基于微分方程的泳池水质安全问题研究

摘　　要：

游泳池水质的安全对游泳者的健康至关重要,杭州电子科技大学游泳馆投入使用近两年，通过适时加氯保持水质在安全范围内，确保在开学季更好地服务师生。针对这类问题，本文通过分析数据确定了泳池人数与余氯浓度之间的函数关系，根据一级动力学模型建立了关于余氯浓度的微分方程模型，对相关问题进行了求解，并为人数控制提供了优化方案，让更多人能够体验杭电游泳馆的舒适。

针对问题一，首先基于一级动力学反应模型建立关于**氯浓度随时间变化**的**微分方程模型**，其次，利用题目所给数据求出余氯下降系数***k*=0.4621**，并画出**余氯浓度随时间变化曲线**。最后对晚上十点后**余氯浓度**随时间变化情况进行求解，解得晚上十点后首次加氯的时刻为第二天凌晨**3点22分39秒。**

针对问题二，首先考虑泳池人数对氯浓度变化速率的影响建立基于泳池人数的氯浓度变化的**微分方程模型**，然后根据附件一的数据使用**最小二乘法**模型参数进行拟合，得到了余氯下降系数关于泳池人数的**三次函数**，从而算出常温下泳池人数为255人时的余氯下降系数为**1.4603**；最后解得首次**加氯时间**为**0.5000h**。

针对问题三，在问题二建立的余氯浓度随时间变化的模型基础上，设计了相关程序求解9点到21点加氯方案，解得本题**加氯次数**为**15次**，并画出当天**余氯浓度随时间变化曲线**。

针对问题四，首先从欧洲中期天气预报中心的**EAR5数据库**获取**杭州2010-2024年8-9月每小时的气温数据**（每小时有9条记录且无缺失值），并对数据进行了**热力值转摄氏度**等预处理操作，其次基于**SARIMA模型**对数据进行分析，进而预测了9月8日至10日的**杭电游泳馆水池温度**，然后基于**余氯浓度变化**与温度的关系建立**微分方程模型**，最后通过问题三所设计的程序求出9月9日9点至21点的加氯方案，求出**加氯次数为10次（包含3次人工加氯）**，并画出**余氯浓度随时间变化曲线**。

针对问题五，考虑到池水温度和人数对余氯浓度的影响，首先在问题二、三的模型基础上构建了基于**水池温度**与**泳池人数**的余氯浓度随时间变化的**微分方程模型**，其次建立了在考虑**泳池水质安全**的情况下**最大化入馆总人数**的**整数规划模型**，最后得到了最优人数控制方案：开放时间内入馆人数均不超过**520人**，此方案下游泳馆的**加氯次数为3次。**

关键词：微分方程模型，最小二乘法，SARIMA模型，整数规划模型

目录

一、 问题重述

1.1 问题背景

为了确保游泳者的身体健康，游泳馆除了设施完整、配备齐全之外，泳池的水质安全格外重要，务必对池水定期检测和消毒。当前常用的依然是氯消毒方法，使用一种以三氯异氰尿酸（化学式C3N3O3Cl3，俗称强氯精，简称TCCA）为主要成分的消毒剂，专用加氯设备在需要时通过多个加药泵同时注入泳池，由于药水浓度较大但体积小，故不需要将泳池的水排出即可完成“加氯”消毒工作。

根据相关标准规定，游泳池的余氯标准一般为0.3-0.6mg/L。这个范围是经过科学研究和实践经验总结得出的，既能够保证游泳池水的清洁卫生，又不会对人体健康造成危害。余氯低于0.3mg/L时，可能导致水质不清洁，容易滋生细菌和藻类，对游泳者的健康造成威胁；而余氯超过0.6mg/L时，可能对皮肤和黏膜产生刺激，引起不适感。因此，泳池水质安全控制问题成为管理者关注焦点。

杭州电子科技大学游泳馆投入使用近两年，为了在开学季能更好地服务师生，泳池通过适时加氯使水质长期保持在安全范围内尤为重要。

1.2 问题提出

基于上述研究背景，题目提供了2个附件，分别包含了常温下游泳人数和0.5小时后余氯值的对应关系、不同时段在池游泳人数均值统计等信息，本文需要根据上述附件解决以下5个问题：

问题一：计算初始余氯浓度0.6mg/L降至最低安全浓度0.05mg/L所需的时间，确定池水自晚上十点后需要再次加氯的时刻。需要根据所给数据建立相应的微分方程模型来模拟氯浓度随时间的变化情况，从而求解氯浓度降至0.05mg/L的时间。

问题二：分析游泳人数对余氯浓度的影响，确定255人游泳时余氯浓度从0.6mg/L降至0.3mg/L的时间，确定首次加氯时间。需要根据附件一给出的数据拟合出泳池人数与0.5小时后余氯浓度的关系函数，然后得到最终的一级动力学模型，并求出首次加氯时间。

问题三：已知杭电游泳馆营业时间为一天四场：上午 9:00-11:00，下午 12:00-14:00 和 15:00-17:00，晚上 18:00-21:00。根据泳池开放时段和附件给出的平均人数数据，计算9:00至21:00期间的加氯次数，并绘制余氯浓度变化曲线。根据问题二的步骤求出各个时间段的k值，通过设计程序求出余氯浓度随时间变化数据。

问题四：考虑到馆内气温影响到池水温度，并对池中余氯值有着直接影响，通过预测9月8日至10日的气温和池水温度变化，可以分析水温对余氯浓度的影响，首先收集气温数据通过数学模型进行分析，从而预测9月8日至10日的杭电游泳馆气温和馆内水池温度，绘制相关温度曲线并填入表中；然后确定9月9日上午9点到晚上21点余氯浓度随时间变化数据，从而确定当天泳池的加氯次数，并绘制该时段内余氯浓度变化曲线。

问题五：综合考虑水温和人数对余氯的影响，结合上述分析优化9月9日四个开放时段的入场人数，确保余氯浓度始终在0.3-0.6mg/L范围内。本题在综合考虑前面问题的基础上对余氯浓度变化模型进行了更新，然后建立了关于最大化泳池人数的整数规划模型，结合两个模型对该问题进行了求解。

二、 模型假设及符号说明

2.1 模型假设

1、假设题目所给出的原始数据是真实有效的；
2、假设每次加氯时间足够短，可以忽略不计；
3、假设每个时间段的人数始终保持不变；

2.2 符号说明

表2- 符号说明

符号	说明
C_t	反应t时刻的余氯浓度
C_0	余氯衰减初始浓度
t	反应时间
k	余氯的一级衰减系数
$g(n)$	余氯浓度变化率和人数的函数关系
ρ	皮尔逊相关系数
AIC	赤池信息准则
BIC	贝叶斯信息准则
ACF	自相关函数
PACF	偏自相关函数
T_w	泳池内水温
T_m	杭州室外气温
$T[i]$	每小时泳池内水温
n_i	每小时泳池内人数
$T(t)$	第t分钟泳池水温
x_t	每分钟泳池人数

三、 问题一模型建立与求解

3.1 问题分析

统计检测发现池水余氯浓度从0.6mg/L降至0.3mg/L平均用时1小时30分钟，在水温保持常温状态下计算初始余氯浓度0.6mg/L降至最低安全浓度0.05mg/L所需的时间，确定池水自晚上十点加氯后需要再次加氯的时刻。根据题目要求需要建立氯浓度随时间变化的微分方程模型，对池水余氯浓度进行拟合，从而求出最终结果。其求解流程如下：

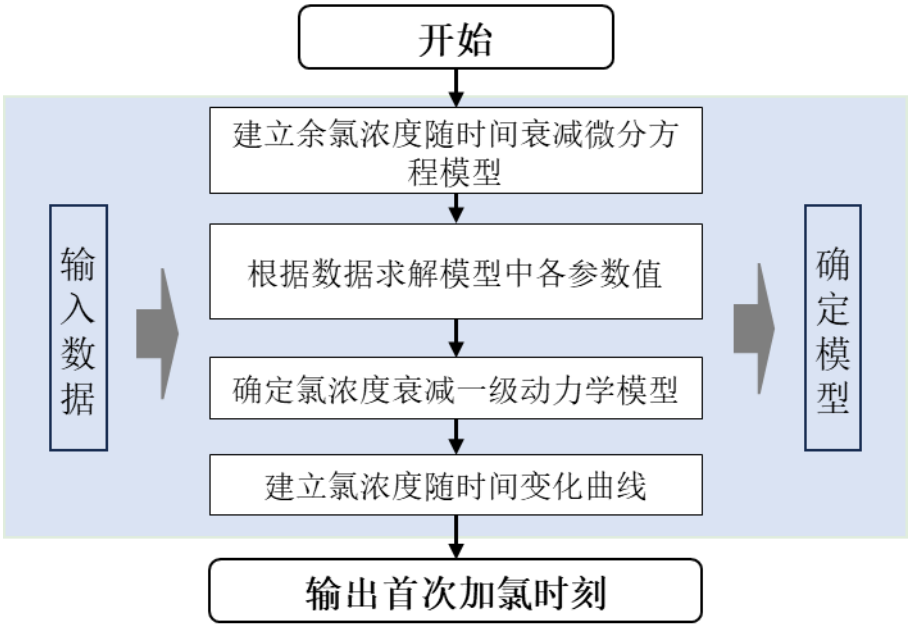

图1 问题一求解流程图

3.2 氯浓度衰减随时间变化的微分方程模型

水中氯的衰减速率可以表示为如下微分方程：

$$\begin{cases}\dfrac{dC_t}{dt} = -kC_t \\ C_t|_{t=0} = C_0\end{cases} \tag{3-2}$$

积分可得氯浓度衰减随时间变化的微分方程模型：

$$C_t = C_0 * e^{-kt} \tag{3-3}$$

式中，C_t表示反应t时刻的余氯浓度，单位为mg/L；C_0表示余氯衰减初始浓度，单位为mg/L；t表示反应时间，单位为h；k表示余氯的一级衰减系数，单位为h^{-1}。

根据题目所提供数据可以得到在C_0=0.6mg/L的前提下，当C_t为0.3mg/L时，时刻t为1.5，可得方程组：

$$\begin{cases}C_t = C_0 * e^{-k*t} \\ C_0 = 0.6 \\ C_t = 0.3 \\ t = 1.5\end{cases} \tag{3-4}$$

根据式3-4的方程组计算得到k=0.4621，并得到最终的一级动力学模型：

$$C_t = 0.6 * e^{-0.4621*t} \quad (3-5)$$

公式3-5为目前应用广泛的一级动力学模型，是一种化学物质衰减反应的速度与该物质的浓度的一次方成正比的反应过程。通过这一模型对氯浓度衰减随时间变化的情况进行了拟合[1]，拟合所得曲线见下图：

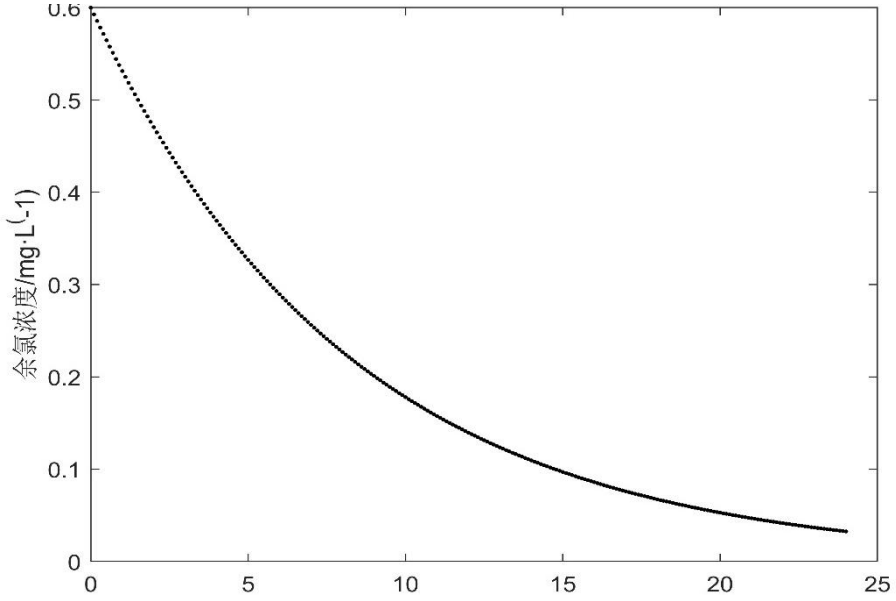

图2 余氯浓度随时间变化曲线

将C_t=0.05mg/L带入模型中可以得到此时的t=5.3774,即再次加氯的间隔时间为5小时22.644分钟，根据题目所给的初次加氯时刻为晚上10点，可得第一次加氯的时刻约为第二天凌晨3点22分39秒。

四、 问题二模型建立与求解

4.1 问题分析

问题二要求在泳池常温情况下根据附件一数据进行建模分析游泳人数对余氯浓度的影响，确定255人游泳时余氯浓度从0.6mg/L降至0.3mg/L的时间，从而确定首次加氯时间。首先要建立基于泳池人数变化的氯浓度随时间变化的微分方程模型，然后根据附件一给出的数据对模型参数进行拟合，最后根据拟合数据建立最最终的微分方程模型，并得到首次加氯时间的最终结果。其求解流程如下：

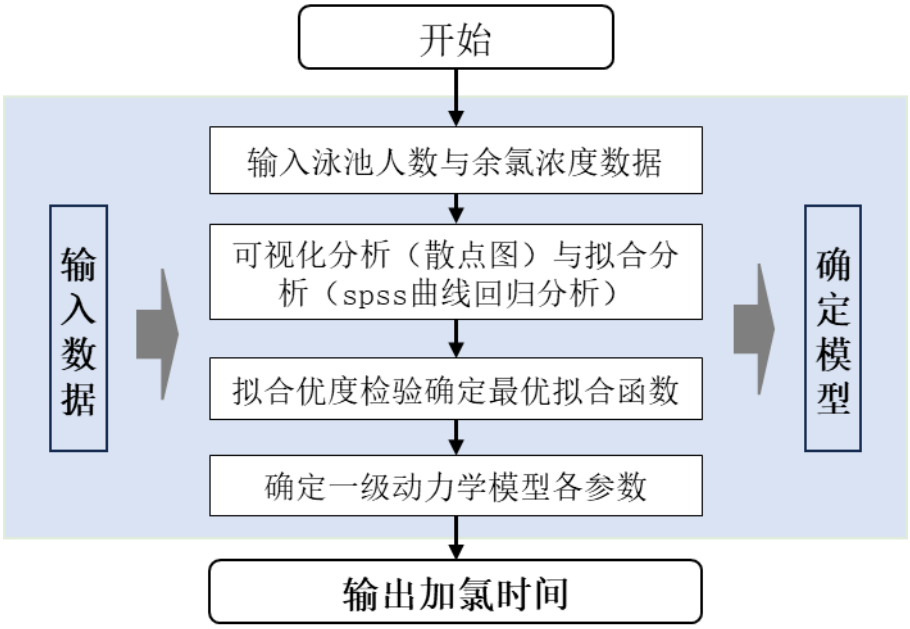

图3 问题二求解流程图

4.2 模型建立

假定余氯浓度变化率和人数之间存在的函数关系为$g(n)$，可得微分方程：

$$\begin{cases}\dfrac{dC_t}{dt} = -kC_t - g(n)C_t \\ C_t|_{t=0} = C_0\end{cases} \tag{4-1}$$

求解方程4-1得到基于泳池人数的氯浓度衰随时间变化的一级动力学模型模型：

$$C_t = C_0 * e^{-(k+g(n))*t} \tag{4-2}$$

移项得到如下表达式：

$$g(n) + k = \frac{\ln\dfrac{C_t}{C_0}}{-t} \tag{4-3}$$

根据附件一数据绘制出如下泳池人数与0.5小时后 $g(n)+k$关系的散点图：

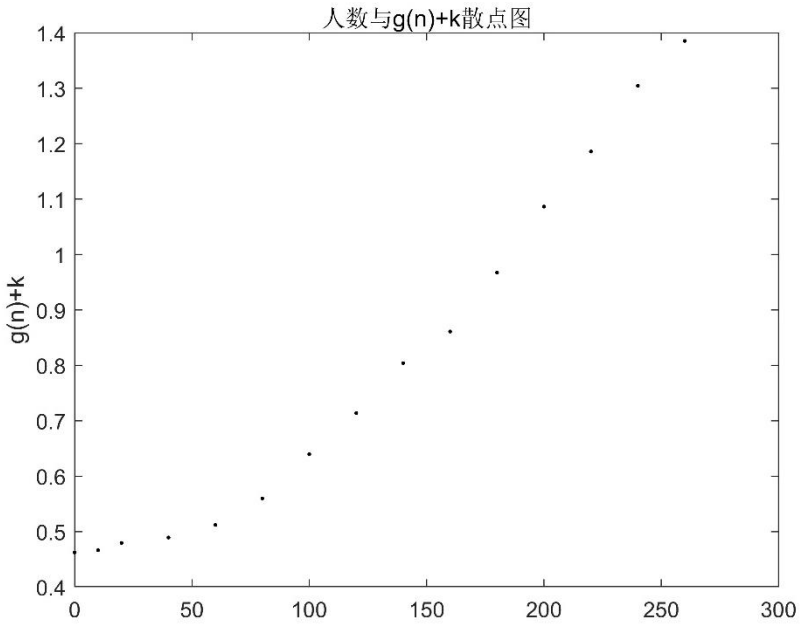

图4 人数与0.5小时后$g(n)+k$关系散点图

通过分析散点图可以推测人数与0.5小时后$g(n)+k$之间存在一定的函数关系，由此通过spss软件对该数据进行曲线回归分析（一次函数、二次函数、三次函数、复合函数、增长函数、指数函数、逻辑斯蒂函数等），其拟合曲线如图4所示：

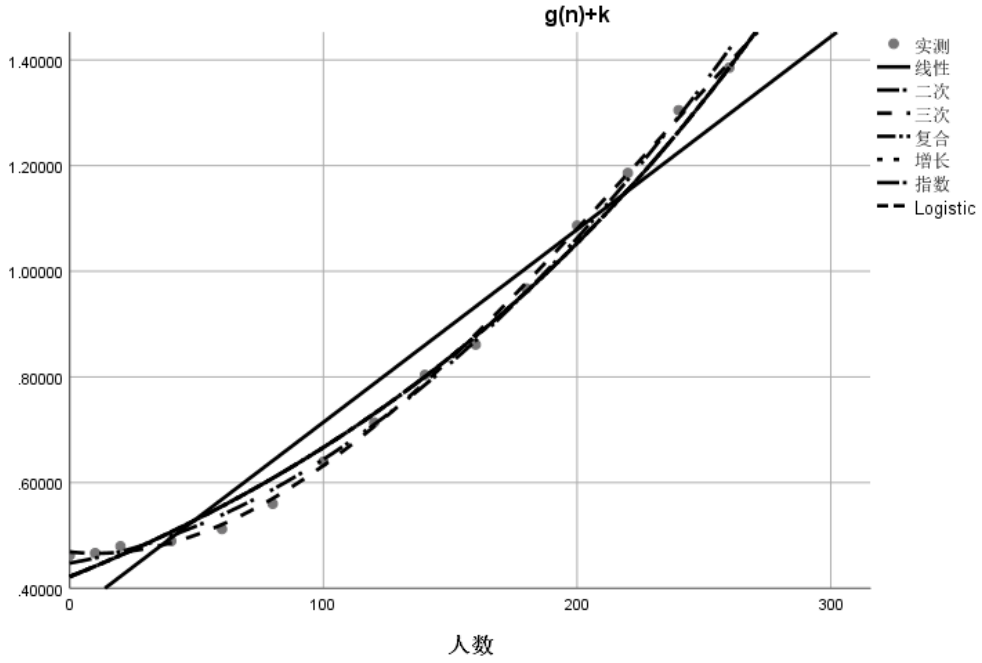

图5 0.5小时后余氯浓度拟合曲线

观察图像可以发现三次函数对散点的拟合结果

对各个函数的拟合优度进行检验，其结果如下表所示：

表1 各函数拟合优度检验

方程	模型摘要					参数估算值			
	R 方	F	自由度1	自由度2	显著性	常量	b1	b2	b3
线性	0.953	264.352	1	13	0.000	0.349	0.004		

二次	0.997	1908.312	2	12	0.000	0.448	0.001	1.111E-05	
三次	0.999	3509.818	3	11	0.000	0.469	-0.001	2.510E-05	-3.624E-08
复合	0.985	870.315	1	13	0.000	0.422	1.005		
增长	0.985	870.315	1	13	0.000	-0.863	0.005		
指数	0.985	870.315	1	13	0.000	0.422	0.005		
Logistic	0.985	870.315	1	13	0.000	2.370	0.995		

根据表1发现三次函数的具有更优的拟合度，其R方值为0.999，由此得到人数与$k+g(n)$之间的三次回归函数方程：

$$g(n)+k=0.469-0.001n+2.510E-05*n^2-3.624E-08*n^3 \qquad (4-4)$$

当泳池人数为255人时，根据公式4-4计算得到$g(255)+k=1.4603$。最终得到问题二游泳人数为255人时余氯浓度随时间变化的一级动力学模型：

$$C_t=0.6*e^{-1.4603*t} \qquad (4-5)$$

根据模型计算得到余氯浓度首次降为0.3mg/L的时间t=0.5h，即泳池中255人时泳池的首次加氯时间为0.5h以后。

五、 问题三模型建立与求解

5.1 问题分析

已知杭电游泳馆营业时间为一天四场：上午 9:00-11:00，下午 12:00- 14:00 和 15:00-17:00，晚上 18:00-21:00。问题三要求根据泳池开放时段和附件给出的平均人数数据，计算9:00至21:00期间的加氯次数，并绘制余氯浓度变化曲线。已知每个时间段的平均泳池人数，根据问题二的求解过程可以最终求出各个时间段的k值，由此确定每个时间段的一级动力学模型，并计算出最终的浓度变化曲线。其求解流程如下：

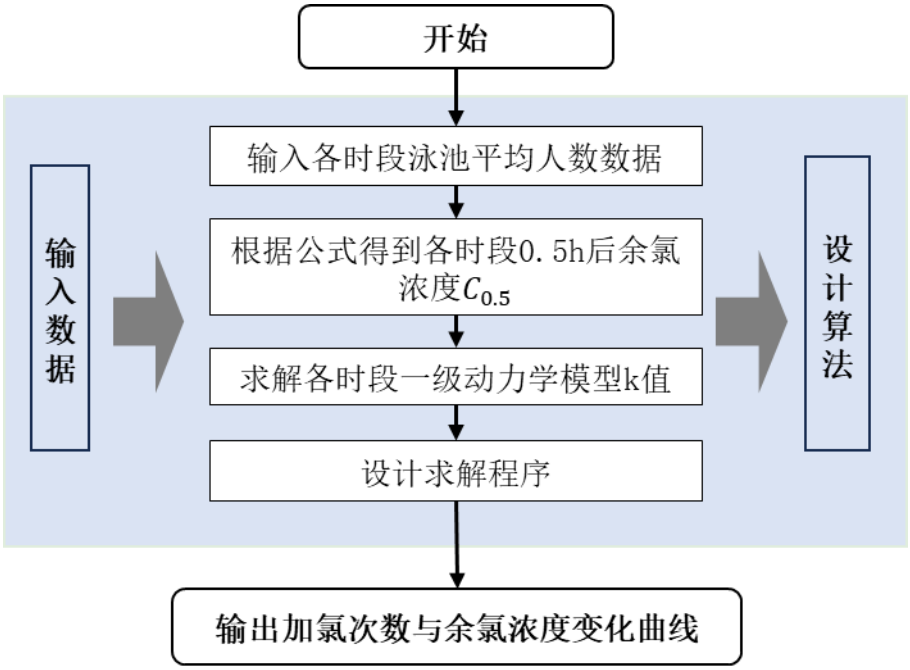

图6 问题三求解流程图

5.2 模型建立

根据问题二求解，可以将每个时间段的泳池平均人数带入到公式4-1在初始氯浓度C_0为0.6mg/L的前提下泳池人数与0.5小时后余氯浓度之间的三次回归函数方程，得到每个时间段0.5h后的余氯浓度$C_{0.5}$,将每个时间段的$C_{0.5}$带入公式3-3中可以求解得到各时间段一级动力学模型的k值。计算结果如下表所示：

表2 各时间段在池平均人数与k值

时间段	在池平均人数	$C_0 = 0.6$时$C_{0.5}$值	K
9:00-10:00	20	0.47462844	0.468794778
10:00-11:00	80	0.45107856	0.57057628
11:00-12:00	0	0.475	0.467229702
12:00-13:00	150	0.39571675	0.832461955
13:00-14:00	180	0.36794716	0.977980629
14:00-15:00	0	0.475	0.467229702

15:00-16:00	210	0.34026253	1.134424377
16:00-17:00	340	0.26368252	1.644367703
17:00-18:00	0	0.475	0.467229702
18:00-19:00	280	0.28575976	1.483556395
19:00-20:00	420	0.29486284	1.420838712
20:00-21:00	190	0.35862367	1.029312179

根据各时间段的一级动力学模型的k值以及9点初始为0.6mg/L的氯浓度可以逐步算出各个时间段的一级动力学模型。

5.3 程序设计

由于时间段较多，步骤复杂，因此设计了以下循环算法来对问题三进行求解，其算法流程图如下：

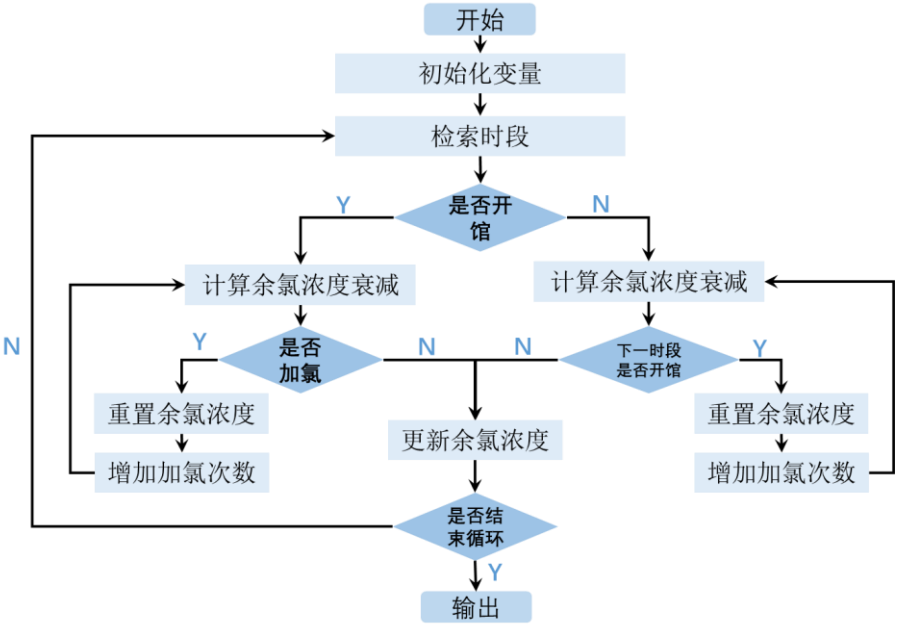

图7 循环算法流程图

1. 初始化变量：设定初始状态下余氯浓度（0.6 mg/L）、加氯次数、时间等参数。
2. 循环判断：当时间小于12小时时，循环持续进行。
3. 是否开馆判断：根据泳池人数是否为零判断是否开馆，若不为零则表示开馆。
4. 计算余氯浓度衰减：计算余氯浓度从当前值下降到0.3 mg/L所需的时间。
5. 判断是否加氯：若余氯浓度在当前时段下降到0.3 mg/L则需重置余氯浓度为0.6 mg/L，并增加加氯次数。
6. 更新余氯浓度：若余氯浓度在该时间段剩余时间未低于0.3 mg/L，则更新余氯浓度并增加时间继续循环。
7. 闭馆处理：更新当前闭馆时间段余氯浓度，若下一时间段开馆，则重置余氯浓度为0.6 mg/L，并增加加氯次数，否则转入下一时间段循环。
8. 输出结果：当循环结束时，输出加氯次数与余氯浓度变化曲线。

通过该程序，最终求得9点到21点这段时间内加氯次数为15次，具体的加氯时刻表如下：

加氯次数	加氯时刻
1	10:23:35'
2	12:00:00'
3	12:49:57'
4	13:33:58'
5	15:00:00'
6	15:36:39'
7	16:09:11'
8	16:34:28'
9	16:59:46'
10	18:00:00'
11	18:28:01'
12	18:56:03'
13	19:25:09'
14	19:54:26'
15	20:32:43'

图8 加氯时刻表

这一时段泳池中的余氯浓度值变化曲线如图所示：

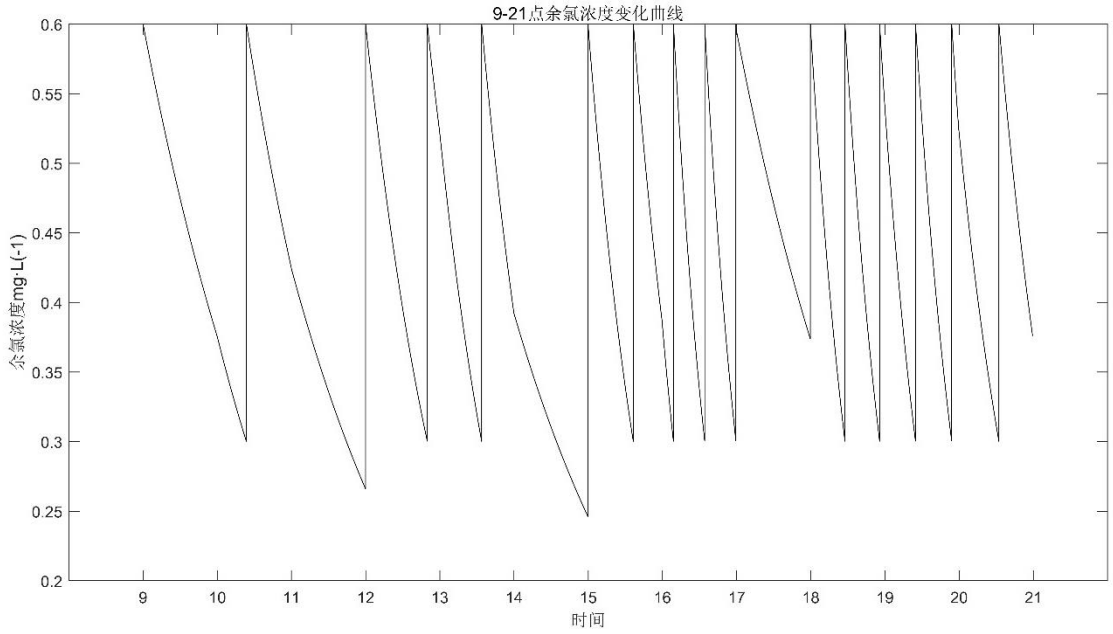

图9 余氯浓度变化曲线

六、 问题四模型建立与求解

6.1 问题分析

问题四要求通过对杭电开学季的气温预测，分析杭电游泳馆池水余氯值受当天水温变化的影响，从而给出加氯时间节点以保障水质安全。首先根据收集的气温数据，预测出杭电开学报到的前三天（即2024年9月8日-10日）杭电校园气温和馆内气温，进一步给出游泳馆的池水温度变化规律，并绘出72小时的杭电游泳馆池水温度的变化曲线。然后在不考虑泳池人数影响的情况下，充分考虑水温变化对池水余氯 影响，建模求解给出2024年9月9日上午9点到晚上21点的加氯次数（忽略每次加氯耗时），总假定早上9点池中余氯浓度0.6mg/L，并求出该时段泳池中余氯浓度值变化曲线。

由于在问题四中涉及到了温度变化对泳池余氯浓度变化的影响，因此需要重新建立相应的数学模型来进行求解。该问题的求解流程如下：

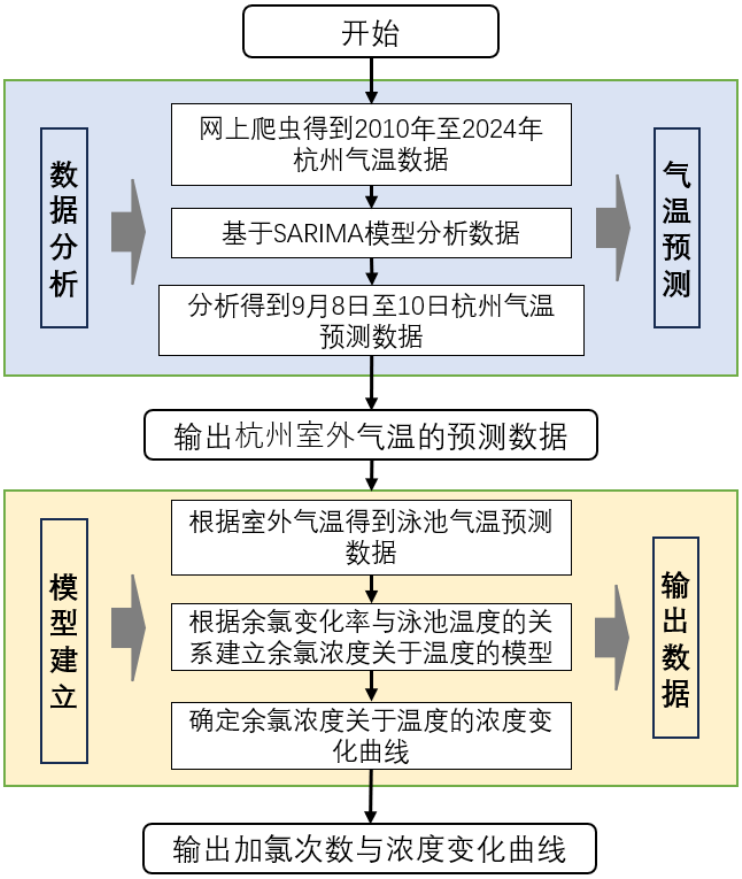

图10 问题四求解流程

6.2 数据获取与预处理

为预测杭州未来某一时段的气温变化情况，首先需要获取杭州历史气温数据来进行分析。数据的获取主要包括数据提取、数据处理两部分。

本文所选用的数据集来自EAR5（ECMWF Reanalysis v5）。EAR5是由欧洲中期天气预报中心（ECMWF）发布的全球气象再分析数据集。再分析数据是通过将历史观测数据与数值天气预报模型结合来生成的，这种方法能为全球的天气和气候研究提供详细且一致的历史记录。

为选取所需数据，需要确定地理位置与时间范围，通过检索得到杭州的地理坐标大约为：30.2741°N，120.1551°E。所截取的时间范围为1979年至2023年。确定好两个重要参数后通过python的cdsapi库下载EAR5中全部符合要求的数据。

EAR5数据涵盖了全球范围内的气象参数，包括温度、湿度、风速等。为获取有用数据（气温数据），本文使用python中的netCDF4库将原数据集转化为CSV格式，并提取所需的杭州室外温度和时间数据。

最后需要对提取数据进行相应处理从而获得标准化后的简化数据。由于本文所提取数据中每小时记录了9个温度数据，现取每小时9个温度数据的平均值作为各小时的气温数据。将温度单位由开尔文转换为摄氏度，同时时区由UTC（世界标准时间）转化为中国时区（东八区）。

6.3 温度预测

为根据杭州历史温度预测未来某一时间段的杭州气温变化情况，本文采用SARIMA模型来进行预测。SARIMA模型即季节性自回归积分滑动平均模型，是在ARIMA模型的基础上额外考虑了季节性因素。

6.3.1 SARIMA模型

首先介绍ARIMA模型，ARIMA全称为自回归差分移动平均模型，其主要由三部分构成，分别为自回归模型（AR）、差分过程（I）和移动平均模型（MA）。SARIMA模型在ARIMA模型的基础上增加了4个季节性参数，分别是3个超参数（P,D,Q）和季节性周期参数s。

具体SARIMA模型如下：

$$y_t = \phi_1 x_{t-1} + \phi_2 x_{t-2} + \ldots + \phi_p x_{t-p} + \Phi_1 x_{t-s} + \Phi_2 x_{t-2s} + \ldots \Phi_P x_{t-P*s} + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2} + \ldots + \theta_q \epsilon_{t-q} + \Theta_1 \epsilon_{t-s} + \Theta_2 \epsilon_{t-2s} + \ldots + \Theta_Q \epsilon_{t-Q*s} \quad (6-8)$$

6.3.2 参数确定

为确定最优模型，首先需要确定差分阶数d，首先观察时间序列图：

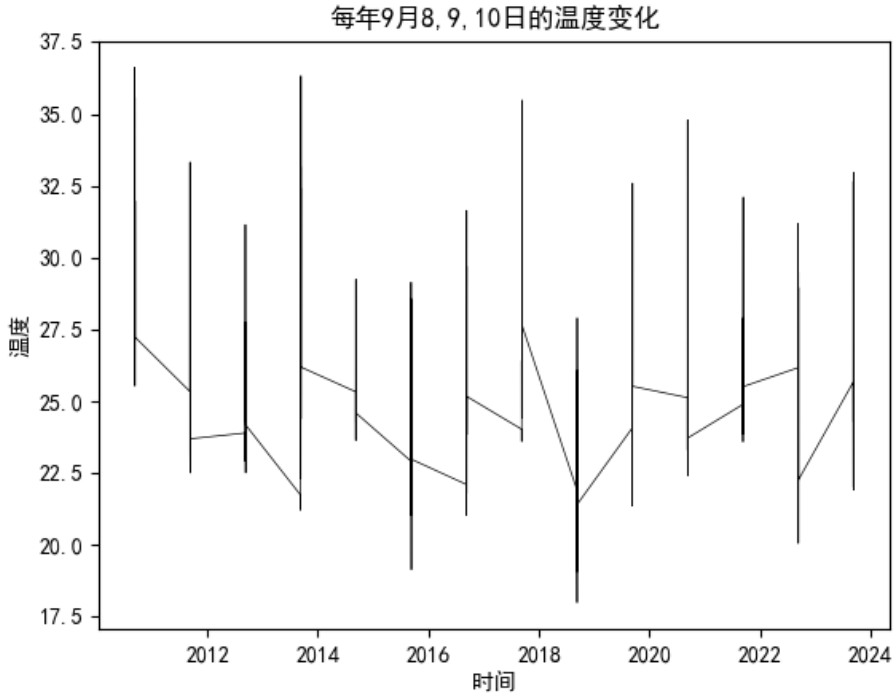

图11时间序列图

观察图像易得气温数据存在非平稳性，由此进行一阶差分得到相对平稳的数据。由于气温数据存在明显的周期性（即24h），因此设定s=24，并对一阶差分后数据继续进行一阶二十四步季节性查分，差分后时间序列图如下：

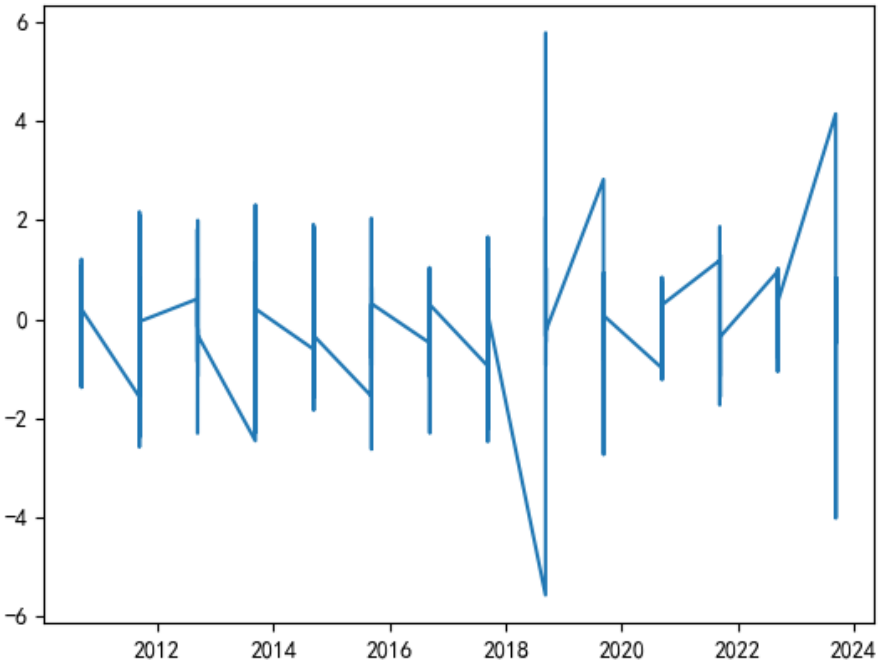

图12 一阶二十四步季节性差分后时间序列图

观察图像可知经季节性差分后时间序列明显趋于平缓，然后需要确定AR项阶数p、MA项阶数q，我们通过ACF图和PACF图初步确定p、q的候选范围：

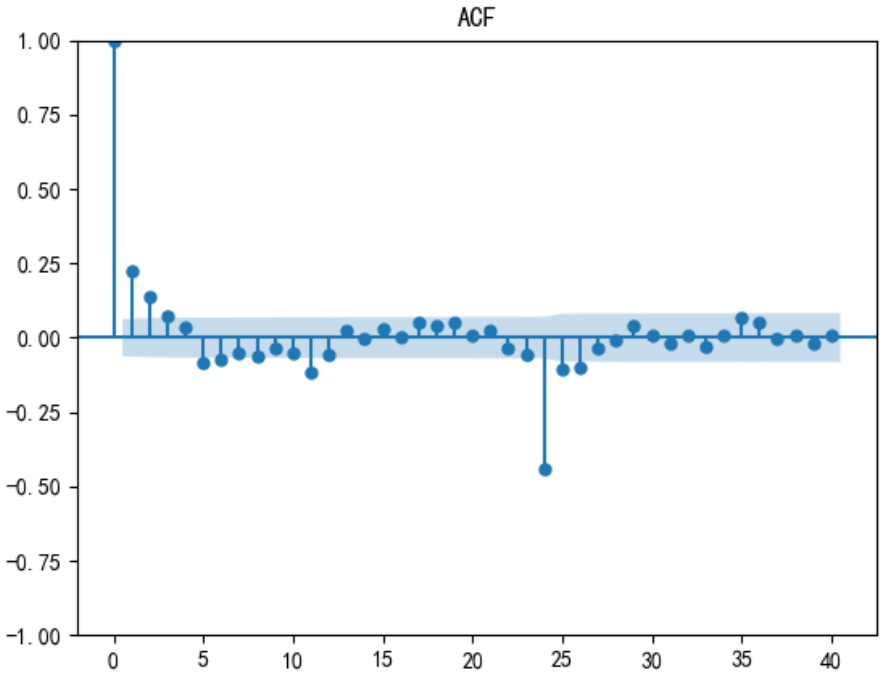

图13ACF图

通过观察ACF图可以确定p的取值范围为[0,2]。

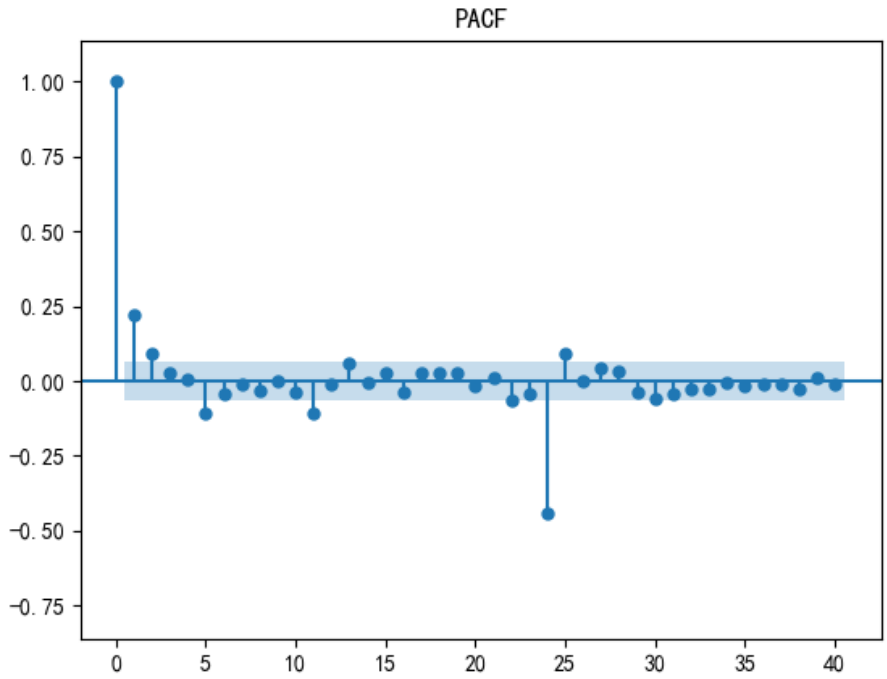

图14PACF图

通过PACF图可以确定q的取值范围为[0,2]。

在初步确定好p、q值后，采用AIC和BIC准则对模型参数进一步优化，即尝试不同p，q组合，从中选取使AIC和BIC最小的参数组合作为最终参数，可以得到当$p=1, q=2$时AIC和BIC取值最小，该模型评估指标如下：

表3 模型评估指标值

评估指标	值
AIC	1679.596
BIC	1708.785
MSE	0.3501302748025323
RMSE	0.5917180703701149
MAE	0.3838432008716365

最终求得模型中各参数取值如下表：

表4 SARIMA模型参数取值

参数	取值
p	1
d	1
q	2
P	0
D	1
Q	2
s	24

最终得到的SARIMA模型为SARIMA*(1,1,2)*(0,1,2,24)。

6.3.3 数据分析

本文共采用了三个方案分别选用了三个不同时间段的数据集分别进行分析与预

测，分别为方案1（2010年至2024年9月8日至9月10日）、方案2（2010年至2024年8，9月份杭州气温数据）、方案3（2000年至2024年9月8日至9月10日杭州气温数据）。

基于该SARIMA模型，我们分别对三个预测方案的不同数据集进行了分析并得到了三个预测结果，其预测方案最终结果对比图如下：

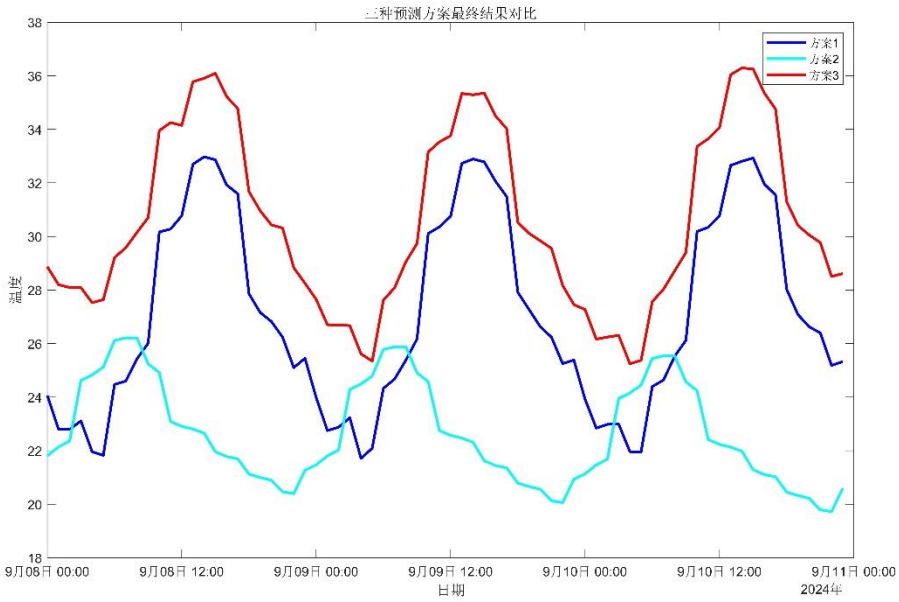

图15预测方案最终结果对比图

通过观察图像，我们最终认定方案1的预测结果更符合今年预期的温度情况，并根据方案1的数据集得到了经SARIMA模型预测后的杭州游泳馆2024年9月8日至9月10日的室外气温数据。

已知池中水温在夏秋季节一般比馆内气温低1.5~2.5℃，在馆内气温很高时（35℃及以上）池水温度会低3.5~4℃，而杭电 游泳馆在开学季内不开空调调温，且馆内气温比室外气温平均低了3℃。由此建立泳池水温关于杭州室外温度的数学模型，当室外温度不低于38℃时，泳池中的水温平均比室外温度低5℃，当室外温度低于38℃时，泳池中的水温平均比室外温度低6.75℃，其表达式如下：

$$T_w = \begin{cases} T_m - 5, & T_m < 38 \\ T_m - 6.75, & T_m \geq 38 \end{cases} \qquad (6-17)$$

其中T_w表示泳池内水温，T_m表示杭州室外气温。

根据所得数据以及公式6-17最终得到杭州游泳馆2024年9月8日至9月10日的池水温度预测值如下表：

表5 1 杭电游泳馆三天相应的池水温度预测值

	2024年9月8日温度	2024年9月9日温度	2024年9月10日温度
最高温度	27.9745℃	27.8923℃	27.9288℃
最低温度	21.0112℃	21.1693℃	21.1217℃
上午10:00	25.1682℃	25.1115℃	25.1856℃
中午12:00	25.7728℃	25.7528℃	25.7792℃
下午14:00	27.9745℃	27.8923℃	27.8070℃
下午16:00	26.9259℃	27.0647℃	26.9560℃

晚上20:00	21.8227℃	21.6295℃	21.6259℃

6.4 模型建立

基于前文所得到的泳池温度数据，由此可以得到9月9日上午9点到晚上21点各个时间段在泳池温度影响下余氯浓度的变化情况的数学模型，下面介绍其具体建模流程。

已知泳池中的温度和余氯变化率之间存在正比例关系：

$$k = \alpha * 10^{\frac{T(t)-25}{5}} \tag{6-18}$$

其中α为未知常数。

根据公式6-17所求得的泳池温度以及公式6-18中余氯变化率与泳池温度之间的正比例关系建立余氯浓度关于温度的微分方程：

$$\begin{cases} \frac{dC_t}{dt} = -C_t * \alpha * 10^{\frac{T(t)-25}{5}} \\ C_t|_{t=0} = C_0 \end{cases} \tag{6-19}$$

将方程6-19改写为分离变量形式：

$$\frac{dC_t}{C_t} = -\alpha * 10^{\frac{T(t)-25}{5}} dt \tag{6-20}$$

对式6-20两边积分得到：

$$\ln|C_t| = -\alpha * \int 10^{\frac{T(t)-25}{5}} dt = \ln|C_0| \tag{6-21}$$

使用龙格库塔法求解后可得常数α=0.4621，由此可以确定余氯浓度关于温度的最终数学模型：

$$C_t = C_0 * e^{-0.4621 * \int 10^{\frac{T(t)-25}{5}} * \mathrm{d}t} \tag{6-22}$$

将9月9日上午9点到晚上21点各个时间段的泳池温度带入公式6-23中可以得到这一时段的余氯浓度随时间变化的情况，再根据问题三的循环算法模型可以得到最终的余氯浓度变化数据，并得到9月9日上午9点到晚上21点杭电泳池加氯次数为10次，其具体时间段如下：

表6 加氯时刻表

加氯次数	加氯时刻
1	'12:00:00'
2	'13:01:27'
3	'13:27:03'
4	'13:52:38'
5	'15:00:00'
6	'15:24:58'
7	'15:49:56'
8	'16:20:46'
9	'16:55:33'
10	'18:00:00'

所得杭电游泳馆9月9日余氯浓度变化曲线如下：

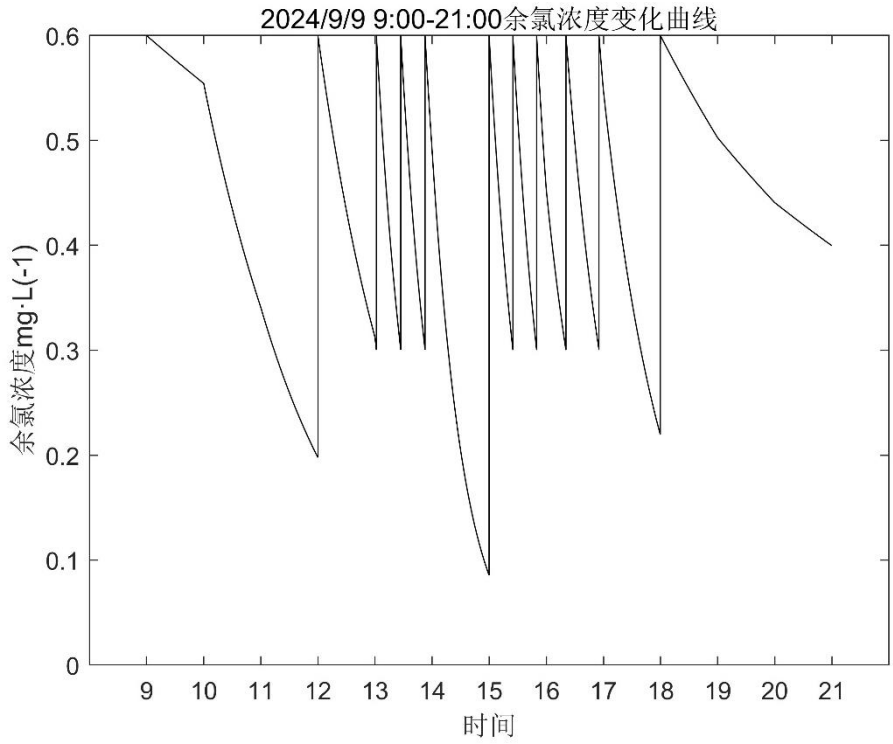

图16杭电游泳馆9月9日余氯浓度变化曲线图

七、 问题五模型建立与求解

7.1 问题分析

问题五要求综合考虑馆内池水温度变化以及泳池人数变动对余氯值的影响，为保证泳池水质始终安全（一直控制在0.3-0.6mg/L 范围内），结合前面的分析并通过合适建模方法，给出2024年9月9日四个开放时段内的入场人数的控制优化方案，尽可能让更多人体验杭电游泳馆。

7.2 模型建立

7.2.1 余氯浓度变化模型

为同时考虑水池温度与泳池人数对余氯浓度变化的影响，需要建立相应的数学模型来进行求解。综合上述四个问题的求解思路可得以下余氯浓度基于水池温度和泳池人数随时间变化的微分方程：

$$\begin{cases}\dfrac{dC_t}{dt} = -\alpha * 10^{\frac{T_w-25}{5}} C_t - g(n)C_t \\ C_t|_{t=0} = C_0\end{cases} \tag{7-1}$$

积分得到余氯浓度基于水池温度和泳池人数的一级动力学模型：

$$C_t = C_0 * e^{-(\alpha * 10^{\frac{T_w-25}{5}} + \mathrm{g(n)}) * t} \tag{7-2}$$

其中:

$\alpha = 0.4621,$

$g(n) = 0.0069 - 0.001n + 2.510E-05 * n^2 - 3.624E-08 * n^3$

得到确定参数的余氯浓度变化的一级动力学模型：

$$C_t = C_0 * e^{-(0.4621 * 10^{\frac{T_w-25}{5}} + 0.0069 - 0.001n + 2.510E-05 * n^2 - 3.624E-08 * n^3) * t} \tag{7-3}$$

7.2.2 泳池人数优化模型

为确定各时间段入场人数控制的最优方案，首先假定加氯时刻都在每分钟的开始时刻，然后将9-21点12个小时按每分钟离散化为720分钟，通过对每分钟初时刻进行浓度检测操作，建立如下关于泳池人数的整数规划数学模型：

1.已知数据

设定各小时温度为$T[i], i \in \{1,2 \dots 12\}$，各小时温度数据如下：

时间	9-10	10-11	11-12	12-13	13-14	14-15	15-16	16-17	17-18	18-19	19-20	20-21
温度	**21.1693**	**25.1115**	**25.3574**	**25.7528**	**27.7305**	**27.8923**	**27.7838**	**27.0647**	**26.4891**	**22.9131**	**22.2714**	**21.6295**

设定时刻$t = 1,2 \dots 719$

设定开馆时段$open[j], j \in \{1,2,3,4\}$;闭馆时段$close[k], k \in \{1,2,3\}$：

$$open[1] \in \{1 \dots 119\}$$
$$close[1] \in \{120,122 \dots 179\}$$
$$open[2] \in \{180,182 \dots 299\}$$
$$close[2] \in \{300-359\} \quad (7-4)$$
$$open[3] \in \{360-479\}$$
$$close[3] \in \{480-539\}$$
$$open[4] \in \{540-719\}$$

2.决策变量

（1）各小时馆内人数为n_i：

$$n_i \in N; 1 \le i \le 12 \quad (7-5)$$

其中已知第3，6，9小时闭馆$n_3 = n_6 = n_9 = 0$

（2）各分钟人数x_t：

$$x_t \in N; 1 < t \le 719 \quad (7-6)$$

（3）各分钟初是否加氯：

$$y_t \in \{0,1\}; 1 < t \le 719 \quad (7-7)$$

3.目标函数为最大化当日泳池入馆总人数：

$$max \sum_{i=1}^{12} n_i \quad (7-8)$$

4.约束条件

（1）各分钟余氯浓度：

$$C(t) = (1-y_t)f(c_{t-1}, x_t) + y_t * 0.6; 1 < t \le 719 \quad (7-9)$$

其中$f(c_{t-1}, x_t)$表示根据第$t-1$分钟初余氯浓度计算第t分钟初余氯浓度，表达式如下:

$$f(c_{t-1}, x_t) = C_{t-1} * e^{-(0.4621*10^{\frac{T(t)-25}{5}}+0.0069-0.001n+(2.510E-05)n^2-(3.624E-08)n^3)*\frac{1}{60}} \quad (7-10)$$

其中$T(t) = T[ceil(\frac{t}{60})]$,为第t分钟的泳池水温。

初始浓度$C(0) = 0.6$

（2）浓度范围约束：

开馆期间余氯浓度要求在0.3-0.6ml/L内：

$$0.3 \le C(t) \le 0.6; t \in open \quad (7-11)$$

闭关期间余氯浓度要求大于0.05ml/L：

$$0.05 \le C(t); t \in close \quad (7-12)$$

（3）加氯间隔时间约束：

要求每次加氯间隔时间不低于10分钟：

$$\sum_{l=t}^{t+10} y_l \le 1; 1 \le t \le 709 \quad (7-13)$$

（4）开馆前氯浓度约束：

要求每次开关前余氯浓度需要增加至0.6ml/L：

$$y_t = 1; t \in \{180,360,540\} \quad (7-14)$$

（5）闭关后加氯约束：

每次闭馆期间均不加氯：

$$y_t = 0; t \in close \tag{7-15}$$

（6）泳池人数约束：

各小时内泳池人数恒定：

$$x_t = n[i]; (i-1) * 60 \le t < i * 60 \tag{7-16}$$

游泳馆泳池人数不超过520人：

$$n[i] \le 520; i\epsilon\{1,2.120\} \tag{7-17}$$

5.最终得到的整数规划模型如下：

$$max \sum_{i=1}^{12} n_i$$

$$\begin{cases} C(t) = (1-y_t)f(c_{t-1}, x_t) + y_t * 0.6; 1 < t \le 719 \\ C(1) = 0.6 \\ 0.3 \le C(t) \le 0.6; t \in open \\ 0.05 \le C(t); t \in close \\ \sum_{l=t}^{t+10} y_l \le 1; 1 \le t \le 709 \\ y_t = 1; t \in \{180,360,540\} \\ y_t = 0; t \in close \\ x_t = n[i]; (i-1) * 60 \le t < i * 60 \\ n[i] \le 520; i\epsilon\{1,2.120\} \\ n_i \in N; 1 \le i \le 12 \\ x_t \in N; 1 < t \le 719 \\ y_t \in \{0,1\}; 1 < t \le 719 \end{cases} \tag{7-18}$$

7.3 问题求解

该模型可通过动态规划算法和智能算法结合进行求解，但是对于本题数据，在各个开放时间段内，计算不同人数情况下氯浓度从0.6mg/L开始十分钟后的余氯浓度，结果如图所示：

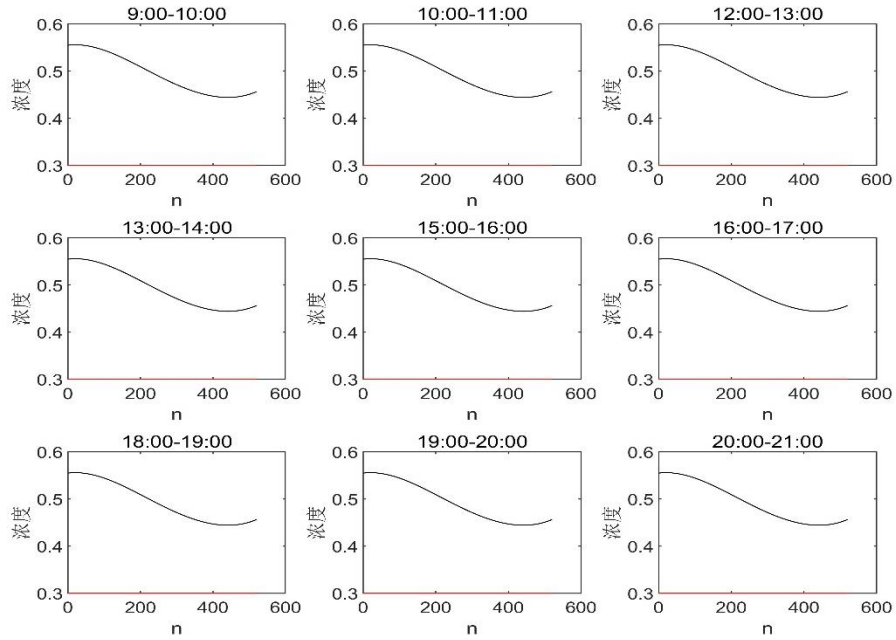

图17各小时十分钟后氯浓度变化曲线图

通过观察图像可以发现，即使各时间段泳池人数均达到上限520人，氯浓度在加氯后十分钟内也不会低于0.3mg/L，即不会出现两次加氯间隔小于10min的情况，因此最优人数方案为各开放时间内入馆人数均不超过520人。

八、 模型评价

8.1 模型的优点

（1）微分方程模型：比较直观，可以准确地描述动态系统的演化过程,可以预测和分析系统的行为，能反应事物之间的内在关系和发展变化的普遍规律。

（2）SARIMA模型：能够处理复杂的时间序列数据,包括季节性和趋势性。同时,它也能够处理序列的噪声和缺失值。

（3）整数规划模型：能够处理实际问题中的离散决策变量，通过引入整数变量，模型可以精确地描述某些约束条件。

8.2 模型的缺点

（1）微分方程模型：求解过程可能比较困难，需要使用数值方法或近似方法进行求解，实际情况并不完全满足假设条件时，中长期预测容易产生较大的误差；其次微分方程解得存在性和唯一性证明困难，且不易求出。

（2）SARIMA模型：预测结果可能受到某些模型参数选择的影响，对于一些非线性时间序列数据，它的预测结果可能不太理想。

8.3 模型的推广与改进

（1）动态优化模型：引入动态优化或实时调整机制，使模型能够根据实际情况和即时数据进行调整，从而进一步提高效率和适应性。

（2）多目标优化：在考虑最大化泳池人数的同时，可以引入其他优化目标，如最小化加氯次数、温度调节等，通过多目标优化模型，综合考虑多个因素，提高模型的实际应用价值。

（3）改进的启发式算法：可以研究和应用改进的启发式算法（如遗传算法、模拟退火算法等），在求解数据量更大的问题时，能够在较短时间内找到接近最优解的解决方案。

参考文献

[1] 许润.集雨窖水的水质保持及净化技术研究[D].华中科技大学,2019.DOI:10.27157/d.cnki.ghzku.2019.003190.

[2]. Chowdhury, S., K. Alhooshani and T. Karanfil, Disinfection byproducts in swimming pool: Occurrences, implications and future needs. Water Research, 2014. 53: p. 68-109.

[3]. Yang, L., et al., An insight of disinfection by-product (DBP) formation by alternative disinfectants for swimming pool disinfection under tropical conditions. Water Research, 2016. 101: p. 535-546.

[4] CSDN, 时间序列模型(四)：ARIMA模型, https://blog.csdn.net/Lvbaby_/article/details/130971374, 2024.8.8

[5] 季节性时间序列预测之SARIMA模型, https://blog.csdn.net/qq_48521968/article/details/133742331，2024.8.9

附录

代码1

介绍：用matlab对第一题进行求解

```
clc;clear;

c_0 = 0.6;
c_t = 0.3;
t = 1.5;
k = - (log(c_t/c_0))/t;



% 计算余氯为0.05时间
c_t2 = 0.05;
t2 = - (log(c_t2/c_0))/k;

%画出随时间变化图
T = [0:0.1:24];
y = func1(c_0,k,T);
figure(1)
plot(T,y,".k");
xlabel("时间/h");ylabel("余氯浓度/mg·L^(-1)");
title("25°余氯浓度随时间变化曲线")

function y = func1(c_0,k,t)
    y = c_0 * exp(-k*t);
end
```

代码2

介绍：用python对第二题进行求解

```
clc;clear;



data = readtable('附件1 常温下游泳人数与0.5小时后余氯值.xlsx');
data = table2array(data);
ns = data(:,1); cts = data(:,2);

c0 = 0.6;
t = 0.5;
gn_k = -log(cts./c0)./t;
```

```
figure;
plot(ns,gn_k,'.k','LineWidth', 2);
ylabel('人数');ylabel('g(n)+k');
title('人数与g(n)+k散点图');

data_new = [ns gn_k];
csvwrite('第二题处理数据.csv',data_new);
clc;clear;

x = 255; %  人数

c_0 = 0.6;
c_t = func2(x);
t = 0.5;
k = - (log(c_t/c_0))/t;



%  计算余氯为0.05时间
c_t2 = 0.3;
t2 = - (log(c_t2/c_0))/k;



function y = func2(x)
%
    b = [0.469      -.001      2.510E-5 -3.624E-8];

    y = b(1) + b(2) .* x + b(3) .* x.^2 + b(4) .* x.^3;
end

function y = func1(c_0,k,t)
    y = c_0 * exp(-k*t);
```

代码3

介绍：用python对问题三进行求解

```
clc;clear;

%  人数
ns = [20      80      0     150     180      0     210     340      0     280     420     190];

%  估计浓度
cs = func2(ns);

%  估计k
```

```
ks = func3(cs);

t0 = 9; %初始时间
c_0 = 0.6; %  初始浓度
add_times = 0;%加氯次数
add_t = [];

delta_t = 0;


%  时间步长0.01
T = [0];
C = [0.6];


%  第几个小时
ind = 1;
while ind <= 21-9

    if ns(ind) ~= 0
        %这个小时有人

        %  计算降到0.3的时间
        t = func4(c_0,0.3,ks(ind));
        if ind == ceil(delta_t + t)
            %当没到下一个小时时
            T = [T (T(end)+[0:0.001:t])];
            c = func1(c_0,ks(ind),[0:0.001:t]);
            C = [C c];

            delta_t = delta_t + t;
            c_0 = 0.6;add_times = add_times+1; add_t = [add_t delta_t];%加氯

        else
            T = [T (T(end)+[0:0.001:ind-delta_t])];
            c = func1(c_0,ks(ind),[0:0.001:ind-delta_t]);
            C = [C c];

            %  计算这个小时末浓度
            c_0 = func1(c_0,ks(ind),ind-delta_t);
            delta_t = ind;
            ind = ind + 1;
```

```
            % %若下一小时有人，加满氯
            % if ind <= 12 && ns(ind-1)==0 && ns(ind)~=0
            %         c_0 = 0.6;add_times = add_times +1;
            % end
        end
    else
        %没人
        T = [T (T(end)+[0:0.001:ind-delta_t])];
        c = func1(c_0,ks(ind),[0:0.001:ind-delta_t]);
        C = [C c];

        % 计算这个小时末浓度
        c_0 = func1(c_0,ks(ind),ind-delta_t);
        delta_t = ind;
        ind = ind + 1;

        %若下一小时有人，加满氯
        if ind <= 12 && ns(ind-1)==0 && ns(ind)~=0
            c_0 = 0.6;add_times = add_times +1;add_t = [add_t delta_t];
        end
    end
end

add_t = add_t +9;
datetime_str = datestr(add_t./24, 'HH:MM:SS');


figure(1)
plot(9+T,C,'k');hold on;
xlabel("时间");ylabel("余氯浓度mg・L(-1)");
title("9:00-21:00余氯浓度变化曲线");
xlim = [9:21];xticks(9:1:21);




function y = func1(c_0,k,t)
%根据初始浓度和时间计算结束浓度
    y = c_0 * exp(-k*t);

end
```

```
function y = func4(c_0,c_t,k)
%  计算降到0.3耗时
    y = - (log(c_t/c_0))/k;

end

function k = func3(c_t)
%  根据0.5小时后浓度估计k
    c_0 = 0.6;

    t = 0.5;
    k = - (log(c_t./c_0))./t;

end

function y = func2(x)
%  根据人数求半小时后浓度
    b = .475;
    a1 = 9.207E-5;
    a2 = -5.747E-6;
    a3 = 1.073E-8;
    y = b + a1 .* x + a2 .* x.^2 + a3 .* x.^3;
end
```

代码4

介绍：用python对问题四进行求解

```
数据获取与预处理
import cdsapi
import pandas as pd

#  创建一个CDS API客户端
c = cdsapi.Client()

#  请求数据
c.retrieve(
    'reanalysis-era5-single-levels',
    {
        'product_type': 'reanalysis',
        'variable': '2m_temperature',
        # 'year': [str(year) for year in range(2019, 2025)],
```

```
            'year': [str(year) for year in range(2000, 2025)],
            # 'year': '2024',
            # 'month': ['01', '02','03','04','05','06','07','08', '09','10','11','12'],
            'month': ['09'],
            # 'month' : ['7','8','9'],
            # 'day': [str(day).zfill(2) for day in range(1, 32)],
            'day': ['7','8','9','10'],
            'time': [f'{hour:02d}:00' for hour in range(24)],
            'area': [30.38, 119.92, 29.86, 120.45],  # 杭州区域（北，西，南，东）
            'format': 'netcdf',
        },
        # 'hangzhou_18_24_full.nc'
        'hangzhou_00_23_9_7890.nc'
)

# 读取NetCDF文件并转换为Pandas DataFrame
import netCDF4 as nc
import pandas as pd
from datetime import datetime, timedelta
import time

# 指定.nc文件的路径
nc_file_path = 'hangzhou_00_23_9_7890.nc'

# 打开NetCDF文件
dataset = nc.Dataset(nc_file_path, mode='r')

# 检查文件中的变量名
print("Variables in the dataset:")
for var in dataset.variables:
        print(var)

temperature = dataset.variables['t2m'][:] - 273.15   # 't2m' 是2米温度的标准名称
# 提取日期
time = dataset.variables['time'][:]

time_var = dataset.variables['time']
# 转换时间为可读格式
time_units = time_var.units
time_data = nc.num2date(time_var[:], units=time_units)

#  时间转化为北京时间
```

```
time = time_data + timedelta(hours=8)

lat = dataset.variables['latitude'][:]
lon = dataset.variables['longitude'][:]

# 将数据组合成一个DataFrame
data = {
    'time': time,
    'latitude': lat,
    'longitude': lon,
    'temperature': temperature.flatten()
}

# 如果有多个维度需要处理，可能需要使用网格来匹配每个纬度和经度点
# 例如，你可以使用以下方式处理时间、纬度、经度和温度：
import numpy as np

time_grid, lat_grid, lon_grid = np.meshgrid(time, lat, lon, indexing='ij')

# 然后，创建一个 DataFrame
df = pd.DataFrame({
    'time': time_grid.flatten(),
    'latitude': lat_grid.flatten(),
    'longitude': lon_grid.flatten(),
    'temperature': temperature.flatten()
})


# 保存为CSV文件
csv_file_path = 'hangzhou_00_23_9_7890.csv'
df.to_csv(csv_file_path, index=False)

print(f"Data successfully converted to CSV and saved to {csv_file_path}")


# 数据处理
# 导入数据
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler
from statistics import mean
```

```
import datetime

# 读取数据
data = pd.read_csv('hangzhou_00_23_9_7890.csv')
# 查看数据
print(data.head())

time = data['time']
temperature = data['temperature']

databyhour = []
dataok = []
value = []
for i in range(len(time)-1):
    if i+1 == len(time) or time[i] == time[i+1]:
        value.append(temperature[i])
    else:
        dataok.append([time[i], mean(value)])
        ind = datetime.datetime.strptime(time[i], "%Y-%m-%d %H:%M:%S")
        databyhour.append([ind.year, ind.month, ind.day, ind.hour, mean(value)])

        value = []

# 去掉日期不在9月8，9，10日的数据
temp = []
for i in range(len(dataok)):
    ind = datetime.datetime.strptime(dataok[i][0], "%Y-%m-%d %H:%M:%S")
    if ind.day == 8 or ind.day == 9 or ind.day == 10:
        temp.append(dataok[i])

dataok = temp

# 把dataok输出为csv文件，表头为time, temperature
dataok = pd.DataFrame(dataok, columns=['time', 'temperature'])
dataok.to_csv('hangzhou_00_23_9_7890_ok.csv', index=False)
预测
    import pandas as pd
import numpy as np
import matplotlib
import statsmodels.api as sm
import seaborn as sns
import matplotlib.pyplot as plt
```

```
from scipy import stats
from statsmodels.tsa.stattools import adfuller
import warnings

warnings.filterwarnings("ignore")  # 忽略输出警告
df = pd.read_csv('temperature_data.csv')
df = df.set_index('time')
df.index = pd.to_datetime(df.index)
# 过滤每年 8 月和 9 月的数据
df = df[df.index.month.isin([9]) & df.index.day.isin([8, 9, 10])]
matplotlib.rcParams['font.family'] = 'SimHei'
plt.rcParams['axes.unicode_minus'] = False
plt.figure()
plt.plot(df.index, df['temperature'], color='black', linewidth=0.5)
plt.xlabel('时间')
plt.ylabel('温度')
plt.title('每年9月8,9,10日的温度变化')
plt.show()
temperature = df['temperature']


# 创建一个函数来检查数据的平稳性
def test_stationarity(timeseries):
    # 执行Dickey-Fuller测试
    print('Results of Dickey-Fuller Test:')
    dftest = adfuller(timeseries, autolag='AIC')
    dfoutput = pd.Series(dftest[0:4], index=['Test Statistic', 'p-value', '#Lags Used', 'Number of Observations Used'])
    for key, value in dftest[4].items():
        dfoutput['Critical Value (%s)' % key] = value
    print(dfoutput)


# 检查原始数据的平稳性
test_stationarity(temperature)

# 将数据化为平稳数据
# 一阶差分
temperature_diff1 = temperature.diff(1)
# 24步差分
temperature_seasonal = temperature_diff1.diff(24)  # 非平稳序列经过d阶常差分和D阶季节差分变为平稳时间序列
```

```
# 十二步季节差分平稳性检验结果
test_stationarity(temperature_seasonal.dropna())
temperature_seasonal = temperature_seasonal.dropna()
# 可视化原始数据和差分后的数据
plt.figure()
plt.plot(temperature, label='Original')
plt.show()
plt.figure()
plt.plot(temperature_diff1, label='1st Order Difference')
plt.show()
plt.figure()
plt.plot(temperature_seasonal, label='seasonal Order Difference')
plt.show()
"""ACF,PACF图"""
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf

pacf = plot_pacf(temperature_seasonal, lags=40)
plt.title('PACF')
pacf.show()
acf = plot_acf(temperature_seasonal, lags=40)
plt.title('ACF')
acf.show()

#LB白噪声检验
from statsmodels.stats.diagnostic import acorr_ljungbox


def test_white_noise(data, alpha):
    lb_test_results = acorr_ljungbox(data, lags=1)
    lb = lb_test_results.iloc[0, 0]
    p = lb_test_results.iloc[0, 1]
    if p < alpha:
        print('LB白噪声检验结果：在显著性水平%s下，数据经检验为非白噪声序列' % alpha)
    else:
        print('LB白噪声检验结果：在显著性水平%s下，数据经检验为白噪声序列' % alpha)


#白噪声检验
test_white_noise(temperature_seasonal.dropna(), 0.01)

import itertools
```

```
# 搜索法定阶
def SARIMA_search(data):
    p = q = range(0, 3)
    s = [24]  # 周期为24
    d = [1]  # 做了一次季节性差分
    PDQs = list(itertools.product(p, d, q, s))  # itertools.product()得到的是可迭代对象的笛卡儿积
    pdq = list(itertools.product(p, d, q))  # list是python中是序列数据结构，序列中的每个元素都分配一个数字定位位置
    params = []
    seasonal_params = []
    results = []
    grid = pd.DataFrame()
    for param in pdq:
        for seasonal_param in PDQs:
            # 建立模型
            mod = sm.tsa.SARIMAX(data, order=param, seasonal_order=seasonal_param, enforce_stationarity=False,
                                 enforce_invertibility=False)
            # 实现数据在模型中训练
            result = mod.fit()
            print("ARIMA{}x{}-BIC:{}".format(param, seasonal_param, result.bic))
            # format表示python格式化输出，使用{}代替%
            params.append(param)
            seasonal_params.append(seasonal_param)
            results.append(result.aic)
    grid["pdq"] = params
    grid["PDQs"] = seasonal_params
    grid["bic"] = results
    print(grid[grid["bic"] == grid["bic"].min()])


#SARIMA_search(temperature)
#建立模型
model = sm.tsa.SARIMAX(temperature_seasonal, order=(1, 1, 2), seasonal_order=(0, 1, 2, 24))
SARIMA_m = model.fit()
print(SARIMA_m.summary())
#模型检验
test_white_noise(SARIMA_m.resid, 0.05)  #SARIMA_m.resid提取模型残差，并检验是否为白噪声
fig = SARIMA_m.plot_diagnostics(figsize=(15, 12))  #plot_diagnostics对象允许我们快速生成模型诊断并调查任何异常行为
```

```
#模型预测
from sklearn.metrics import mean_squared_error as mse
from sklearn.metrics import mean_absolute_error as mae


# 获取预测结果，自定义预测误差
def PredictionAnalysis(data, model, start, dynamic=False):
    pred = model.get_prediction(start=start, dynamic=dynamic, full_results=True)
    pci = pred.conf_int()  #置信区间
    pm = pred.predicted_mean  #预测值
    truth = data[start:]  #真实值
    pc = pd.concat([truth, pm, pci], axis=1)  #按列拼接
    pc.columns = ['true', 'pred', 'up', 'low']  #定义列索引
    print("1、MSE:{}".format(mse(truth, pm)))
    print("2、RMSE:{}".format(np.sqrt(mse(truth, pm))))
    print("3、MAE:{}".format(mae(truth, pm)))
    return pc


#绘制预测结果
def PredictonPlot(pc):
    plt.figure()
    plt.plot(pc['true'], label='base data')
    plt.plot(pc['pred'], label='prediction curve')
    plt.legend()
    plt.show()
    return True


pred = PredictionAnalysis(temperature_seasonal, SARIMA_m, start='2010-09-09 01:00:00')  # 预测全部
PredictonPlot(pred)

#预测未来
forecast = SARIMA_m.get_forecast(steps=72)
pre = forecast.predicted_mean
pre.index = pd.date_range(start='2024-09-08 00:00', end='2024-09-10 23:00', freq='H')
pre = np.array(pre)
temperature_copy = np.array(temperature.copy())
pre[0:24] += temperature_copy[-24:]
for i in range(24, 72):
    pre[i] += pre[i - 24]
```

```
pred = pd.DataFrame(pre, index=pd.date_range(start='2024-09-08 00:00', end='2024-09-10 23:00', freq='H'))
#预测整体可视化
plt.figure()
plt.plot(pred.index, pred)
plt.legend(['预测温度'], loc="best")
plt.xlabel("时间")
plt.ylabel("温度")
plt.show()
filltered_pred = pred.between_time(start_time='09:00', end_time='21:00')
filltered_pred.to_csv('9-21点预测温度.csv')
pred.to_csv('全时段预测温度.csv')

求解
clc;clear;



%Temp = [31.0393 30.6877 27.7918 27.2508 26.9028 26.6937 25.4812 25.1034 24.9096 24.0990 24.0520 23.9210];
%Temp = [36.5371 36.0437 35.8349 34.8570 34.6696 34.4999 33.5301 33.4014 35.6331 36.2740 36.7812 37.4578];
% Temp = [29.7356 33.1578 33.5305 33.7628 35.3402 35.2895 35.3553 34.5069 34.0253 30.5248 30.1156 29.8401];
Temp = [ 26.1693 30.1115 30.3574 30.7528 32.7305 32.8923 32.7838 32.0647 31.4891 27.9131 27.2714 26.6295];

for i = 1:12
    if Temp(i) >= 38
        Temp(i) = Temp(i) - 6.75;
    else
        Temp(i) = Temp(i) - 5;
    end
end

ks = 0.4621 .* 10 .^ ((Temp-25)./5);

% 人数
ns = [20 80 0 150 180 0 210 340 0 280 420 190];

t0 = 9; %初始时间
c_0 = 0.6; % 初始浓度
add_times = 0;%加氯次数
```

```
add_t = [];

delta_t = 0;


%  时间步长0.01
T = [0];
C = [0.6];


%  第几个小时
ind = 1;
while ind <= 21-9

    if ns(ind) ~= 0
        %这个小时有人

        %  计算降到0.3的时间
        t = func4(c_0,0.3,ks(ind));
        if ind == ceil(delta_t + t)
            %当没到下一个小时时
            T = [T (T(end)+[0:0.001:t])];
            c = func1(c_0,ks(ind),[0:0.001:t]);
            C = [C c];

            delta_t = delta_t + t;
            c_0 = 0.6;add_times = add_times+1; add_t = [add_t delta_t];%加氯

        else
            T = [T (T(end)+[0:0.001:ind-delta_t])];
            c = func1(c_0,ks(ind),[0:0.001:ind-delta_t]);
            C = [C c];

            %  计算这个小时末浓度
            c_0 = func1(c_0,ks(ind),ind-delta_t);
            delta_t = ind;
            ind = ind + 1;

            % %若下一小时有人，加满氯
            % if ind <= 12 && ns(ind-1)==0 && ns(ind)~=0
            %         c_0 = 0.6;add_times = add_times +1;
            % end
```

```
            end
        else
            %没人
            T = [T (T(end)+[0:0.001:ind-delta_t])];
            c = func1(c_0,ks(ind),[0:0.001:ind-delta_t]);
            C = [C c];

            % 计算这个小时末浓度
            c_0 = func1(c_0,ks(ind),ind-delta_t);
            delta_t = ind;
            ind = ind + 1;

            %若下一小时有人，加满氯
            if ind <= 12 && ns(ind-1)==0 && ns(ind)~=0
                c_0 = 0.6;add_times = add_times +1;add_t = [add_t delta_t];
            end
        end
end

add_t = add_t +9;
datetime_str = datestr(add_t./24, 'HH:MM:SS');



figure(1)
plot(9+T,C,'k');hold on;
xlabel("时间");ylabel("余氯浓度mg・L(-1)");
title("2024/9/9 9:00-21:00余氯浓度变化曲线");
xlim = [9:21];xticks(9:1:21);



function y = func1(c_0,k,t)
%根据初始浓度和时间计算结束浓度
    y = c_0 * exp(-k*t);

end

function y = func4(c_0,c_t,k)
% 计算降到0.3耗时
    y = - (log(c_t/c_0))/k;

end
```

代码5

介绍：用python对问题五进行求解

```
clc;clear;

%温度
Temp = [ 26.1693     30.1115     30.3574     30.7528     32.7305     32.8923     32.7838     32.0647
31.4891     27.9131     27.2714     26.6295];
ind = [1,2,4,5,7,8,10,11,12];
for i = 1:12
    if Temp(i) >= 38
        Temp(i) = Temp(i) - 6.75;
    else
        Temp(i) = Temp(i) - 5;
    end
end

a = func_c(0,30);

figure;
for j = 1:9
    i = ind(j);
    ns = [0:1:520];
    subplot(3,3,j)
    cs = func_c(ns,Temp(i));
    plot(ns,cs,'-k');hold on;
    plot(ns,ones(1,521)*0.3,'r-');
    xlabel('n');ylabel('浓度');
    title([num2str(i+8) ':00-' num2str(i+9) ':00']);
    hold off;

end




function ct = func_c(n,T)
    c0 = 0.6;t = 1/6;
    b = [0.0069 -.001 2.510E-5    -3.624E-8];

    gb = b(1) + b(2) .* n + b(3) .* n.^2 + b(4) .* n.^3;

    ks = 0.4621 + gb;
```

```
    ct = c0 * exp(-ks*t);

end
```

由于本文篇幅有限剩余的代码将在附件中给出