
与独立编写程序的结果进行比较和验证
'''
import matplotlib.pyplot as plt
import numpy as np
from scipy.optimize import leastsq

# #绘图显示中文
plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False

'''
step1: 定义拟合函数
```

```
'''
#拟合函数1：y = a + b t + c t**2
def fitfun1(alpha, t):
    '''
    :param alpha: 拟合函数的参数列表alpha = [a, b, c]
    :param t: 拟合函数的自变量
    :return: y: 拟合函数的因变量
    '''
    a, b, c = alpha #拟合系数
    y = a * t**2 + b * t +c
    return y

#拟合函数2：y = a exp(b/t)
def fitfun2(alpha, t):
    '''
    :param alpha: 拟合函数的参数列表alpha = [a, b]
    :param t: 拟合函数的自变量
    :return: y: 拟合函数的因变量
    '''
    '''
    ------------------------------
    这里作为作业思考，请根据你的理解补充完整
    ----------------------------
    '''


#拟合函数3：y = t/(at+b)
def fitfun3(alpha, t):
    '''
    :param alpha: 拟合函数的参数列表alpha = [a, b]
    :param t: 拟合函数的自变量
    :return: y: 拟合函数的因变量
    '''
    '''
    ------------------------------
    这里作为作业思考，请根据你的理解补充完整
    ----------------------------
    '''


'''
step2: 定义残差项
'''
def error1(alpha, t, y):
    '''
    :param alpha: 拟合函数的未知参数
    :param t:
    :param y:
    :return:
    '''
    return fitfun1(alpha, t) - y

def error2(alpha, t, y):
    '''
    :param alpha: 拟合函数的未知参数
```

```
    :param t:
    :param y:
    :return:
    '''
    '''
    ------------------------------
    这里作为作业思考，请根据你的理解补充完整
    ----------------------------
    '''

def error3(alpha, t, y):
    '''
    :param alpha: 拟合函数的未知参数
    :param t:
    :param y:
    :return:
    '''
    '''
    ------------------------------
    这里作为作业思考，请根据你的理解补充完整
    ----------------------------
    '''

'''
step3: 进行拟合
'''
def fit_main(tdata, ydata):
    '''
    :param tdata: 拟合数据表
    :param ydata:
    :return: 绘图观察拟合效果
    '''
    # 对第一个拟合函数进行参数拟合
    alpha1_0 = np.array([0.1, 0.01, 1])  # 拟合的初始参数设置
    para1 = leastsq(error1, alpha1_0, args=(tdata, ydata))  # 进行拟合
    alpha1 = para1[0] #拟合参数结果

    '''
    ------------------------------
    这里作为作业思考，请根据你的理解补充完整
    ----------------------------
    '''

    #绘图观察拟合效果
    tdata_new = np.linspace(min(tdata), max(tdata), 200)
    # y = a + bt+c t**2
    ydata1_new = fitfun1(alpha1, tdata_new)
    # y =a exp(b/t)
    '''
    ------------------------------
    这里作为作业思考，请根据你的理解补充完整
    ----------------------------
    '''
    # y = t/(a t + b)
    '''
```

```
    ------------------------------
    这里作为作业思考，请根据你的理解补充完整
    ----------------------------
    '''

    plt.plot(tdata, ydata, 'ro', markersize=5, linewidth=2, label='原始数据')
    plt.plot(tdata_new, ydata1_new, '--', linewidth=2, label='$y=a+bt+ct^2$')
    plt.plot(tdata_new, ydata2_new, '--', linewidth=2, label='$y=a e^{b/t}$')
    plt.plot(tdata_new, ydata3_new, '--', linewidth=2, label='$y=t / (at+b)$')
    plt.legend()
    plt.xlabel('t(min)')
    plt.ylabel('$10^3$ g/cm$^3$')
    plt.title('scipy leastsq最小二乘拟合效果')
    plt.savefig('scipy_leastsq_result.svg', dpi=500)


if __name__ == '__main__':
    #拟合数据
    tdata = np.arange(1, 11)
    ydata = np.array([4.00, 6.41, 8.01, 8.79, 9.53, 9.86, 10.33, 10.42, 10.53, 10.61])
    #使用scipy的leastsq()来验证手工编写的最小二乘结果
    fit_main(tdata=tdata, ydata=ydata)
```

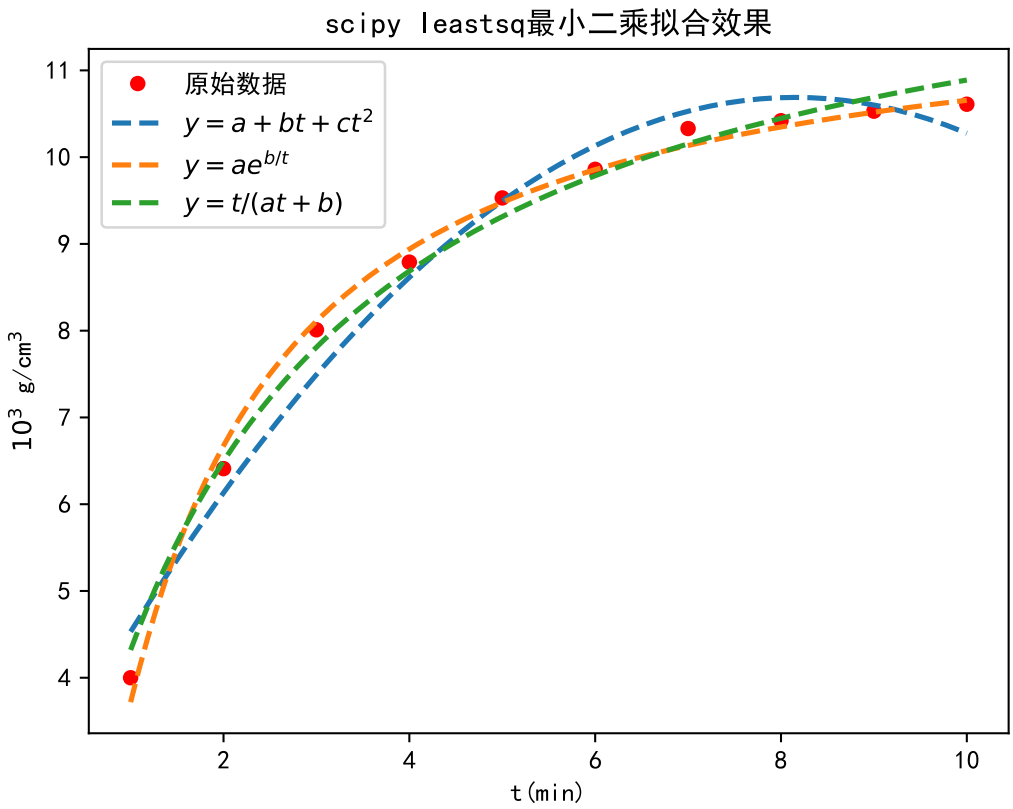


图3: scipy提供的leastsq工具的最小二乘拟合效果

结果分析：三种曲线均能反映数据的增长趋势，从均方误差来看，非线性模型$\varphi(t)=11.816521e^{\frac{-1.112608}{t}}$拟合效果最好。

2.3 基于scipy的`curve_fit`函数的最小二乘拟合

利用scipy提供的最小二乘工具来验证上面的拟合结果的准确性。

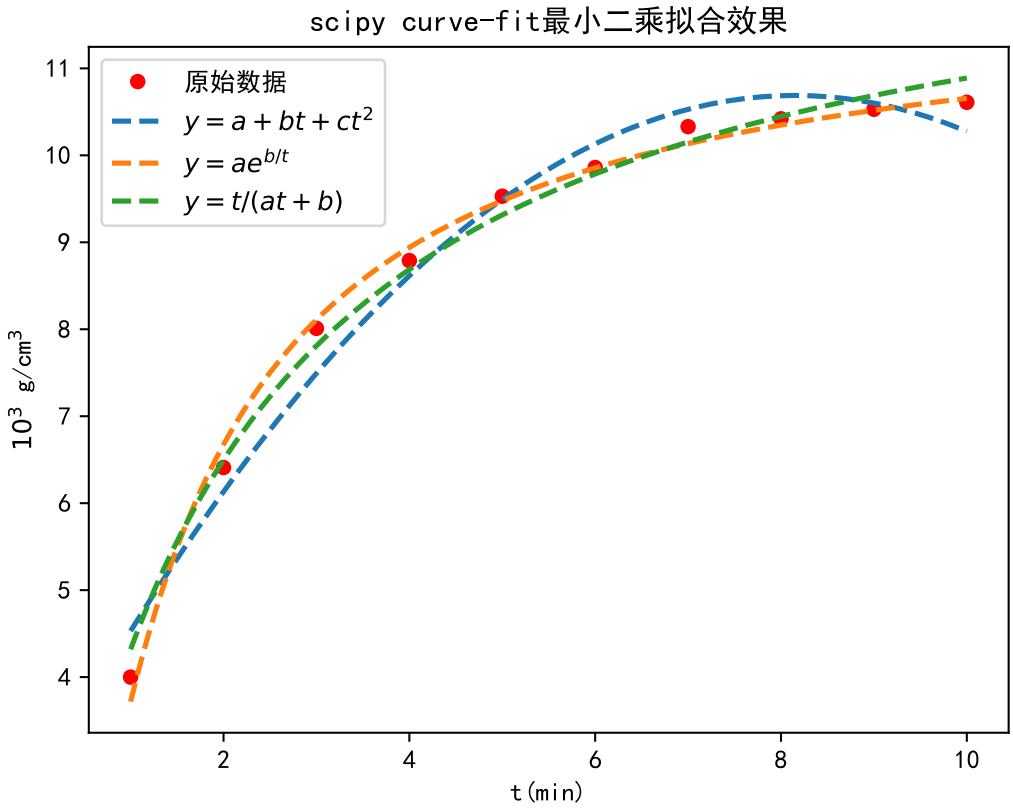
`scipy.optimize`包里提供了最小二乘法的功能函数`curve_fit()`，具体拟合步骤可以参考下面的示例。

```
'''
利用scipy.optimize模块的最小二乘拟合函数curve_fit()
与独立编写程序的结果进行比较和验证
'''
import matplotlib.pyplot as plt
import numpy as np
from scipy.optimize import curve_fit

# #绘图显示中文
plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False

'''
step1: 定义拟合函数
'''
#拟合函数1：y = a + b t + c t**2
def fitfun1(t, a, b, c):
    '''
    :param a, b, c: 拟合函数的参数
    :param t: 拟合函数的自变量
    :return: y: 拟合函数的因变量
    '''
    y = a + b * t + c * t**2
    return y

#拟合函数2：:y = a exp(b/t)
def fitfun2(t, a, b):
    '''
    :param a, b: 拟合函数的参数
    :param t: 拟合函数的自变量
    :return: y: 拟合函数的因变量
    '''
    '''
    ------------------------------
    这里作为作业思考，请根据你的理解补充完整
    ----------------------------
    '''


#拟合函数3：y = t/(at+b)
def fitfun3(t, a, b):
    '''
    :param a, b: 拟合函数的参数
    :param t: 拟合函数的自变量
    :return: y: 拟合函数的因变量
```

```
    '''
    '''
    ------------------------------
    这里作为作业思考，请根据你的理解补充完整
    ----------------------------
    '''
'''
step2: 进行拟合
'''
def fit_main(tdata, ydata):
    '''
    :param tdata: 拟合数据表
    :param ydata:
    :return: 绘图观察拟合效果
    '''
    # 对第一个拟合函数进行参数拟合
    para1, pcov1 = curve_fit(fitfun1, tdata, ydata)  # 进行拟合
    print(f'para1 = {para1}')
    a1, b1, c1 = para1 #获取拟合参数结果

    para2, pcov2 = curve_fit(fitfun2, tdata, ydata)  # 进行拟合
    print(f'para2 = {para2}')
    a2, b2 = para2 #拟合参数结果
    #拟合曲线3时，达不到拟合效果，引入bounds参数：Constrain the optimization to the region of ``0 <= a <= 0.1``, ``0 <= b <= 0.5``:
    para3, pcov3 = curve_fit(fitfun3, tdata, ydata, bounds=(0, [0.1, 0.5])) # 进行拟合
    print(f'para3 = {para}')
    a3, b3 = para3 #拟合参数结果

    #绘图观察拟合效果
    tdata_new = np.linspace(min(tdata), max(tdata), 200)
    # y = a + bt+c t**2
    ydata1_new = fitfun1(tdata_new, a1, b1, c1)
    # y =a exp(b/t)
    '''
    ------------------------------
    这里作为作业思考，请根据你的理解补充完整
    ----------------------------
    '''
    # y = t/(a t + b)
    '''
    ------------------------------
    这里作为作业思考，请根据你的理解补充完整
    ----------------------------
    '''
    plt.plot(tdata, ydata, 'ro', markersize=5, linewidth=2, label='原始数据')
    plt.plot(tdata_new, ydata1_new, '--', linewidth=2, label='$y=a+bt+ct^2$')
    plt.plot(tdata_new, ydata2_new, '--', linewidth=2, label='$y=a e^{b/t}$')
    plt.plot(tdata_new, ydata3_new, '--', linewidth=2, label='$y=t / (at+b)$')
    plt.legend()
```

```
    plt.xlabel('t(min)')
    plt.ylabel('$10^3$ g/cm$^3$')
    plt.title('scipy curve-fit最小二乘拟合效果')
    plt.savefig('scipy_curve_fit_result.svg', dpi=500)


if __name__ == '__main__':
    #拟合数据
    tdata = np.arange(1, 11)
    ydata = np.array([4.00, 6.41, 8.01, 8.79, 9.53, 9.86, 10.33, 10.42,
10.53, 10.61])
    #使用scipy的curve_fit()
    fit_main(tdata=tdata, ydata=ydata)
```

```
para1 = [ 2.68716669  1.96378029 -0.12049242]
para2 = [11.9777792  -1.16969515]
para3 = [0.07631912 0.15517181]
```



2.4 最小二乘拟合案例1：人口增长预测问题

下表为中国1971年到1990年人口数的统计数据（单位：亿），建立我国人口数量增长的近似曲线，并用它来预测2020年、2030年和2050年的人口数。

年份	1971	1972	1973	1974	1975	1976	1977	1978	1979	1980
人口	8.523	8.718	8.921	9.086	9.242	9.372	9.497	9.626	9.754	9.871
年份	1981	1982	1983	1984	1985	1986	1987	1988	1989	1990
人口	10.007	10.165	10.301	10.436	10.585	10.751	10.93	11.103	11.27	11.433

实验过程：

（1）**观察人口增长趋势**

```python
import matplotlib.pyplot as plt
import numpy as np

# #绘图显示中文
plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False

data = np.loadtxt(r'../data/populataion.txt') #读取数据
tdata = data[[0, 2], :].reshape((1, -1))[0] #把原始数据的第1,3行拼接成新的一行
xdata = data[[1, 3], :].reshape((1, -1))[0]

plt.plot(tdata, xdata, 'o-', linewidth=2)
plt.xlabel('年份')
plt.ylabel('人口数量（单位：亿）')
plt.show()
```

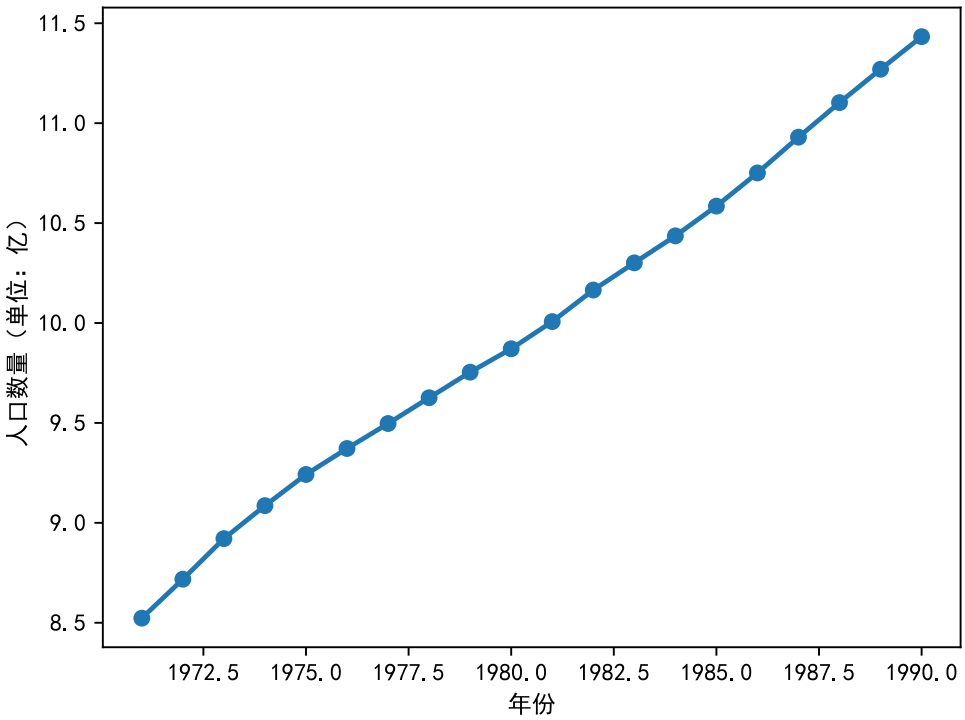


图4: 观察人口增长趋势

（2）**选择人口增长模型**

人口增长两个基本模型：

a）**Malthus指数增长模型**

设t时刻的人口数量为$x(t)$，增长率（单位时间单位人口的增长量）为r，可利用微元法建立微分方程模型

$$\begin{cases} \dfrac{dx(t)}{dt} = rx(t) \\ x(t_0) = x_0 \end{cases} \tag{11}$$

求解微分方程，可得

$$x(t) = x_0 e^{r(t-t_0)} \tag{12}$$

b)**Logistic阻滞增长模型**

$$x(t) = \frac{x_m}{1 + \left(\frac{x_m}{x_0} - 1\right)e^{-r(t-t_0)}} \tag{13}$$

利用Python的`scipy.optimize`包里最小二乘法函数`leastsq()`拟合参数，并进行预测

```
'''
利用scipy.optimize模块的最小二乘拟合函数leastsq()
'''
import matplotlib.pyplot as plt
import numpy as np
from scipy.optimize import leastsq

# #绘图显示中文
plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False

data = np.loadtxt(r'../data/populataion.txt') #读取数据
tdata = data[[0, 2], :].reshape((1, -1))[0] #把原始数据的第1,3行拼接成新的一行
xdata = data[[1, 3], :].reshape((1, -1))[0]

'''
step1: 定义拟合函数
'''
#拟合函数1：指数增长人口模型 x = x0 exp(r*(t-t0))
def popu1(alpha, t):
    '''
    :param r: 拟合函数的参数列表r
    :param t: 拟合函数的自变量年份
    :return: x: 拟合函数的因变量人口数量
    '''
    r = alpha[0]
    t0 = 1971
    x0 = 8.523
    x = x0 * np.exp(r*(t-t0))
    return x

#拟合函数2：阻滞增长人口模型 x = xm/(1+(xm/x0-1)*exp(-r(t-t0)))
def popu2(alpha, t):
    '''
    :param alpha: 拟合函数的参数列表alpha = [xm, r]
    :param t: 拟合函数的自变量，年份
    :return: y: 拟合函数的因变量，人口数量
    '''
    xm, r = alpha #拟合系数
    t0 = 1971
    x0 = 8.523
    x = xm / (1 + (xm/x0-1) * np.exp(-r*(t-t0)))
    return x
'''
step2: 定义残差项
'''
def error1(r, t, y):
    #指数增长模型的误差
```

```python
    return popu1(r, t) - y

def error2(alpha, t, y):
    #阻滞增长模型的误差
    '''
    ------------------------------
    这里作为作业思考，请根据你的理解补充完整
    ----------------------------
    '''
    return popu2(alpha, t) - y


'''step3: 进行拟合'''
def fit_main(tdata, xdata):
    '''
    :param tdata: 拟合数据表
    :param xdata:
    :return: 绘图观察拟合效果
    '''
    # 对第一个拟合函数进行参数拟合
    alpha0 = [0.1]  # 指数增长模型拟合参数的初始设置
    para1 = leastsq(error1, alpha0, args=(tdata, xdata))  # 进行拟合
    alpha1 = para1[0] # 拟合参数结果

    t2020 = 2020 #2020年份
    t2030 = 2030
    t2050 = 2050
    x2020 = popu1(alpha1, t2020)
    x2030 = popu1(alpha1, t2030)
    x2050 = popu1(alpha1, t2050)
    print('----' * 10)
    print(f'指数增长模型预测{t2020}年人口值为：{x2020}亿')
    print(f'指数增长模型预测{t2030}年人口值为：{x2030}亿')
    print(f'指数增长模型预测{t2050}年人口值为：{x2050}亿')


    alpha0 = np.array([20, 0.1]) # 指数增长模型拟合参数的初始设置
    para2 = leastsq(error2, alpha0, args=(tdata, xdata))  # 进行拟合
    alpha2 = para2[0] # 拟合参数结果

    x2020_Logistic = popu2(alpha2, t2020)
    x2030_Logistic = popu2(alpha2, t2030)
    x2050_Logistic = popu2(alpha2, t2050)
    print('----' * 10)
    print(f'阻滞增长模型预测{t2020}年人口值为：{x2020_Logistic}亿')
    print(f'阻滞增长模型预测{t2030}年人口值为：{x2030_Logistic}亿')
    print(f'组织增长模型预测{t2050}年人口值为：{x2050_Logistic}亿')


    #绘图观察拟合效果
    tdata_new = np.linspace(min(tdata)-20, max(tdata) + 60, 200) #更大时间范围观察拟合效果
    xdata1_new = popu1(alpha1, tdata_new)
    xdata2_new = popu2(alpha2, tdata_new)

```

```
    plt.plot(tdata, xdata, 'ro', markersize=5, linewidth=2, label='原始数据')
    plt.plot(tdata_new, xdata1_new, linewidth=2, label='指数增长人口模型')
    plt.plot(tdata_new, xdata2_new, linewidth=2, label='阻滞增长人口模型')
    plt.legend()
    plt.xlabel('年份')
    plt.ylabel('人口数量（单位：亿）')
    plt.title('不同人口增长模型的最小二乘拟合效果')
    plt.savefig('不同人口增长模型的最小二乘拟合效果比较.svg', dpi=500)

if __name__ == '__main__':
    fit_main(tdata, xdata)
```

预测2020年、2030年和2050年的人口数：

```
----------------------------------------
指数增长模型预测2020年人口值为：18.438435108080167亿
指数增长模型预测2030年人口值为：21.58332037576629亿
指数增长模型预测2050年人口值为：29.573767450211655亿
----------------------------------------
阻滞增长模型预测2020年人口值为：14.720763606548834亿
阻滞增长模型预测2030年人口值为：15.440761267117674亿
组织增长模型预测2050年人口值为：16.400339867662943亿
```

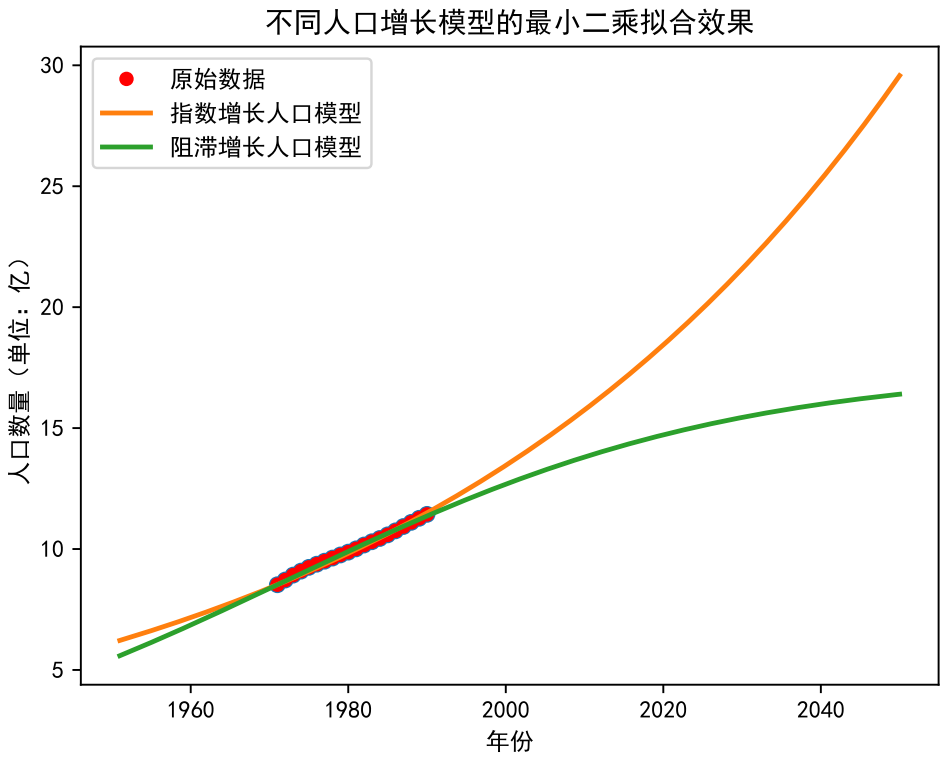


图5: 不同人口增长模型的最小二乘拟合效果比较（leastsq）

同理，可以利用`curve_fit()`来对人口增长进行预测。

```
'''
利用scipy.optimize模块的最小二乘拟合函数leastsq()
'''
import matplotlib.pyplot as plt
```

```python
import numpy as np
from scipy.optimize import curve_fit

# #绘图显示中文
plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False

data = np.loadtxt(r'../data/populataion.txt') #读取数据
tdata = data[[0, 2], :].reshape((1, -1))[0] #把原始数据的第1,3行拼接成新的一行
xdata = data[[1, 3], :].reshape((1, -1))[0]

plt.plot(tdata, xdata, 'o-', linewidth=2)
plt.xlabel('年份')
plt.ylabel('人口数量（单位：亿）')
# plt.savefig('观察人口增长趋势.svg', dpi=500)
# plt.show()
'''
step1: 定义拟合函数
'''
#拟合函数1：指数增长人口模型 x = x0 exp(r*(t-t0))
def popu1(t, r):
    '''
    :param r: 拟合函数的参数r
    :param t: 拟合函数的自变量年份
    :return: x: 拟合函数的因变量人口数量
    '''
    t0 = 1971
    x0 = 8.523
    x = x0 * np.exp(r*(t-t0))
    return x

#拟合函数2：阻滞增长人口模型 x = xm/(1+(xm/x0-1)*exp(-r(t-t0)))
def popu2(t, r, xm):
    '''
    :param r, xm: 拟合函数的参数r--增长率，xm--环境能够承受的最大人口数量
    :param t: 拟合函数的自变量，年份
    :return: y: 拟合函数的因变量，人口数量
    '''
    t0 = 1971
    x0 = 8.523
    x = xm / (1 + (xm/x0-1) * np.exp(-r*(t-t0)))
    return x

'''step2: 进行拟合'''
def fit_main(tdata, xdata):
    '''
    :param tdata: 拟合数据表
    :param xdata:
    :return: 绘图观察拟合效果
    '''
    # 对第一个拟合函数进行参数拟合
    para1, pcov = curve_fit(popu1, tdata, xdata)  # 进行拟合
    # 返回para1: 参数列表，pcov误差或协方差
    print(para1,pcov)
    r1 = para1[0] # 拟合参数结果
```

```
    t2020 = 2020 #2020年份
    t2030 = 2030
    t2050 = 2050
    x2020 = popu1(t2020, r1)
    x2030 = popu1(t2030, r1)
    x2050 = popu1(t2050, r1)
    print('----' * 10)
    print(f'指数增长模型预测{t2020}年人口值为：{x2020}亿')
    print(f'指数增长模型预测{t2030}年人口值为：{x2030}亿')
    print(f'指数增长模型预测{t2050}年人口值为：{x2050}亿')

    '''
    ------------------------------
    下面作为作业思考，请根据你的理解补充完整
    ------------------------------
    '''
    para2, pcov2 =  # 进行拟合
    r2 =  # 获取para2的拟合系数
    xm =

    x2020_Logistic =
    x2030_Logistic =
    x2050_Logistic =
    print('----' * 10)
    print(f'阻滞增长模型预测{t2020}年人口值为：{x2020_Logistic}亿')
    print(f'阻滞增长模型预测{t2030}年人口值为：{x2030_Logistic}亿')
    print(f'组织增长模型预测{t2050}年人口值为：{x2050_Logistic}亿')


    #绘图观察拟合效果
    tdata_new = np.linspace(min(tdata)-20, max(tdata) + 60, 200) #更大时间范围观察拟合效果
    xdata1_new = popu1(tdata_new, r1)
    xdata2_new = popu2(tdata_new, r2, xm)

    plt.plot(tdata, xdata, 'ro', markersize=5, linewidth=2, label='原始数据')
    plt.plot(tdata_new, xdata1_new, linewidth=2, label='指数增长人口模型')
    plt.plot(tdata_new, xdata2_new, linewidth=2, label='阻滞增长人口模型')
    plt.legend()
    plt.xlabel('年份')
    plt.ylabel('人口数量（单位：亿）')
    plt.title('不同人口增长模型的最小二乘拟合效果')
    plt.savefig('不同人口增长模型的最小二乘拟合效果比较-curve_fit.svg', dpi=500)

if __name__ == '__main__':
    fit_main(tdata, xdata)
```

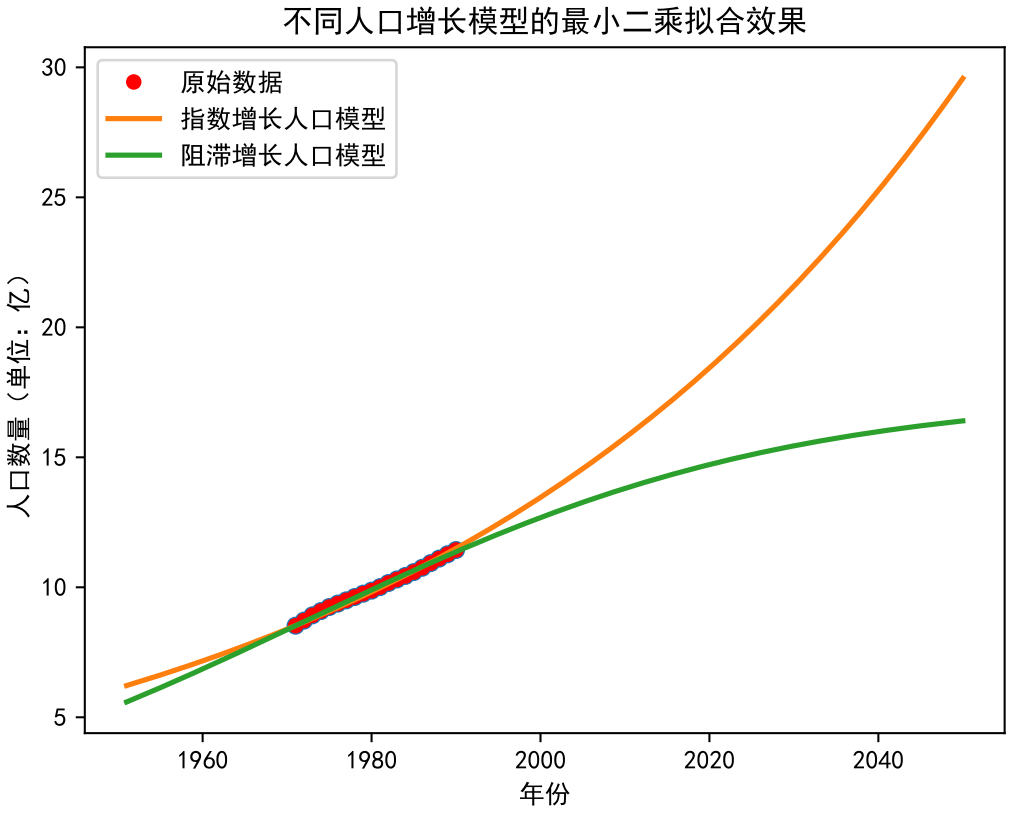


图6: 不同人口增长模型的最小二乘拟合效果比较（curve_fit）