2023 第八届“数维杯”大学生数学建模挑战赛论文

题　目　河流-地下水系统水体污染的研究

摘　要

对河流中的有机污染物的治理始终是环境治理的关键问题，本文分别建立了河流地下水系统的对流-弥散-吸附模型与微生物的有机污染物降解模型，并根据实验数据求得了较为理想的解。

对于问题一，建立了河流-地下水系统的对流-弥散-吸附模型，分别分析了河流的对流、有机物自发弥散以及河流沉积物的吸附作用对有机污染物浓度的影响。对于对流-弥散过程，建立了对流-弥散方程；对于河底沉积物吸附过程，建立了沉积物吸附模型。

对于问题二，本文在问题一所建模型的基础上，求解得到了河流-地下水系统的对流-弥散模型，并对不同的沉积物求解得到了河流-地下水系统的吸附模型。最终基于问题二给出的数据求解得到了四种不同沉积物的有机污染物迁移转化机理。

对于问题三，本文建立了 Monod 模型与 Logistic 微生物生长模型，定量描述了微生物降解有机污染物的过程，并利用 Python 编程，对数据进行可视化处理，定性描述了微生物浓度与有机污染物浓度的关系。

关键词: Langmuir 模型、Monod 方程、对流-弥散-吸附模型、降解动力学模型

目 录

一、 问题重述

1.1 引言

河流对地下水有着直接影响，当河流补给地下水时，河流一旦被污染，将容易导致地下水以及附近水源地收到不同程度的污染，并严重影响工农业正常运作、社会经济发展与饮水安全。在地下水污染治理过程中，最难解决的是有机污染，因此对有机污染物在河流-地下水系统中的行为特征具有十分重要的理论意义与实际价值。在已有的研究中，河流地下水系统中有机污染物的行为特征主要涉及对流迁移、水动力弥散、吸附与阻滞等物理过程、化学过程与生物过程。设地下水渗流场为各向同性均质的稳态流，研究有机污染物的迁移和转化规律，并完成问题。

1.2 要解决的具体问题

1. 问题一：分析建立河流-地下水系统中有机污染物的对流、弥散及吸附作用的数学模型。
2. 问题二：利用给出的试验参数以及数据依据对数学模型研究有机物在河流-地下水系统中的迁移转换机理。
3. 问题三：考虑生物降解作用对有机物污染转化的影响，建立模型，并根据试验数据分析微生物对该有机污染物的降解特性。

二、 问题分析

2.1 问题一的分析

对于问题一，研究有机污染物在河流-地下水系统中的对流、弥散与吸附作用的数学模型，有助于我们了解有机污染物在河流-地下水系统中的迁移和转化规律，为环境保护和污染治理提供科学依据，帮助我们制定合理

的环境保护政策和管理措施。

根据题目要求，从对流、弥散及吸附作用三个方面，分别建立数学模型。对于对流方程，主要考虑在河流水体中有机污染物受水流的影响；对弥散模型，考虑有机污染物自发的弥散过程；对于吸附方程，主要考虑污染物在地下水中与土壤、岩石等固体颗粒表面发生的吸附作用的影响。

2.2 问题二的分析

问题二要求在问题一模型基础上，根据给出的相关数据分析某有机物在河流某段的迁移转化机理。题目给出了对流-弥散参数，因此可以以此建立对流-弥散模型，分析有机污染物的对流-弥散规律。根据题目给出的吸附动力学实验与等温平衡吸附试验的结果，建立吸附模型，研究吸附物的吸附能力。最后，综合上述内容，总结给出有机污染物的迁移转化机理。

2.3 问题三的分析

题目要求分析微生物对有机污染物的讲解特性，可以建立生物降解动力学模型，这些模型通常基于微生物生长和代谢的基本原理。经典模型有 Monod 方程、Haldane 模型、Contois 模型、Logistic 模型等，基于给出数据拟合求解得到方程相关参数。同时可以绘制相关图表，直观的分析微生物降解有机物的降解特性。

三、 模型假设

1. 假设在河流-地下水系统中的有机污染物之间不发生化学反应导致有机物含量降低。
2. 不考虑河道口、河流分流或并流对河流流速的影响。
3. 在河道中沉积物和河水的比例一定

4. 在研究微生物降解过程时，忽略河流对流、弥散等影响因素。

四、 符号说明

表 4-1: 符号与说明

序号	符号	符号说明
1	C	有机污染物浓度
2	u	平均孔隙流速
3	α	质量转移系数
4	D	弥散系数
5	q	吸附量
6	k_d	扩散速率常数
7	K	Langmuir 等温常数

五、 模型的建立与求解

5.1 问题一的模型建立

1. 有机污染物在河流-地下水环境中的状态[1]

有机污染物渗入土壤后，在重力作用或雨水冲刷的作用下向下渗透，通过地下的毛细管带时进入地下水系统。对于密度小于水的有机污染物，会在饱和含水层水面横向迁移；密度大于水的有机物会透过含水层，向下迁移达到不透水层或弱透水层，横向迁移形成重有机物聚集区。随着有机污染物不断渗入地下水介质，会在含水层形成自由相[2]。自由相在迁移转化过程中，随水流发生对流、弥散、吸附等变化。

分别建立对流-弥散模型与吸附模型。

2. 对流-弥散-吸附模型

在河流-地下水系统中，有机污染物的浓度变化遵循质量守恒，根据题设，该河流-地下水系统是一个各向各向同性均质的稳态流，因此，对于扩

散层，可以将其看作时在 x 方向上延伸的一个一维系统，质量守恒方程描述了有机污染物浓度在时间和空间上的变化：

$$\frac{\partial C}{\partial t} = -\frac{1}{n}\nabla J + R \tag{1}$$

其中，C 表示有机污染物的浓度，是空间位置 x 与时间 t 的函数；n 表示多孔介质孔隙度；R 表示有机污染物的产生和消失，在和河流-地下水系统中表示有机污染物被吸附、解附、降解；J 表示这种物质的通量，在一维系统中可以分解为对流通量和弥散通量，下面分别从对流通量和扩散通量分析。

考虑有机污染物在对流中的运输过程，设对流速度为 u，则有机污染物的对流通量可以表示为：

$$J_1 = uC \tag{2}$$

考虑有机污染物在空间上的扩散过程，由 Fick 扩散定律，得到有机污染物在空间上的扩散通量：

$$J_2 = -D\frac{\partial C}{\partial x} \tag{3}$$

其中，D 表示物质的弥散系数。于是得到有机污染物的通量：

$$J = uC - D\frac{\partial C}{\partial x} \tag{4}$$

综上，可以得到对流-弥散-吸附模型：

$$\frac{\partial C}{\partial t} = -u\frac{\partial C}{\partial x} + D\frac{\partial^2 C}{\partial x^2} + R \tag{5}$$

其中，R 表示有机污染物被吸附、解附的过程。在固体表面，有机污染物会被吸附，浓度变化与吸附能力成正比。

3. 吸附模型

考虑有机污染物在土壤或岩石表面的吸附作用。一般而言，地下水中的有机物在各种固体吸附作用是一个复杂的过程，受到有机污染物的浓度、沉积物性质等多方面的影响。利用等温吸附实验可确定在特定的条件下，沉积物对有机物的吸附量，即可预测在不同条件下相同沉积物吸附有机物的含量变化。用途是评估地下水污染物在土壤、沉积物等固态界面上的吸附特性。

建立 Langmuir 等温吸附模型，假设吸附在固体表面的吸附位点是相互独立的。

根据平衡浓度和液相质量计算吸附量：

$$q = \frac{V(C_0 - C_e)}{m} \tag{6}$$

其中 q 为吸附量，描述单位时间内单位吸附剂质量吸附或解吸单位物质量的能力，V 表示吸附达到平衡时的溶液体积，C_0 为初始液相浓度，C_e 为平衡时污染物的液相浓度，m 表示吸附剂的质量。根据模型假设，河道中的沉积物质量与吸附平衡时溶液体积比例恒定，即为一个常数，为方便计算，我们设 $\frac{V}{m} = 1$.

由等温线方程：

$$q = \frac{q_m k C_e}{1 + K C_e} \tag{7}$$

其中，q_m 是最大吸附量，表示单位质量的沉积物对于特定污染物在一档体哦阿健下单最大吸附容量，C_e 表示液相平衡浓度，K 为 Langmuir 常数。对

上式进行线性转化，可得：

$$\frac{1}{q} = \frac{1}{q_m} + \frac{1}{Kq_mC_e} \tag{8}$$

由此，将等温线方程化为线性方程：

$$y = ax + b \tag{9}$$

其中，$y = \frac{1}{q}$,$a = \frac{1}{Kq_m}$,$b = \frac{1}{q_m}$，其中 K 用来描述沉积物对有机污染物的吸附能力。

综上所述，我们得到了河流-地下水系统的对流-弥散-吸附模型：

$$\frac{\partial C}{\partial t} = -u\frac{\partial C}{\partial x} + D\frac{\partial^2 C}{\partial x^2} - KC \tag{10}$$

4. 边界条件

在河流-地下水系统中，边界条件包括：Dirichlet 边界条件与 Neumann 边界条件。

Dirichlet 边界条件表示在河流-地下水系统流入或流出的污染物浓度：

$$C = C_i, (x \in \partial\Omega) \tag{11}$$

其中，C_i 表示边界的污染物浓度。

Neumann 边界条件用来描述模型边界的自然边界条件，假设河流-地下水系统与外界无流通，即：

$$\frac{\partial C}{\partial m} = 0 \tag{12}$$

其中，m 为边界法向量

5. 初始条件

设置初始条件如下：

$$C(x,0) = C_0 \tag{13}$$

表示在 $t = 0$ 时刻，有机污染物初始浓度。

5.2 问题二的分析和求解

5.2.1 模型建立

根据题设，要求在问题一的数学模型基础上分析某有机物的迁移转换机理，建立对流-扩散-吸附模型。

1. 对流-弥散-吸附模型

$$\frac{\partial C}{\partial t} = -u\frac{\partial C}{\partial x} + D\frac{\partial^2 C}{\partial x^2} - KC \tag{14}$$

2. 等温吸附模型

$$q = \frac{q_m k C_e}{1 + KC_e} \tag{15}$$

5.2.2 模型求解

1. 对流-弥散模型的求解

根据题设，地下水渗流场为各向同性均质的稳态流，因此在同一空间位置时间对浓度没有影响，将对流-弥散-吸附方程化简为：

$$0 = -\frac{d}{dx}(qc) + D\frac{d^2 C}{dx^2} - KC \tag{16}$$

即稳态场中的一维扩散方程，是一个二阶常系数线性微分方程，其通解为：

$$C(x) = c_1 e^{r_1 x} + c_2 e^{r_2 x} \tag{17}$$

其中，r_1、r_2 分别为特征方程的根。

利用二阶常系数线性微分方程的特征方程、Dirichlet 边界条件与 Neumann 边界条件，得到方程组：

$$\begin{cases} c_1 + c_2 = C(0,t) \\ c_1 e^{r_1 L} + c_2 e^{r_2 L} = C(L,t) \\ r_1 c_1 e^{r_1 L} + r_2 c_2 e^{r_2 L} = 0 \end{cases} \tag{18}$$

根据题目给出的吸附动力学实验结果对于四种不同的沉积物其边界条件如表5-2：

表 5-2: 四种沉积物的边界条件

沉积物 / 边界条件	S_1	S_2	S_3	S_4
$C(0,t)$	0.495	0.495	0.495	0.495
$C(L,t)$	0.364	0.347	0.173	0.199

分别带入式18, 解得四种不同的沉积物的浓度：

$$S1 : C(x,t) = 0.139e^{-0.131x} + 0.361e^{101.971x} \tag{19}$$

$$S2 : C(x,t) = 0.278e^{-0.154x} + 0.222e^{103.631x} \tag{20}$$

$$S3 : C(x,t) = 0.323e^{0.0088x} + 0.177e^{101.812x} \tag{21}$$

$$S4 : C(x,t) = 0.305e^{0.043x} + 0.195e^{102.983x} \tag{22}$$

2. 等温吸附模型求解

题目给出了四种不同沉积物对 10 种不同初始浓度的某有机污染物 24 小时的等温平衡吸附试验结果，观察表中数据，可以发现，等温条件下，平

衡浓度与初始浓度大致成正比，分别绘制四种沉积物对 10 中不同初始浓度的等温吸附平衡浓度的折线图，如图5-1.

从图5-1可以看出，四种沉积物的等温吸附平衡浓度和初始浓度呈明显的线性关系。

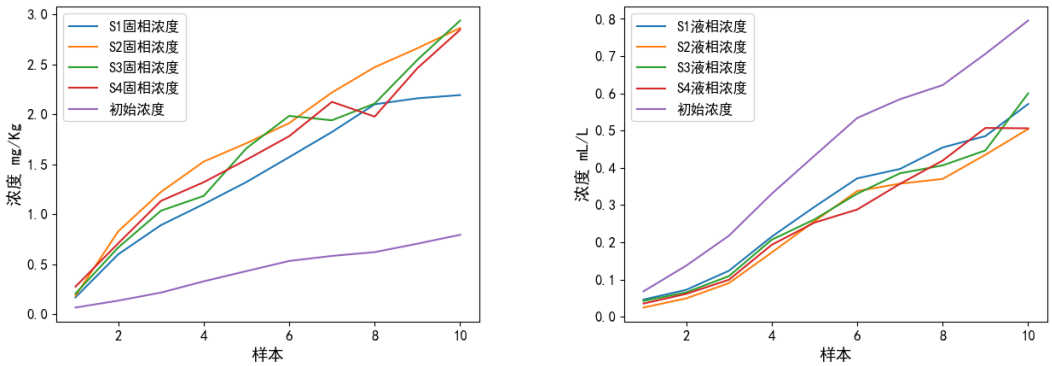
利用最小二乘法对这些曲线进行拟合，计算得到最大吸附量 q_{max} 和等温常数 k_L，结果如表5-3：

表 5-3: 最大吸附量与等温常数

沉积物	最大吸附量 q_{max}	等温常数 K
S_1	4.42	0.13
S_2	0.46	12.57
S_3	2.16	0.34
S_4	0.64	1.67

由此得到了拟合后的等温线，如图5-2。

等物吸附实验的结果反映了吸附过程的一些性质，包括最大吸附量和等温常数。最大吸附量反映了沉积物表面的有效吸附位点数目，等温常数反映了吸附作用受到强度和亲和性。



(a) 固相平衡浓度 (b) 液相平衡浓度

图 5-1: 等温平衡吸附试验

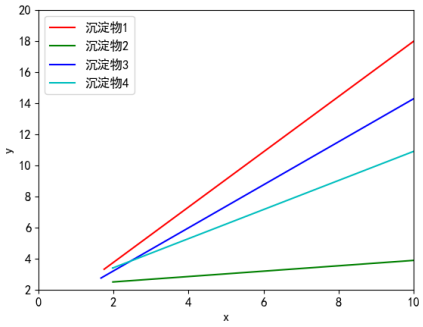


图 5-2: 等温线

利用最大吸附量和等温常数，可以计算得到吸附容量 Ac 和吸附能力指数 Api：

$$\text{Ac} = q_m \times k \tag{23}$$

$$\text{Api} = q_m \frac{kC}{m} \tag{24}$$

计算结果如表5-4

表 5-4: 吸附容量与吸附能力指数

沉积物	最大吸附量	吸附能力指数
S_1	0.56	0.28
S_2	5.79	2.89
S_3	0.72	0.36
S_4	1.07	0.53

由吸附容量和吸附能力指数可以看出，沉积物 S_2 在四种沉积物中吸附能力最强。

5.3 问题三的模型建立与求解

5.3.1 模型建立

考虑微生物对有机污染物的降解特性，通常使用生物降解动力学模型来描述, 其中，更常用的是 Monod 方程。Monod 方程基于 3 个基本假设[3]：

1. 微生物的生长为均衡式生长，因此描述微生物生长的唯一变量是微生物的浓度；
2. 培养基中只有一种基质是生长限制性基质，而其他组分为过量，不影响微生物的生长
3. 细胞的生长视为简单的单一反应，细胞得率为一常数

建立 Monod 方程：

$$\mu = \frac{\mu_m C}{K_s + C} \tag{25}$$

其中，μ 表示微生物比生长速率，描述单位微生物量的增长速率；μ_m 表示最大比生长速率，K_s 表示半饱和常数，是比生长速率为最大生长速率的一半时候的浓度。

绘制出每天的微生物浓度的折线图，如图5-3：

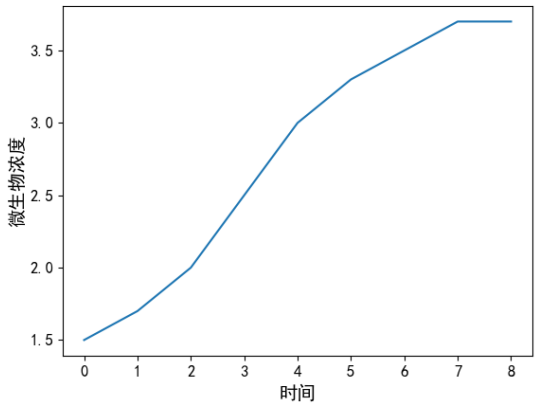


图 5-3: 微生物浓度

从图像可以看出，微生物浓度与时间大致成 Logistic 函数[4]的关系，因此建立 Logistic 生长模型：

$$y = \frac{K}{1 + e^{-rt}} \tag{26}$$

其中，r 表示每个微生物个体单位时间的生长速率，表示为：

$$\mu = rC_m(1 - \frac{C_m}{C_{m\max}}) \tag{27}$$

其中，C_m 表示微生物浓度，$C_{m\max}$ 表示最大微生物浓度。

建立降解动力学方程：

$$\frac{dC_m}{dt} = -kC_m \tag{28}$$

其中 k 表示有机污染物降解速率常数。将式28、式27与式25联立，得到微生物降解模型：

$$\frac{dC_m}{dt} = -kC \cdot \frac{\mu}{K_s + C} \tag{29}$$

5.3.2 模型求解

由图5-3，对微生物浓度与时间关系进行拟合，得到 logistic 曲线，如图5-4：

由式27，求得 μ 的值，如表5-5并将其绘制为折线图，如图5-5。

表 5-5: μ 的值

时间	0	1	2	3	4	5	6	7
μ	0.076	0.075	0.072	0.069	0.063	0.059	0.056	0.055

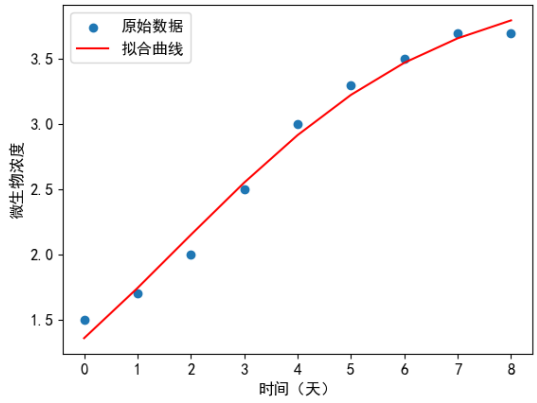


图 5-4: 微生物-时间曲线拟合

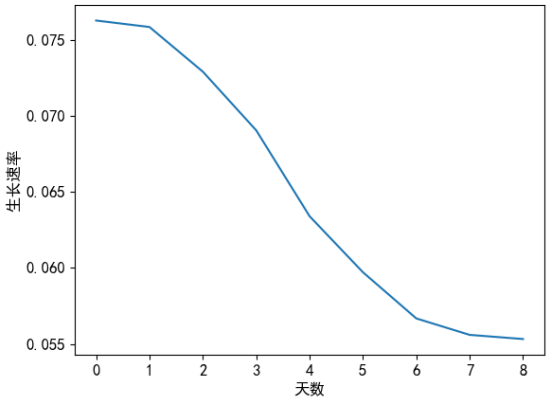


图 5-5: 微生物生长比速率

综合上述结果，可以得到以下结论：

1. 在有机污染物浓度较大时，微生物的增长速率较高，对有机污染物的降解速度缓慢提升。
2. 当降解一段时间后，微生物浓度达到一定数量并趋于稳定，微生物的增长速率放缓，此时微生物已经降解掉一部分有机污染物，每个微生物个体平均分到达有机污染物显著减小，此时微生物对有机物的降解速率减慢
3. 当较高的微生物降解掉大部分有机污染物后，剩余的有机污染物不足以供养数量庞大的微生物种群，此时微生物将会由于养料不足而死亡，

使得微生物浓度降低。

六、 模型的评价与改进

6.1 模型的评价

在问题一与问题二中，建立了河流-地下水系统的对流-弥散-吸附模型，并根据问题二给出的部分数据对这个模型进行求解，并得到了较为满意的解。在问题三中，建立了微生物降解模型并绘制了相关图像，便于直观分析。但是，在建立模型过程中，忽略了有机污染物的排放、河流流速变化、地下水水位变化等影响因素。

6.2 模型的改进

对于问题一和问题二，根据更多的实验数据，如河流流速的变化、地下水水位变化、环境温度等，建立更为完善的对流-弥散-吸附模型。

对于问题三，在分析微生物对有机污染物的影响时，应该同时考虑河流-地下水系统对流、弥散、吸附等七态因素的影响；其次，获取更多的实验数据，能够拟合出更为精确的模型参数，使得模型更为完善。

参考文献

[1] 谷广锋, 曹兴涛, 刘铭辉, 等. 滨海地区石化行业地下水污染物迁移转化研究[J]. 中国石油和化工标准与质量, 2022, 42(93-95).

[2] 冉新民, 李小琴, 殷丽霞. 油污废水中污染物对土壤-地下水环境影响的模拟分析[J/OL]. 兰州大学学报 (自然科学版), 2021, 57(167-175+184). DOI: 10.13885/j.issn.0455-2059.2021.02.004.

[3] 孔维宝, 董妙音, 李万武, 等. 从莫诺方程谈起——生物类本科专业教学中 3 个相似方程的探讨[J/OL]. 微生物学通报, 2015, 42(1599-1602). DOI: 10.13344/j.microbiol.china.140906.

[4] 李华中. Logistic 模型在人口预测中的应用[J]. 江苏石油化工学院学报, 1998.

附 录

程序一

```
#!/usr/bin/env python
# coding: utf-8

# In[1]:


import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import matplotlib as mpl


# In[2]:


df = pd.read_csv('表3.csv')
df


# In[3]:


# 提取数据
x_data = df['序号']

y_data_1 = df['S1液相']
y_data_2 = df['S2液相']
y_data_3 = df['S3液相']
y_data_4 = df['S4液相']
y_data_5 = df['初始浓度']

# 创建画布和子图
fig, ax = plt.subplots()

mpl.rcParams['font.family'] = 'SimHei'
mpl.rcParams['font.size'] = 12
plt.rcParams['font.family'] = 'SimHei'
plt.rcParams['font.size'] = 12

# 绘制折线图
ax.plot(x_data, y_data_1, label='S1液相浓度')
```

```
ax.plot(x_data, y_data_2, label='S2液相浓度')
ax.plot(x_data, y_data_3, label='S3液相浓度')
ax.plot(x_data, y_data_4, label='S4液相浓度')
ax.plot(x_data, y_data_5, label='初始浓度')

# 添加标题和标签
ax.set_xlabel('样本', fontsize=14)
ax.set_ylabel('浓度 mL/L', fontsize=14)

# 添加图例
ax.legend()

# 显示图形
plt.savefig('液相.png')
plt.show()


# In[4]:


# 提取数据
x_data = df['序号']

y_data_1 = df['S1固相']
y_data_2 = df['S2固相']
y_data_3 = df['S3固相']
y_data_4 = df['S4固相']
y_data_5 = df['初始浓度']

# 创建画布和子图
fig, ax = plt.subplots()

mpl.rcParams['font.family'] = 'SimHei'
mpl.rcParams['font.size'] = 12
plt.rcParams['font.family'] = 'SimHei'
plt.rcParams['font.size'] = 12

# 绘制折线图
ax.plot(x_data, y_data_1, label='S1固相浓度')
ax.plot(x_data, y_data_2, label='S2固相浓度')
ax.plot(x_data, y_data_3, label='S3固相浓度')
ax.plot(x_data, y_data_4, label='S4固相浓度')
ax.plot(x_data, y_data_5, label='初始浓度')

# 添加标题和标签
```

```
ax.set_xlabel('样本', fontsize=14)
ax.set_ylabel('浓度 mg/Kg', fontsize=14)

# 添加图例
ax.legend()

# 显示图形
plt.savefig('固相.png')
plt.show()


# In[5]:


# 最小二乘法
def least_squares(x, y):
    n = len(x)
    A = np.vstack([x, np.ones(n)]).T
    a, b = np.linalg.lstsq(A, y, rcond=None)[0]
    return a, b


# In[6]:


# 沉淀物1
x1 = df['x1']
y1 = df['y1']
a1, b1 = least_squares(x1, y1)
q_max1 = -(1 / b1)
k1 = 1 / (a1*q_max1)
q_max1, k1


# In[7]:


# 沉淀物2
x2 = df['x2']
y2 = df['y2']
a2, b2 = least_squares(x2, y2)
q_max2 = -(1 / b2)
k2 = 1 / (a2*q_max2)
q_max2, k2
```

```
# In[8]:


# 沉淀物3
x3 = df['x3']
y3 = df['y3']
a3, b3 = least_squares(x3, y3)
q_max3 = -(1 / b3)
k3 = 1 / (a3*q_max3)
q_max3, k3


# In[9]:


# 沉淀物4
x4 = df['x4']
y4 = df['y4']
a4, b4 = least_squares(x4, y4)
q_max4 = -(1 / b4)
k4 = 1 / (a4*q_max4)
q_max4, k4


# In[10]:




mpl.rcParams['font.family'] = 'SimHei'
mpl.rcParams['font.size'] = 12

# 生成 x 和 y 的数据
x = np.linspace(0, 10, 100)
y1 = a1 * x1 + b1
y2 = a2 * x2 + b2
y3 = a3 * x3 + b3
y4 = a4 * x4 + b4

# 绘制直线
plt.plot(x1, y1, 'r-', label='沉淀物1')
plt.plot(x2, y2, 'g-', label='沉淀物2')
plt.plot(x3, y3, 'b-', label='沉淀物3')
plt.plot(x4, y4, 'c-', label='沉淀物4')
```

```
# 添加图例和坐标轴标签
plt.legend()
plt.xlabel('x')
plt.ylabel('y')

# 指定纵坐标和横坐标范围
plt.xlim(0, 10)
plt.ylim(2, 20)

# 显示图形
plt.savefig('沉淀物.png')
plt.show()


# In[11]:


# 吸附容量 ac
ac1 = q_max1 * k1
ac2 = q_max2 * k2
ac3 = q_max3 * k3
ac4 = q_max4 * k4

# 吸附能力指数 api
api1 = q_max1 * k1 * 0.5
api2 = q_max2 * k2 * 0.5
api3 = q_max3 * k3 * 0.5
api4 = q_max4 * k4 * 0.5

ac1, ac2, ac3, ac4, api1, api2, api3, api4


# In[12]:


# 提取数据
x_data = df['序号']

y_data_1 = df['S1固相']
y_data_2 = df['S2固相']
y_data_3 = df['S3固相']
y_data_4 = df['S4固相']
y_data_5 = df['初始浓度']
```

```
# 创建画布和子图
fig, ax = plt.subplots()

mpl.rcParams['font.family'] = 'SimHei'
mpl.rcParams['font.size'] = 12
plt.rcParams['font.family'] = 'SimHei'
plt.rcParams['font.size'] = 12

# 绘制折线图
ax.plot(x_data, y_data_1, label='S1固相浓度')
ax.plot(x_data, y_data_2, label='S2固相浓度')
ax.plot(x_data, y_data_3, label='S3固相浓度')
ax.plot(x_data, y_data_4, label='S4固相浓度')
ax.plot(x_data, y_data_5, label='初始浓度')

# 添加标题和标签
ax.set_xlabel('样本', fontsize=14)
ax.set_ylabel('浓度 mg/Kg', fontsize=14)

# 添加图例
ax.legend()

# 显示图形
plt.savefig('固相.png')
plt.show()


# In[13]:


0.1702/(0.0461*(4.424046757923152-0.1702))


# In[14]:


0.6005/(0.0722*(4.424046757923152-0.6005))


# In[ ]:
```

程序二

```
#!/usr/bin/env python
# coding: utf-8

# In[1]:


import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import matplotlib as mpl


# In[2]:


df = pd.read_csv('表4.csv')
df


# In[14]:


mpl.rcParams['font.family'] = 'SimHei'
mpl.rcParams['font.size'] = 12

# 提取数据
x_data = df['天数']

y_data_1 = df['浓度']
y_data_2 = df['微生物浓度']
y_data_3 = df['有机物浓度比']

# 创建画布和子图
fig, ax = plt.subplots()

# 绘制折线图
#ax.plot(x_data, y_data_1, label='有机污染物浓度')
ax.plot(x_data, y_data_2)
#ax.plot(x_data, y_data_3, label='有机物污染物浓度比')

# 添加标题和标签
```

```
ax.set_xlabel('时间', fontsize=14)
ax.set_ylabel('微生物浓度', fontsize=14)


# 添加图例


plt.savefig('第三问原始.png')
# 显示图形
plt.show()


# In[4]:


from scipy.optimize import curve_fit
import matplotlib.pyplot as plt
import numpy as np

# 设置中文显示
plt.rcParams['font.family'] = 'SimHei'
plt.rcParams['font.size'] = 12

# 读取数据
data = np.array([[0,0.483,1.5,1],
                 [1,0.479,1.7,0.991718427],
                 [2,0.452,2,0.935817805],
                 [3,0.418,2.5,0.865424431],
                 [4,0.371,3,0.768115942],
                 [5,0.342,3.3,0.708074534],
                 [6,0.319,3.5,0.660455487],
                 [7,0.311,3.7,0.64389234],
                 [8,0.309,3.7,0.639751553]])

# Logistic模型
def logistic(t, K, r, t0):
    y = K / (1 + np.exp(-r*(t-t0)))
    return y

# 设置初始值
def initial_k(data):
    n = len(data)
    i = np.argmax(data[:,1])
    K0 = data[i,2]
```

```
    t0 = data[i,0]
    r0 = 0.1

    print("K0:", K0)
    print("r0:", r0)
    print("t0:", t0)
    return [K0, r0, t0]

# 代入数据
def calculate_mumax(data):
    p0 = initial_k(data)

    # curve fitting
    popt, y_fit = fit_data(data[:, 0], data[:, 2], logistic, p0)

    # Calculating mu_max and Ks
    Ks = popt[0] / 2.0
    Ymax = popt[0]
    mu_max = popt[1] * popt[0] / 4.0
    t_max = (1 / popt[1]) * np.log((popt[0]/2) / (popt[0] - (popt[0]/2)))

    return mu_max, Ks, t_max, popt, y_fit

# Logistic模型
def inverse_logistic(y, K, r, t0):
    t = (1 / r) * np.log(K/y - 1) + t0
    return t

# 最小二乘法拟合
def fit_data(x, y, func, p0):
    # curve fitting
    popt, pcov = curve_fit(func, x, y, p0, maxfev=10000)
    y_fit = func(x, *popt)
    return popt, y_fit

# 计算最大生长速率、半饱和常数和达到最大生长速率所需的时间
mu_max, Ks, t_max, popt, y_fit = calculate_mumax(data)
u_max = mu_max * 3.0 * (1 - (3.0/3.7))
print("------------------------------------------------------------")
print("最大生长速率 :", u_max)
print("半饱和常数 :", Ks)
#rint("达到最大生长速率所需的时间 :", t_max)
print("------------------------------------------------------------")
print("饱和生长浓度K0: ",popt[0])
print("生长速率常数r",popt[1])
```

```
print("时间常数t",popt[2])  # 表示生长速率从半最大生长速率开始增长的时间


# In[5]:


# 绘制拟合曲线

mpl.rcParams['font.family'] = 'SimHei'
mpl.rcParams['font.size'] = 12
plt.rcParams['font.family'] = 'SimHei'
plt.rcParams['font.size'] = 12

plt.scatter(data[:,0], data[:,2], label='原始数据')
plt.plot(data[:,0], logistic(data[:,0], *popt), 'r-', label='拟合曲线')
plt.legend()
plt.xlabel('时间（天）')
plt.ylabel('微生物浓度')
plt.savefig('第三问曲线拟合.png')
plt.show()
print(popt)


# In[6]:


# 获取每一天
days = data[:, 0]
# 获取每一天的微生物数
microbes = data[:, 3]

# 计算每一天的微生物增长速率（每天的生长速率）
daily_growth_rates = u_max * microbes / (Ks + microbes)

# 打印每一天的微生物增长速率
for day, daily_growth_rate in zip(days, daily_growth_rates):
    print(f"天数 {day}: 生长速率 {daily_growth_rate}")


# In[7]:


# 绘制时间-生长速率曲线
plt.rcParams['font.family'] = ['SimHei', 'Microsoft YaHei']
plt.plot(days, daily_growth_rates)
```

```
plt.xlabel("天数")
plt.ylabel("生长速率")
plt.savefig('生长速率曲线.png')
plt.show()


# In[8]:


# 计算dC/dt(降解速率)
# 输入数据
time_points = [0, 1, 2, 3, 4, 5, 6, 7, 8]
concentrations = [0.483, 0.479, 0.452, 0.418, 0.371, 0.342, 0.319, 0.311, 0.309]

# 计算dC/dt 用的差分法只能求出前八天的
degradation_rates = [(concentrations[i+1] - concentrations[i]) / (time_points[i+1] - tim

# 打印结果
for i in range(len(degradation_rates)):
    print('第{}天的降解速率为：{:.4f}'.format(i, degradation_rates[i]))


# In[9]:


# 绘制降解速率-时间曲线
time_point = [0, 1, 2, 3, 4, 5, 6, 7]
xdata = np.array(time_point)
ydata = np.array(concentrations)
plt.rcParams['font.family'] = ['SimHei', 'Microsoft YaHei']
#plt.plot(xdata, ydata, 'o', label='data')
plt.plot(xdata, degradation_rates)
plt.xlabel("天数")
plt.ylabel("降解速率")
plt.savefig('降解速率-时间.png')
plt.show()


# In[10]:


Ks = 2.056  # 常数
data = [[0, 0.483, 1.5, 1, 0.166669549, -0.004],
        [1, 0.479, 1.7, 0.991718427, 0.120977669, -0.027],
        [2, 0.452, 2, 0.935817805, 0.022477575, -0.034],
```

```
        [3, 0.418, 2.5, 0.865424431, -0.221588644, -0.047],
        [4, 0.371, 3, 0.768115942, -0.565529108, -0.029],
        [5, 0.342, 3.3, 0.708074534, -0.819833024, -0.023],
        [6, 0.319, 3.5, 0.660455487, -1.009343817, -0.008],
        [7, 0.311, 3.7, 0.64389234, -1.214834489, -0.002]]

t = [row[0] for row in data]
C = [row[1] for row in data]
mu = [row[4] for row in data]
dCdt = [row[5] for row in data]

k = [-dCdt[i] * (Ks + C[i]) / (C[i] * mu[i]) for i in range(len(data))]

# 打印结果
for i in range(len(k)):
    print('第{}天的降解速率常数为：{:.4f}'.format(i, k[i]))


# In[11]:


# 绘制降解速率常数-时间曲线
xdata = t
ydata = k

plt.rcParams['font.family'] = ['SimHei', 'Microsoft YaHei']
#plt.plot(xdata, ydata, 'o', label='data')
plt.plot(t, k)
plt.xlabel("天数")
plt.ylabel("降解速率常数")
plt.savefig('降解速率常数-时间.png')
plt.show()


# In[ ]:
```