基于微分方程的游泳馆余氯衰减模型

摘要

在本文当中我们建立了不同的微分方程，探究了在不同影响因素之下游泳馆中余氯浓度的变化。同时在保证人数尽量多，加氯次数尽量少以及保证泳池水质始终安全建立微分方程，使得更多杭电人能进场体验杭电游泳馆的舒适和清凉。

对于问题一, 本文通过二级动力学和反应方程建立了温度恒定的**余氯衰减微分模型**，通过常微分方程的特征根求法我们可以得到在晚上 10 点快速加氯到 0.6mg/L 过后第一次降低到 0.05mg/L 所需要的时间为 **5 小时 22 分钟 13 秒**，因此第一次需要加氯的时刻为晚上 **3 点 22 分 41 秒**。

对于问题二, 因为游泳者进入泳池过后会在**余氯浓度**和**反应速率**方面对于余氯衰减微分方程产生影响，同时在附件一给出了不同人数下泳池余氯浓度在半个小时内从 0.6mg/L 降低的情况，因此我们针对此对于微分方程的更新重新构建微分方程，对于反应速率参数和常数项进行修改，得到重构过后的微分方程。同时本文求解到 255 人游泳的情况下余氯第一次 0.6mg/L 降低到 0.3mg/L 需要 **34 分钟**。

对于问题三, 在不同的开放场次之下人数不同，在只考虑人数影响的前提之下我们只要将问题二中的微分方程代入不同人数进行求解便可，本文给出了在开放时刻泳池余氯浓度变化曲线，并且计算出来泳池加氯次数为 **16 次**。

对于问题四, 本文收集了 2014-2023 往年的 9 月 8 日-10 日的日气温、9 月均温数据以及 2024 年 8 月 9 日的气温数据，基于**气温余弦公式**利用最小二乘法进行拟合预测，然后进行 R^2 检验，得到 $R^2 = 0.897$，说明利用气温曲线进行预测拟合是合理的。对于开学 9 月 8 日至 9 月 10 日气温预测，首先我们要预测气温余弦曲线中 a、b、c 三个参数，因为 c 参数是地区参数所以与上述 8 月 9 日的数值是一样的，我们得到 **c=0.49**，对于 a，b 的预测本文比较了 LSTM、ARIMA、随机森林，并选用 **ARIMA 模型**进行预测。从而得到 9 月 8 日至 9 月 10 日的三个气温余弦模型，其中 9 月 8 日的参数 **a=26.11，b=3.87，c=0.49**，在任务二中本文已知余氯变化率跟余氯浓度的 $10^{\frac{T_{water}-25}{5}}$ 成正比例，因此在任务二中本文对于模型一中微分方程中 k 常数进行修改成 $10^{\frac{T_{water}-25}{5}}k$，并且利用**龙格库塔方法**进行求解，得到只考虑温度的余氯浓度值变化曲线。

对于问题五, 本文在考虑游泳人数对于余氯浓度和反应速率的影响，温度的影响下结合问题二和问题四的微分模型，在以人数尽量多以及加氯次数尽量少的约束之下，建立优化微分模型。本文可以求解得到整个 9 月 9 日营业期间不同时间段能够容纳的人数分别为 **389**、**350**、**452**、**417**。

关键字：余氯衰减微分模型　气温余弦公式　ARIMA 模型　龙格库塔方法

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。目前常用的依旧是氯消毒方法，专用加氯设备在需要时通过多个加药泵将氯药水加入泳池当中，由于药水浓度较大但是体积较小，因此不需要将泳池中的水排除泳池便可以完成加氯消毒工作。游泳池的标准余氯含量为0.3−0.6mg/L，低于 0.3mg/L 时消毒效果较差，略高于 0.6mg/L 时，会产生刺鼻气味，给游泳者带来不好的体验。因而游泳池中余氯浓度保持在 0.3 − 0.6mg/L 效果最好。

在泳池运营的过程当中诸如：消毒剂、气温、游泳者、营业时间、泳池容积等会影响游泳池中含氯量。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，泳池如何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 问题要求

问题 1 已知为了保证池中水质不变坏，至少要让余氯浓度不得低于 0.05mg/L（即跟自来水的余氯标准相当），在已知池水的余氯浓度从 0.6mg/L 降到 0.3mg/L 平均用时为 1 个小时 30 分钟的前提下，建立氯浓度衰减模型预测氯浓度第一次到达 0.05mg/L 需要加氯的时刻。

问题 2 在泳池有游泳者的情况下，池水因为游泳活动加上皮肤代谢产物、汗液等势必加速池中余氯衰减。附件 1 统计了水温是常温情况下游泳人数和 0.5 小时后对应的池中余氯值数据。请你根据附件 1 的数据进行建模分析人数对于余氯浓度的影响，进一步确定泳池常温下 255 人游泳（初始余氯 0.6mg/L）的第一次加氯时间。

问题 3 附件 2 中给出了游泳馆一天四场统计的平均在池游泳人数，假定当天上午九点池中余氯浓度为 0.6mg/L，请在问题 2 分析的基础上，构建合适模型给出 9 点到 21 点这段时间内的加氯次数（忽略每次加氯耗时），并求出该时段泳池中余氯浓度值变化曲线。

问题 4 问题 4 中提出了两个两个任务，其中任务一要求我们通过对于数据的收集过后预测开学 9 月 8 日至 9 月 10 日三天的气温变化曲线，并且完成气温变化曲线的绘制。

任务二要求我们在不考虑人数影响的前提下，考虑气温对于池水余氯的影响建模求解给出 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。

问题 5 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），请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，使得更多的人可以进场体验杭电游泳馆的舒适和清凉。

二、 问题分析

2.1 问题一分析

对于问题一，我们我们查阅论文过后假设游泳池中发生的氯衰减方程符合动力学和反应方程，建立微分方程。同时题目中余氯浓度从 0.6mg/L 降到 0.3mg/L 的平均用时为 1 个小时 30 分钟，基于此对于微分方程进行求解得到第一次余氯从起始浓度 0.6mg/L 降到余氯浓度为 0.05mg/L 需要加氯时间。

2.2 问题二分析

对于问题二，游泳者进入泳池过后会在余氯浓度和反应速率方面对于问题一所建立的微分方程产生影响，因此我们重新构建微分方程进行求解，再对于 255 人游泳的情况之下余氯第一次从 0.6mg/L 降低到 0.3mg/L 第一次需要加氯的时间进行求解。

2.3 问题三分析

对于问题三，从附件二中我们可以得知杭电游泳馆从九点开放到晚上九点，一共有四个开放时间。不同的开放时间泳池中的人数是不同的，因此我们只需要在问题二的基础之上对于不同时间不同人数的不同微分方程进行求解便可，同时给出余氯浓度变化曲线。

2.4 问题四分析

对于问题四，任务一要求我们对于杭电开学前三天即 9.8 号到 9.10 号的气温进行预测，我们进行论文查询过后可以知道可以利用气温余弦曲线进行拟合，我们先对于 9.8 号的气温曲线进行拟合，然后进行 R^2 检验，发现检验效果较好，因此我们 9.9 号和 9.10 号的气温都可以利用气温余弦曲线进行拟合。

任务二则是要求我们在不考虑人数影响的情况之下充分考虑水温变化对池水余氯影响，建立数学模型求解给出 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）。因为在题目当中我们知道了余氯变化率跟余氯浓度的 $10^{\frac{T_{water}-25}{5}}$ 倍成正比例关系（T_{water} 表示池水温度），因此我们可以对于微分方程中原本余氯变化率跟余氯浓度关系系数 k 进行改变，然后求解 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），以及该时段泳池中余氯浓度值变化曲线。

2.5 问题五分析

在问题五当中要求我们基于上述问题的分析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，在保证人数尽量多，加氯次数尽量少以及保证泳池水质始终安全（一直控制在 0.3-0.6mg/L 范围内）建立数学模型，因此我们综合考虑问题二和问题四的微分模型，重构出新的模型对于此进行求解。

三、 模型假设

为简化问题，本文做出以下假设：

- 假设 1，在问题当中加氯时间忽略不计。
- 假设 2，池中水温在夏秋季节比馆内气温低 2 度，在馆内气温很高时（35 度及以上）池水温度会低 3.5 度。
- 假设 3，在问题五当中只考虑优化人数，使得尽量在游泳池中的人数尽量多。

四、 符号说明

符号	说明
c	反应物浓度, 在本文中指余氯浓度
k	反应速率常数
t	反应时间
k'	游泳者进入泳池后对于 k 的影响更新后的速率常数
dc_1	单位时间内游泳者对余氯浓度的间接更新量
dc_2	单位时间内游泳者对余氯浓度的直接更新量，
a_1	反应速率与人数关系的关联参数
a_2	反映游泳人数对余氯浓度直接变化量的影响力
...	...

五、 问题一的模型的建立和求解

5.1 模型建立

余氯一般指剩余氯，是衡量游泳池消毒能力的因素，也是是泳池水质重要的指标。不过泳池余氯会受到很多因素的影响，例如消毒剂、PH 值、温度、人数等等。在本问中，我们主要考虑温度恒定的情况下，余氯自身反应速率对游泳池水中余氯浓度的影响情况。

根据查阅论文 [1]。我们基于二级动力学和反应方程，假设游泳池中发生的氯衰减方程为：

$$Cl + X \longrightarrow P$$

式中 X——不同反应性的所有反应物的集合，单位为 $mg{\cdot}L^{-1}$；P——所有反应生成物的集合，单位为 $mg{\cdot}L^{-1}$。

进而我们得到：

$$\frac{dc_{Cl}}{dt} = \frac{dc_X}{dt} = -k_{vt}{\cdot}c_{Cl}{\cdot}c_x$$

其中，$k_{vt} = F{\cdot}e^{\frac{-E_A}{R{\cdot}T}}$ 被称为反应速率系数，单位为 $mg{\cdot}L^{-1}{\cdot}h^{-1}$；式中的 F 为指前因子（单位与 k_{vt} 相同）；E_A 是活化能，单位为 kJ/mol；R 是理想气体常数，单位为 $kJ{\cdot}mol^{-1}{\cdot}K^{-1}$，都需要靠实验来得知。

易发现当温度不变时，反应速率也不变。而在本题中，反应物无从得知，因此我们将模型简化：当气温保持 25 度不变时，根据反应速率与反应速率常数的关系，得到如下的微分方程模型：

$$\frac{dc}{dt} = -kc$$

其中 c 表示反应物浓度，k 表示反应速率常数。

5.2 模型求解

由上可以得到该微分方程的解为：

$$c = ae^{-kt}$$

根据题目中已知条件：余氯从起始浓度 0.6mg/L 降到 0.3mg/L 的平均用时为 1 个小时 30 分钟，因而带入微分方程的解得到：

$$0.3 = 0.6e^{-k{\cdot}1.5}$$

解得 $k = -\frac{ln_{0.5}}{1.5} \approx 0.462$。将 k 值、余氯从起始浓度 0.6mg/L 降到余氯浓度为 0.05mg/L 代入微分方程，我们可以得到：

$$0.05 = 0.6e^{-0.462{\cdot}t}$$

解得 $t \approx 5.3774$，因此池水过 5 小时 22 分钟需要再次加氯，为晚上 3 点 22 分。

5.3 求解结果

六、 问题二的模型的建立和求解

6.1 模型建立

问题二需要考虑当余氯浓度下降至 0.3mg/L 是会开启加药泵，使得余氯量恢复至 0.6mg/L。考虑有游泳者的情况时，因为其在水中的搅动，加上汗液和身体细菌以及小孩尿液等等，会对游泳池中余氯浓度产生明显的影响，同时也会对于反应速率产生影响。因此本文需要探讨游泳者对余氯浓度和反应速率的影响。依据题目这里假设不考虑加药时间。给出游泳人数对于余氯浓度的影响情况示意图：

6.1.1 游泳者对于反应速率的更新

游泳者进入泳池过后，因为各种行为，以及带来的其他物质会对于反应速率产生一定的影响。因为在问题一当中我们已经得到微分方程模型 $\frac{dc}{dt} = -kc$，问题二中游泳者进入反映到模型当中的是 k 的变化：

$$k' = a_2 nk$$

其中 k' 为更新过后的速率常数，n 为游泳人数，a_2 是参数，反映游泳人数对速率常数的影响力。基于附件一我们给出人数与 k 值比例和游泳人数的示意图：

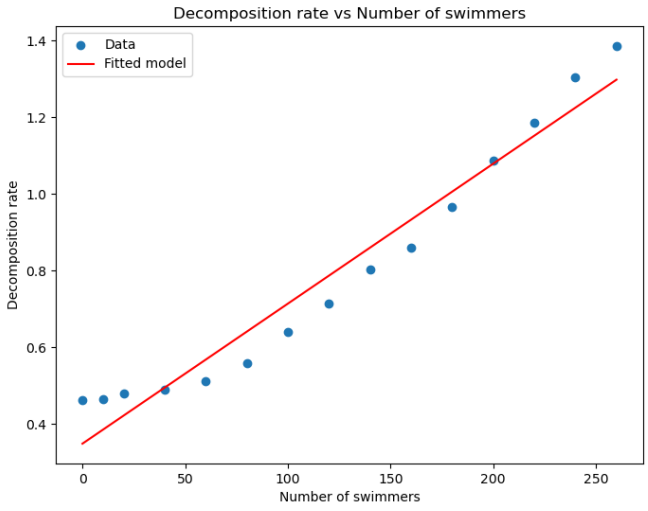

图 1　人数与 k 值比例和游泳人数的示意图

由此我们可以写出该微分方程：

$$\frac{dc_1}{dt} = -a_2 nkc$$

其中 dc_1 表示单位时间内游泳者对余氯浓度的间接更新量, 该微分方程就是含游泳者的余氯衰减微分方程

6.1.2 游泳者对于余氯浓度的更新

除去对于反应速率的影响，游泳者可能对于余氯浓度有直接性的影响，这种变化与当前余氯浓度是不相关的，因此这种直接对余氯浓度产生的变化量有:

$$\frac{dc_2}{dt} = -(a_1 n^2 + a_3)kc$$

其中 dc_2 表示单位时间内游泳者对余氯浓度的直接更新量，a_1 与 a_3 是参数反映游泳人数对余氯浓度直接变化量的影响力。因此我们可以将问题一中给出的常系数微分方程进行一定的修改成:

$$\frac{dc_1}{dt} = -(a_1 n^2 + a_2 n + a_3)kc$$

其中 dc 表示更新后单位时间内余氯的浓度变化量，该微分方程为存在游泳者的余氯衰减微分方程。

6.2 模型求解

6.2.1 求解参数 a1，a2，a3

为了能够运用存在游泳者的余氯衰减微分方程，首先要求解出参数 a_1,a_2,a_3 的值，对此本文根据附件 1 的数据，对 a_1,a_2,a_3 的值进行从大范围逐步缩减范围的搜索，以求出最合适的 a_1,a_2,a_3。首先定义参数 a_1,a_2,a_3 初始值均为 0.01, 接着采用损失函数 loss，量化模型预测余氯浓度值与实际余氯浓度观测值之间的偏差，帮助找到一组参数，使得预测值与观测值之间的残差平方和最小，最后运用最小二乘法找到使 loss 函数值最小的参数 a_1,a_2,a_3。

最终得到 $a_1 = 0.000025$,$a_2 = 0.00155$,$a_3 = 0.9765$。因此常系数微分方程为:

$$\frac{dc_1}{dt} = -(0.000025n^2 + 0.00155n + 0.9765)kc$$

下图我们给出拟合出来的方程图像:

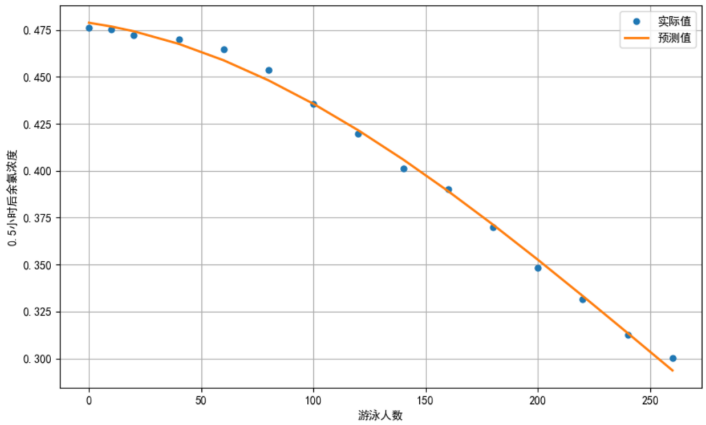

图 2　游泳人数拟合图

6.2.2 求解微分方程

在问题一中我们先得到微分方程的解过后，从而进行模型的求解，在问题二中我们也遵项该流程。解得微分方程的解为：

$$c(t) = ce^{-\left(0.000025n^2+0.00155n+0.9765\right)kt}$$

在问题二中要求我们对于在 255 人游泳的情况下，给出第一次氯浓度从 0.6mg/L 降低到 0.3mg/L 时的时刻。因此我们将其代入微分方程进行求解：

$$0.3 = 0.6e^{-\left(0.000025n^2+0.00155n+0.9765\right)kt}$$

通过将上述微分方程代入 python 中进行求解，我们可以得到 t 的具体取值，得到过多久过后第一次进行加氧。

6.3 求解结果

通过求解微分方程我们可以得到 $t \approx 0.54$，即为过 34 分钟过后进行第一次加氧。

七、 问题三的模型的建立和求解

7.1 模型建立

在问题三给出的附件二当中我们知道了杭电游泳馆从早上九点开放到晚上九点，一共有四个开放时间，分别为 9:00-11:00，12:00-14:00，15:00-17:00，18:00-21:00。其中间隔的 11:00-12:00，14:00-15:00，17:00-18:00 为闭馆维护，没有人数在泳池当中。

表 1　在池平均人数统计表

开放场次	时间段	在池平均人数统计
第一场	9:00–10:00	20
第一场	10:00–11:00	80
闭馆维护	11:00–12:00	0
第二场	12:00–13:00	150
第二场	13:00–14:00	180
闭馆维护	14:00–15:00	0
第三场	15:00–16:00	210
第三场	16:00–17:00	340
闭馆维护	17:00–18:00	0
第四场	18:00–19:00	280
第四场	19:00–20:00	420
第四场	20:00–21:00	190

在问题二当中我们已经建立了存在人数影响余氯浓度和余氯反应速率的微分方程：

$$c(t) = ce^{-\left(0.000025n^2+0.00155n+0.9765\right)kt}$$

因此我们需要在不同时间段，代入不同人数进入问题二的微分方程，因此我们可以将该方程重写为：

$$\begin{cases} c(t) = ce^{-\left(0.000025*20^2+0.00155*20+0.9765\right)kt}, n = 20 \\ c(t) = ce^{-\left(0.000025*80^2+0.00155*80+0.9765\right)kt}, n = 80 \\ c(t) = ce^{-\left(0.000025*150^2+0.00155*150+0.9765\right)kt}, n = 150 \\ c(t) = ce^{-\left(0.000025*180^2+0.00155*180+0.9765\right)kt}, n = 180 \\ c(t) = ce^{-\left(0.000025*210^2+0.00155*210+0.9765\right)kt}, n = 210 \\ c(t) = ce^{-\left(0.000025*340^2+0.00155*340+0.9765\right)kt}, n = 340 \\ c(t) = ce^{-\left(0.000025*280^2+0.00155*280+0.9765\right)kt}, n = 280 \\ c(t) = ce^{-\left(0.000025*420^2+0.00155*420+0.9765\right)kt}, n = 420 \\ c(t) = ce^{-\left(0.000025*1900^2+0.00155*190+0.9765\right)kt}, n = 190 \end{cases}$$

因为在营业过程当中有闭馆维护，因此 n 是可以得到 0 的，所以我们的微分方程可以写成：

$$c(t) = c_0 e^{-kt}, n = 0$$

综上我们模型三的建立其实便是在考虑实际开放过程当中不同人数的基础之上，重构问题二的微分方程模型。在模型求解当中问题三要求我们对于 9 点到 21 点这段时间

内的加氯次数（忽略每次加氯耗时）进行求解，并求出该时段泳池中余氯浓度值变化曲线。

7.2 模型求解

问题要求的是降低到 0.3mg/L 便要进行加氯，因此我们在求解过程当中利用 python 编写程序规则进行求解，当氯浓度从 0.6mg/L 降低到 0.3mg/L 时便统计一次加氯次数，并且在瞬间将氯浓度添加到 0.6mg/L。同时在附注当中给出了在闭馆休息期间是停止加氯的，在营业之前进行一次加氯使氯浓度达到 0.6mg/L，因此我们允许在闭馆休息期间氯浓度降低到 0.3mg/L 以下，只有在营业期间氯浓度达到 0.3mg/L 时进行加氯。

7.3 求解结果

通过 python 求解我们可以得到加氯次数为 8 次，下面为 9 点到 21 点的氯浓度变化曲线：

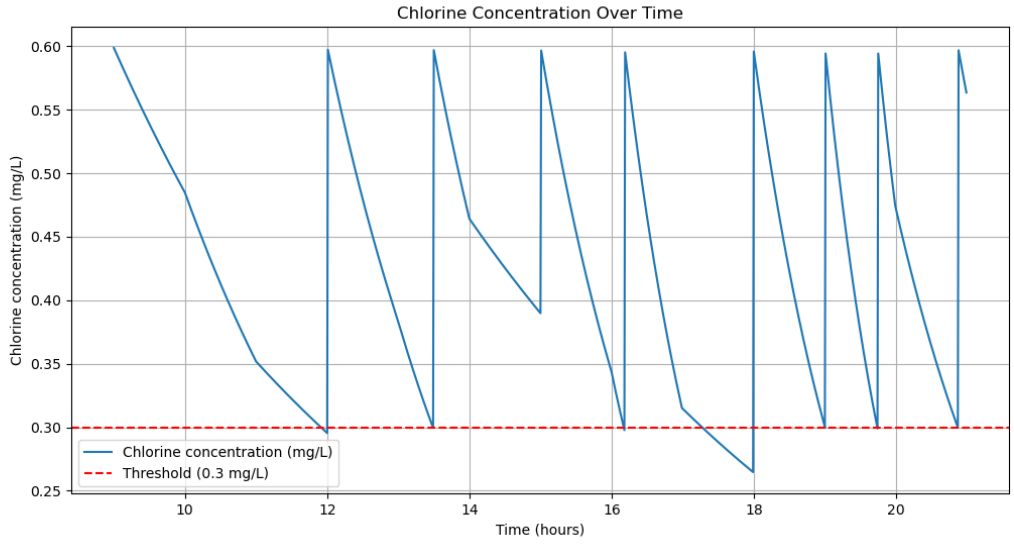

图 3　9 点到 21 点的氯浓度变化曲线图

其中同时我们能够得到加氯时间点分别为:

表 2　加氯时间点

加氯时间点	9	12	13.5	15	16.189	18	19.01	19.75	20.88

八、 问题四的模型的建立和求解

8.1 任务一模型建立

问题四的任务一要求根据收集的气温数据，预测杭电 9 月 8 日至 9 月 10 日的气温变化曲线。因此我们从中国气象站中获取了 2024 年 8 月 9 日的逐小时气温和从 2014 年开始历年 9 月均温以及 9 月 8 日、9 日、10 日的最高气温和最低气温，来对后续的建模进行准备。

对于气温日变化，想要用随时间变化的表达式进行描述，首先绘制 2024 年 8 月 9 日的气温变化曲线图如下：

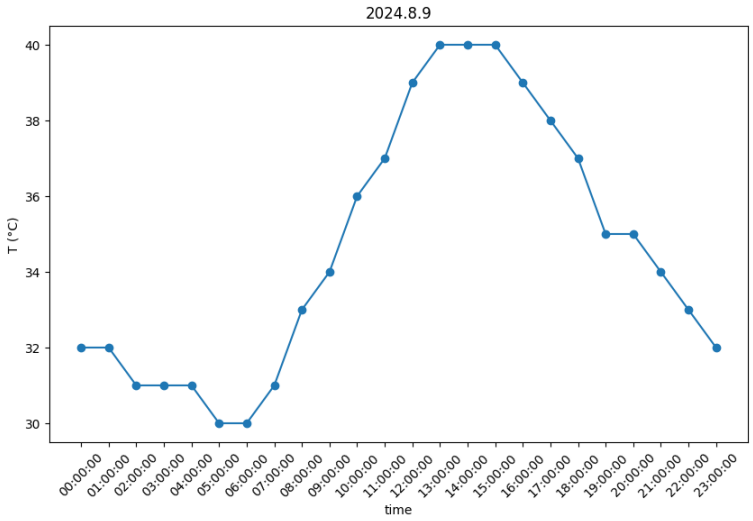

图 4　气温变化曲线图

根据上图易知，8 月 9 日在 15 时左右出现最高气温，为 36□；在 4 时左右出现最低气温，为 28□。整个气温变化从 0 时开始先下降后上升再下降。对于气温日变化规律，由此可以总结出一种表达式来表示气温在一天内逐小时的变化为：

$$T = a - bcos(2\pi(h - c)/24)$$

其中 h 表示时间，参数 a 表示该月份的平均气温，参数 b 表示该日气温变化幅度的二分之一，c 是时角的位移量，与地区所处位置相关。在该式子中，参数 a 和参数 b 可以直接由气温数据得到。

不难发现上式其实是对余弦公式做出相关变换得到的，因此根据余弦公式对上式做出调整可得：

$$T(t) = a - bcos(\frac{\pi}{12}t - c)$$

其中参数 a、b 不变，c 做相关变化，但是依然只与地区位置有关，t 表示该日的时间。接下来需要对该式子利用 ARIMA 时间序列进行拟合。参数 a，b，c 的估计值为：

$$\hat{a},\hat{b},\hat{c} = argmin(\sum_{i=1}^{n}(y_i - a + bcos(\frac{\pi}{12}x_i - c)^2)$$

8.2 任务一模型求解

8.2.1 求解 8 月 9 日气温日变化模型

对于气温余弦方程，我们利用最小二乘法进行拟合求解出参数 a, b, c。求解得到的参数值如下：

$$a = 34.58, b = 4.58, c = 0.49$$

因此我们能够得到余弦方程为

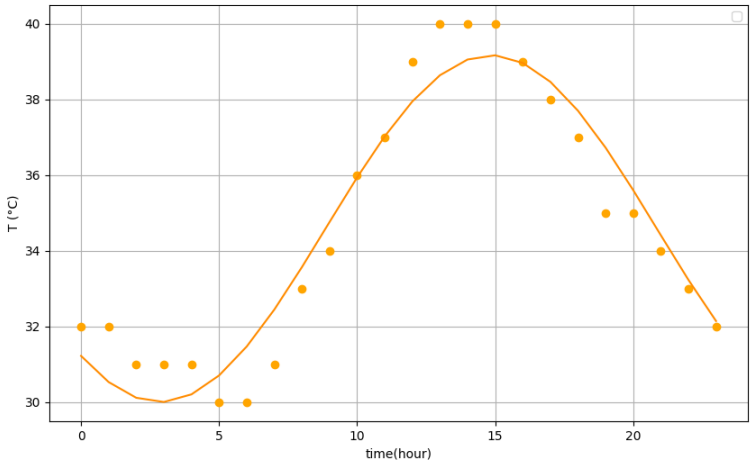

图 5　余弦方程曲线

$$y = 34.58 - 4.58 \cdot \cos\left(\frac{\pi}{12}x - 0.49\right)$$

在该参数数值拟合下，我们能够得到曲线的 $R^2 = 0.928$ 因此我们能够知道拟合效果较好，同时对于余弦方程和散点图的观察我们能够知道拟合出来的最高温和拟合出来的最低温跟原始的最高最低温基本一致，因而该气温余弦方程模型是准确的，可以用日最高温、日最低温、月平均温度来预测日气温变化情况。

8.2.2 预测 9 月 8 日-10 日气温变化曲线

确定参数 a，b

本题中我们采用 ARIMA 时间序列预测方法，并对其采用自相关函数 ACF 检验。

ARIMA 模型是一种流行且广泛使用的时间序列预测统计方法。本题中我们采用差分自回归移动平均模型：ARIMA（p，d，q）来对 9 月 8 日-10 日的气温进行预测，其中 AR 是自回归，p 是自回归项；MA 是移动平均，q 为移动平均项数，d 是时间序列成为平稳时所做的差分次数。下面对该模型进行说明：

AR：自回归模型，描述的是当前值与历史值之间的关系，用变量自身的历史时间数据对自身进行预测。p 阶自回归过程的公式为：

$$y_t = \mu + \sum_{i=1}^{p} \gamma_i y_{t-i} + \epsilon$$

其中 y_t 是当前值，μ 是常数项，P 是阶数,γ_i 是自相关系数，ϵ 是误差。

MA：移动平均模型，关注的是自回归模型中的误差项的累加。q 阶自回归过程的公式为：

$$y_t = \mu + \epsilon_t + \sum_{i=1}^{q} \theta_i \epsilon_{t-i}$$

由此得出 ARIMA 模型的公式为：

$$y_t = \mu + \sum_{i=1}^{p} \gamma_i y_{t-i} + \epsilon + \sum_{i=1}^{q} \theta_i \epsilon_{t-i}$$

其中 d 为时间序列成为平稳时所做的差分次数。特别地，在本题中我们将 p，q，d 均设为 1。

综上，我们可以得出参数 a，b 的值分别为：参数 a 均为 26.12，参数 b 的值对应 9 月 8 日，9 月 9 日，九月 10 日分别为 3.88，3.67 和 3.27。下面我们给出 ARIMA 预测 9 月 8 日到 9 月 10 日曲线图：

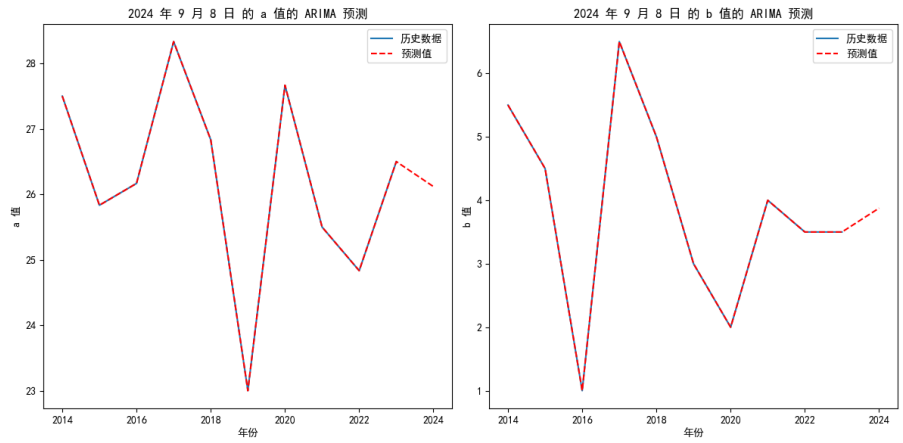

图 6　预测 9 月 8 日曲线图

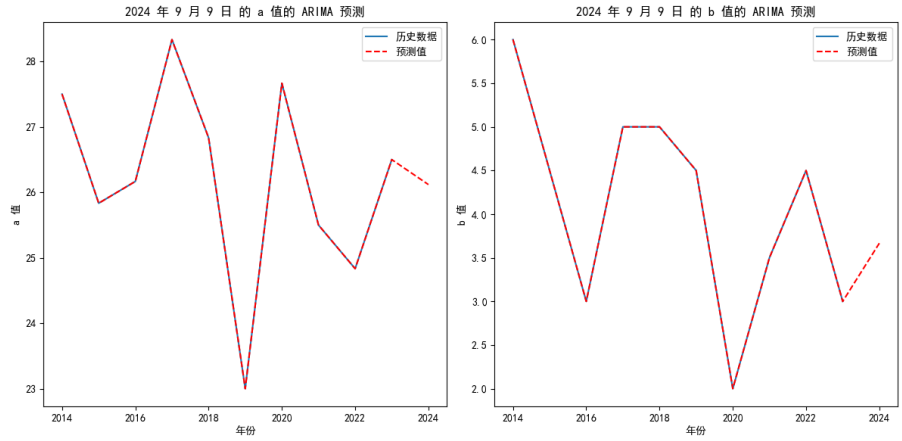

图 7　预测 9 月 9 日曲线图

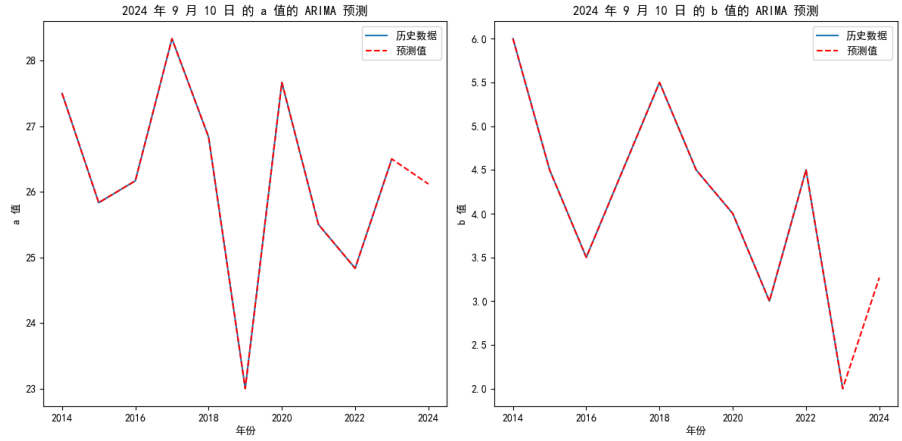

图 8　预测 9 月 10 日曲线图

同时为了检验模型的预测的效果，我们采用 ACF 进行检验。ACF 是一个完整的自相关函数，可将有序的随机变量与其自身比较，反映了同一序列在不同时序的取值之间的相关性。公式为：

$$ACF_k = \rho_k = \frac{Cov(y_t, y_{t-k})}{Var(y_t)}$$

预测的 ACF 值如下图所示：

a 值：

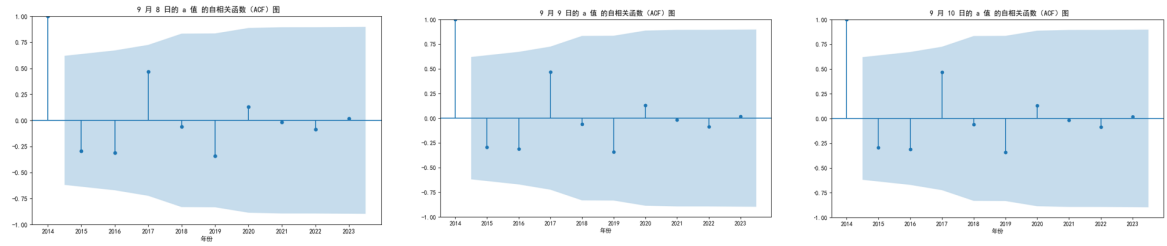

图 9　预测 a 值时的 ACF 值

b 值：

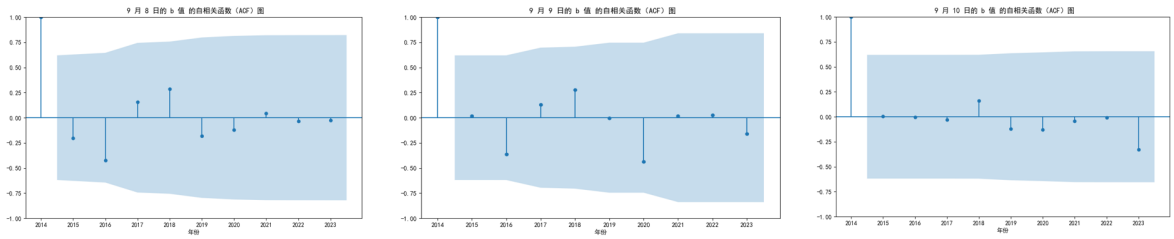

图 10　预测 b 值时的 ACF 值

我们可以得知在预测 a 值和预测 b 值时的 ACF 值都在 0 附近，因此 ARIMA 的预测结果是可信的。

确定参数 c

由于参数 c 只与地区有关，而该测试点都是在杭州进行，地区不变，因而参数 c 不变。由 2024 年 8 月 9 日的拟合曲线可得 c = 0.49。

确定预测气温余弦公式

根据确定好的参数 a，b，c，我们可以得到 9 月 8 日、9 月 9 日、9 月 10 日的气温变化关系式为：

$$\begin{cases} T_{9.8}(t) = 26.12 - 3.88cos(\frac{\pi}{12}t - 0.49) \\ T_{9.9}(t) = 26.12 - 3.67cos(\frac{\pi}{12}t - 0.49) \\ T_{9.10}(t) = 26.12 - 3.27cos(\frac{\pi}{12}t - 0.49) \end{cases}$$

由于游泳池与室外有温差，所以我们加入 ϵ 表示室内外的温度差，在室外气温低于 $35°C$ 时 $\epsilon = 2$，在室外气温高于 $35°C$ 时 $\epsilon = 3.5$。

$$\begin{cases} T_{9.8}(t) = 26.12 - 3.88cos(\frac{\pi}{12}t - 0.49) - \epsilon \\ T_{9.9}(t) = 26.12 - 3.67cos(\frac{\pi}{12}t - 0.49) - \epsilon \\ T_{9.10}(t) = 26.12 - 3.27cos(\frac{\pi}{12}t - 0.49) - \epsilon \end{cases}$$

8.3 任务一求解结果

根据上述预测所得的气温变化式，我们可以完善下表:

表 3　杭电游泳馆三天相应的池水温度预测值

	2024 年 9 月 8 日	2024 年 9 月 9 日	2024 年 9 月 10 日
最高温度	24.99°C	24.78°C	24.39°C
最低温度	21.00°C	21.02°C	21.03°C
上午 10：00	23.16°C	23.04°C	22.83°C
中午 12：00	24.53°C	24.34°C	23.99°C
下午 14：00	24.99°C	24.78°C	24.39°C
下午 16：00	24.41°C	24.24°C	23.90°C
晚上 20：00	21.00°C	21.02°C	21.03°C

8.4 任务二模型建立与求解

对于任务二，不考虑泳池人数的影响，仅考虑水温变化所引起的对余氯值的影响，根据题式给出的余氯变化率跟余氯浓度的关系式，我们可以得出余氯浓度的微分方程为:

$$\frac{dc}{dt} = c_0 \cdot e^{-10^{\frac{T_{water}-25}{10}}kt}$$

通过重构的微分方程，我们将 T_{water} 代入可以得到

$$\begin{cases} \frac{dc}{dt}_{9.8} = c_0 \cdot e^{-10^{\frac{26.12-3.88cos(\frac{\pi}{12}t-0.49)-\epsilon-25}{10}}kt} \\ \frac{dc}{dt}_{9.9} = c_0 \cdot e^{-10^{\frac{26.12-3.67cos(\frac{\pi}{12}t-0.49)-\epsilon-25}{10}}kt} \\ \frac{dc}{dt}_{9.10} = c_0 \cdot e^{-10^{\frac{26.12-3.27cos(\frac{\pi}{12}t-0.49)-\epsilon-25}{10}}kt} \end{cases}$$

对于该方程的求解我们利用四阶龙格库塔方法进行求解，龙格库塔法是一种求解微分方程数值解的方法，因其在许多情况下能够提供较高的精度而广泛使用，它属于隐式迭代方法，以泰勒公式和使用斜率近似替代微分，求解精度高。在已知方程导数和初值

信息的情况下，利用计算机仿真时应用，省去求解微分方程的复杂过程。对于已知初值的一阶常微分方程：

$$y' = f(t, y), \quad y(t_0) = y_0$$

则对于该问题，由如下方程给出：

$$y_{n+1} = y_n + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

其中：

$$\begin{aligned} k_1 &= f(t_n, y_n) \\ k_2 &= f\left(t_n + \frac{h}{2}, y_n + \frac{h}{2}k_1\right) \\ k_3 &= f\left(t_n + \frac{h}{2}, y_n + \frac{h}{2}k_2\right) \\ k_4 &= f(t_n + h, y_n + hk_3) \end{aligned}$$

我们得到四阶龙格库塔的方法过后，对于余氯曲线的求解我们通过 python 程序计算机模拟可以求解得到如下所示：

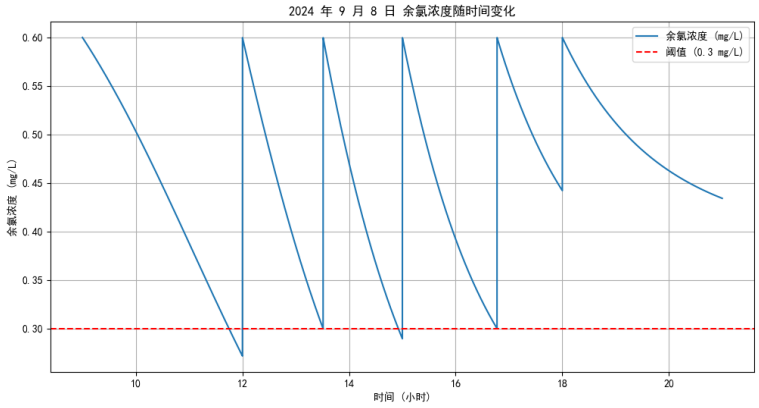

图 11　9.8 日的余氯变化曲线

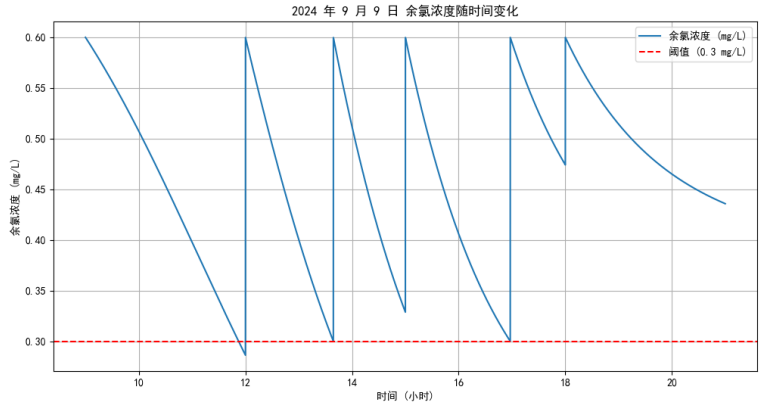

图 12　9.9 日的余氯变化曲线

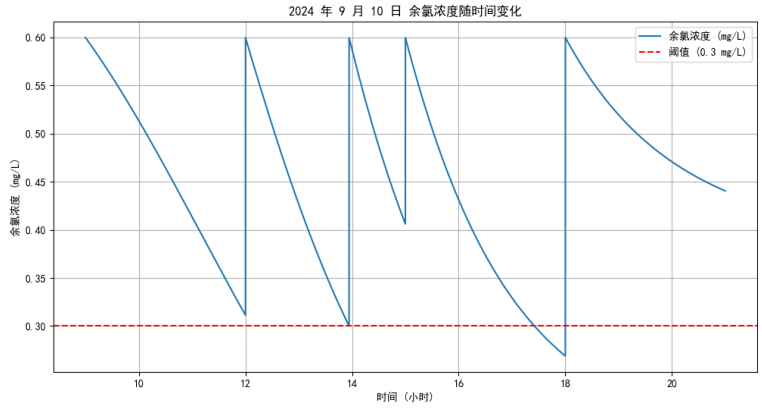

图 13　9.10 日的余氯变化曲线

九、 问题五的模型的建立和求解

9.1 模型建立

在问题五当中要求我们综合考虑温度，人数等因素重新构造模型，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，让更多杭电人能进场体验杭电游泳馆的舒适和清凉。

首先仅考虑泳池人数，得出余氯浓度与时间的关系：

$$\frac{dc_1}{dt} = -(a_1 n^2 + a_2 n + a_3)ck$$

由问题二可知，$a_1 = 0.000025$，$a_2 = 0.00155$，$a_3 = 0.9765$。进一步考虑我们在问题四当中求出了 9 月 9 号的气温余弦方程，即考虑到池水温度对余氯浓度的影响，模型

被修正为:

$$\frac{dc_2}{dt} = \frac{dc_1}{dt} * 10^{\frac{A-B\cdot(cos\frac{\pi t}{12}-C)-29.5}{5}}$$

其中，A=26.118，B=3.666，C=0.49。

综上所述，我们可以将微分方程重构为:

$$\frac{dc_2}{dt} = -(a_1n^2 + a_2n + a_3)ck * 10^{\frac{A-B\cdot(cos\frac{\pi t}{12}-C)-29.5}{5}}$$

因此我们将优化模型写成

$$Max \quad n$$

$$\begin{cases} \frac{dc_2}{dt} = -(a_1n^2 + a_2n + a_3)ck * 10^{\frac{A-B\cdot(cos\frac{\pi t}{12}-C)-29.5}{5}} \\ 0.3 \leq c \leq 0.6 \\ n \leq 520 \end{cases}$$

在问题五中我们以人数最大为优化目标，在加氯次数尽量少的约束之下对于重构过后的微分方程进行灵敏度分析，并且给出每个时间段可容纳的最大人数。

9.2 模型求解

题目要求每一个开放场次开始时进行一次加氯，考虑这个前提下，我们在模型求解过程中，首先力求减少加氯次数。由于在同一时刻水温保持恒定，模型中余氯的分解速率与人数之间呈单调关系。

基于此，我们固定加氯次数，并采用二分查找法确定人数。我们将二分查找的最小人数设定为 0 人，最大人数设定为 520 人。在每次查找过程中人数都将是一个定值，通过实时计算水温以此模拟氯浓度的变化并计算相应的加氯次数。同时，在搜索时我们需要考虑实际情况。这样，我们就能得出在确定加氯次数的前提下，每个时间段最多可容纳的人数。通过编写 Python 程序进行求解，得到了以下结果:

表 4　杭电游泳馆三天相应的池水温度预测值

开放场次	第一场	第二场	第三场	第四场
最多人数	390	350	452	417

十、 模型的评价

10.1 模型的优点

- 优点 1，基于二级动力学和反应方程我们建立了微分模型对于问题进行求解，可解释性和可靠度高。

• 优点 2，利用 ARIMA 模型进行气温余弦方程中参数的预测，能够得到一个较为准确的解。

• 优点 3，不同因素对于泳池余氯浓度微分方程的各种影响都考虑到，更加贴近实际。

10.2 模型的缺点

• 缺点 1，在问题五中只考虑了人数的优化，并没有考虑舒适度，门票盈利等因素，这些可能是可以进行改进的方面。

• 缺点 2，除去人数、温度等影响可能还有其它方面的因素会影响泳池余氯的变化，是可以进一步考虑的方面。

参考文献

[1] 王旭冕, 黄廷林, 邸尚志, 等. 余氯静态衰减过程及影响因素分析[J/OL]. 给水排水, 2006(11):8-12. DOI: 10.13789/j.cnki.wwe1964.2006.11.003.

[2] 董晓磊, 信昆仑, 刘遂庆, 等. 基于 Matlab 的供水管网余氯衰减模拟[J]. 中国给水排水, 2009, 25(01):49-52.

附录 A　文件列表

文件名	功能描述
problem1.py	问题一程序代码
problem2.py	问题二程序代码
problem3.py	问题三程序代码
problem4.py	问题四程序代码
problem5.py	问题五程序代码

附录 B　代码

problem1.py

```python
import math

C0 = 0.6
C1 = 0.3
t1 = 1.5
C_min = 0.05

k = -math.log(C1 / C0) / t1

t = -math.log(C_min / C0) / k

print(k)
print(t)
```

problem2.py

```python
import numpy as np
import pandas as pd
from scipy.optimize import curve_fit
import matplotlib.pyplot as plt

data = pd.read_excel('xlsx1.xlsx')
# print(data)

```

```
data_without_header = data.iloc[1:]
n = data_without_header.iloc[:, 0].to_numpy()
C_0_5h = data_without_header.iloc[:, 1].to_numpy()
C_0_5h = C_0_5h.astype('float')
C_0_5h_transformed = (np.log(C_0_5h / 0.6)) / -0.5
#print(n)
#print(C_0_5h)
print(C_0_5h_transformed)

#def f(X, A, B):
#    return 0.6 * np.exp(-0.5 * (A * X + B))
def f2(X, A, B):
    return A * X + B

popt, pcov = curve_fit(f2, n, C_0_5h_transformed)
#print(popt)
#print(pcov)

a_fit, b_fit = popt
#print(a_fit)
#print(b_fit)

plt.figure(figsize=(8, 6))
plt.scatter(n, C_0_5h_transformed, label='Data')
plt.plot(n, f2(n, popt[0],popt[1]), color='red', label='Fitted
    model')
plt.xlabel('Number of swimmers')
plt.ylabel('Decomposition rate')
plt.title('Decomposition rate vs Number of swimmers')
plt.legend()
plt.show()

print(a_fit)
print(b_fit)
```

problem3.py

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

A = 0.003648497349718996
B = 0.34939455820847437
K = 0.46209812037329684


def f1(t, C0):
    return C0 * np.exp(-K * t)
def f2(t, n, C0):
    return C0 * np.exp(-0.5 * (A * n + B) * t)

df = pd.read_excel('xlsx2.xlsx')
#print(df)
non_maintenance_df = df.dropna(subset=['时间段'])
time_slots = [(int(time.split('-')[0].split(':')[0]), int(time
    .split('-')[1].split(':')[0]))
              for time in non_maintenance_df['时间段']]
n_values = non_maintenance_df['在池游泳人数平均统计'].tolist()

#print(time_slots)
#print(n_values)
#print(time_slots2)
#print(n_values2)

def simulate_chlorine_concentration(time_slots, n_values):
    C0 = 0.6
    chlorine_concentration = []
    chlorine_times = []
    current_time = time_slots[0][0]

    for i, (start, end) in enumerate(time_slots):
```

```
        n = n_values[i]
        if (i == 0 or i == 3 or i == 6 or i == 9):
            chlorine_times.append(start)
            C0 = 0.6
        tt=0.01
        while current_time < end:
            print(current_time)
            #if n == 0:
            #    C = f1(tt, C0)
            #else:
            C = f2(tt, n, C0)
            chlorine_concentration.append((current_time, C))
            current_time += tt
            if(i == 2 or i == 5 or i == 8):
                C0 = C
            else:
                if C <= 0.3:
                    chlorine_times.append(current_time)
                    C0 = 0.6
                else:
                    C0 = C

    return chlorine_concentration, chlorine_times


chlorine_concentration, chlorine_times =
   simulate_chlorine_concentration(time_slots, n_values)

times, concentrations = zip(*chlorine_concentration)
plt.figure(figsize=(12, 6))
plt.plot(times, concentrations, label='Chlorine concentration
   (mg/L)')
plt.axhline(y=0.3, color='r', linestyle='--', label='Threshold
    (0.3 mg/L)')
plt.xlabel('Time (hours)')
```

```
plt.ylabel('Chlorine concentration (mg/L)')
plt.title('Chlorine Concentration Over Time')
plt.legend()
plt.grid(True)
plt.show()

print(f'加氯时间点: {chlorine_times}')
```

problem4.py

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit
from statsmodels.graphics.tsaplots import plot_acf
from statsmodels.tsa.arima.model import ARIMA
from datetime import datetime, timedelta

plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False

df = pd.read_excel('xlsx2.xlsx')
non_maintenance_df = df.dropna(subset=['时间段'])
time_slots = [(int(time.split('-')[0].split(':')[0]), int(time
    .split('-')[1].split(':')[0]))
              for time in non_maintenance_df['时间段']]

df = pd.read_excel('xlsx3.xlsx')
df.set_index('Unnamed: 0', inplace=True)
df.index.name = 'Parameter'

time_points = {
    "上午 10:00": 10,
    "中午 12:00": 12,
    "下午 14:00": 14,
    "下午 16:00": 16,
```

```
    "晚上 20:00": 20
}
dates = ["2024 年 9 月 8 日", "2024 年 9 月 9 日", "2024 年 9 月 10 日"]
aa = []
bb = []
cc = []

def temperature_model(t, c, a, b):
    return a - b * np.cos(np.pi / 12 * t - c)

def f(t, T, C0):
    return C0 * np.exp(-(10 ** ((T - 25) / 5)) * t)

def fit(model, time, temperature, a, b):
    popt, _ = curve_fit(lambda t, c: model(t, c, a, b), time, temperature)
    return popt[0]

def predict_ab(a, b, current_time):
    years = np.arange(2014, 2014 + len(a))
    def plot_acf_graph(data, title):
        month_day = title.split('年')[1]
        lags = min(9, len(data) - 1)
        plt.rcParams["figure.figsize"] = (10, 6)
        plot_acf(data, lags=lags)
        plt.title(f'{month_day} 的自相关函数（ACF）图')
        plt.xticks(ticks=np.arange(len(data)), labels=years)
        plt.xlabel('年份')
        plt.show()

    plot_acf_graph(a, f'{current_time}的 a 值')
    plot_acf_graph(b, f'{current_time}的 b 值')

    def predict_arima(series, order):
```

```
        model = ARIMA(series, order=order)
        model_fit = model.fit()
        forecast = model_fit.forecast(steps=1)
        return forecast[0], model_fit

    a_predicted, model_a = predict_arima(a, order=(1, 1, 1))
    b_predicted, model_b = predict_arima(b, order=(1, 1, 1))

    print(f'预测的今年的{current_time} 的 a 值: {a_predicted:.2f}')
    print(f'预测的今年的{current_time} 的 b 值: {b_predicted:.2f}')

    plt.figure(figsize=(12, 6))

    plt.subplot(1, 2, 1)
    plt.plot(years, a, label='历史数据')
    plt.plot(np.append(years, 2014 + len(a)), np.append(a, a_predicted), 'r--', label='预测值')
    plt.xlabel('年份')
    plt.ylabel('a 值')
    plt.title(f'{current_time} 的 a 值的 ARIMA 预测')
    plt.legend()

    plt.subplot(1, 2, 2)
    plt.plot(years, b, label='历史数据')
    plt.plot(np.append(years, 2014 + len(b)), np.append(b, b_predicted), 'r--', label='预测值')
    plt.xlabel('年份')
    plt.ylabel('b 值')
    plt.title(f'{current_time} 的 b 值的 ARIMA 预测')
    plt.legend()

    plt.tight_layout()
    plt.show()
```

```
90
91      return a_predicted,b_predicted
92
93  def solve(id,current_time):
94      time = []
95      temperatures = []
96      a = []
97      b = []
98      c = []
99      pa = []
100     pb = []
101
102     for i in range(0, len(df.columns), 3):
103         cols_all = df.columns[i: i + 3]
104         data_all = df[cols_all]
105         max_temps_all = data_all.loc['最高气温'].values
106         min_temps_all = data_all.loc['最低气温'].values
107         mean_temps_all = (max_temps_all.sum() + min_temps_all.
    sum()) / 6
108
109         cols = df.columns[i + id]
110         data = df[cols]
111         max_temps = data.loc['最高气温']
112         min_temps = data.loc['最低气温']
113         mean_temps = (max_temps + min_temps) / 2
114
115         time.extend([12, 13, 14, 20, 21, 22,
116                      0, 1, 2, 6, 7, 8])
117         temperatures.extend([
118             max_temps, max_temps, max_temps, mean_temps + 1,
    mean_temps, mean_temps - 1,
119             min_temps, min_temps, min_temps, mean_temps - 1,
    mean_temps, mean_temps + 1
120         ])
121         a.extend([mean_temps_all] * 12)
```

```
        b.extend([((max_temps - min_temps) / 2)] * 12)
        pa.extend([mean_temps_all])
        pb.extend([((max_temps - min_temps) / 2)])

    time = np.array(time)
    temperatures = np.array(temperatures)
    a = np.array(a)
    b = np.array(b)
    pa = np.array(pa)
    pb = np.array(pb)

    predict_a, predict_b = predict_ab(pa, pb, current_time)
    aa.append(predict_a)
    bb.append(predict_b)

    for i in range(len(a)):
        c.append(fit(temperature_model, time, temperatures, a[
    i], b[i]))

    c_mean = np.mean(c)
    #print(f'拟合得到的统一c值: {c_mean:.2f}')
    cc.append(c_mean)

def predict_pool_temperatures():
    predictions = {}
    for i, date_str in enumerate(dates):
        day_prediction = {}
        for time_label, time in time_points.items():
            temperature = temperature_model(time, cc[i], aa[i
    ], bb[i]) - 3
            if(temperature >= 35):temperature -= 3.5
            else:temperature -= 2
            day_prediction[time_label] = temperature
        predictions[date_str] = day_prediction
    return predictions
```

```
155
156 def simulate_chlorine_concentration(time_slots, id):
157     C0 = 0.6
158     chlorine_concentration = []
159     chlorine_times = []
160     current_time = time_slots[0][0]
161
162     for i, (start, end) in enumerate(time_slots):
163         if (i == 0 or i == 3 or i == 6 or i == 9):
164             chlorine_times.append(start)
165             C0 = 0.6
166         tt = 0.001
167         while current_time < end:
168             Tem = temperature_model(current_time, cc[id], aa[
    id], bb[id]) - 3
169             if Tem > 35:
170                 Tem -= 2
171             else:
172                 Tem -= 3.5
173             C = f(tt, Tem, C0)
174             # print(current_time, Tem)
175             chlorine_concentration.append((current_time, C))
176             current_time += tt
177             if i == 2 or i == 5 or i == 8:
178                 C0 = C
179             else:
180                 if C <= 0.3:
181                     chlorine_times.append(current_time)
182                     C0 = 0.6
183                 else:
184                     C0 = C
185
186     times, concentrations = zip(*chlorine_concentration)
187     plt.figure(figsize=(12, 6))
188     plt.plot(times, concentrations, label='余氯浓度 (mg/L)')
```

```
    plt.axhline(y=0.3, color='r', linestyle='--', label='阈值(0.3 mg/L)')
    plt.xlabel('时间 (小时)')
    plt.ylabel('余氯浓度 (mg/L)')
    plt.title(f'{dates[id]} 余氯浓度随时间变化')
    plt.legend()
    plt.grid(True)
    plt.show()
    print(f'{dates[id]} 加氯时间点: {chlorine_times}')

for i in range(0, 3):
    solve(i,dates[i])
    print(f"日期: {dates[i]}")
    print(f"predict_a = {aa[i]},predict_b = {bb[i]},predict_c = {cc[i]}")

predicted_temperatures = predict_pool_temperatures()

for date, temps in predicted_temperatures.items():
    print(f"日期: {date}")
    max_temp = max(temps.values())
    min_temp = min(temps.values())
    print(f"最高温度: {max_temp:.2f}°C")
    print(f"最低温度: {min_temp:.2f}°C")

    for time, temp in temps.items():
        print(f"{time}: {temp:.2f}°C")
    print()

for i in range(0, 3):
    simulate_chlorine_concentration(time_slots, i)
```

problem5.py

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

```python
import pandas as pd

df = pd.read_excel('xlsx2.xlsx')
non_maintenance_df = df.dropna(subset=['时间段'])
time_slots = [(int(time.split('-')[0].split(':')[0]), int(time.split('-')[1].split(':')[0]))
              for time in non_maintenance_df['时间段']]

A = 0.003648497349718996
B = 0.34939455820847437

def f(t, T, n, C0):
    return C0 * np.exp(-(A * n + B) * 10 ** ((T - 25) / 5) * t)
def temperature_model(t):
    predict_a = 26.118397177120908
    predict_b = 3.666764465966214
    predict_c = 0.4967543828609517
    return predict_a - predict_b * np.cos(np.pi / 12 * t - predict_c)

def simulate_chlorine_concentration(start, end, n, c, i):
    C0 = c
    chlorine_concentration = []
    chlorine_times = []
    current_time = start
    cnt = 0;
    tt = 0.001
    while current_time < end:
        T = temperature_model(current_time) - 3
        if T > 35:
            T -= 2
        else:
            T -= 3.5
        C0 = f(tt, T, n, C0)
```

```
        chlorine_concentration.append((current_time, C0))
        current_time += tt
        if i == 2 or i == 5 or i == 8:
            continue
        else:
            if C0 <= 0.3:
                chlorine_times.append(current_time)
                C0 = 0.6
                cnt = cnt + 1

    return cnt, chlorine_times, chlorine_concentration, C0;

c = 0.6
cc = 0
flag = 0
chlorine_times = []
chlorine_concentration = []

def display(chlorine_times, chlorine_concentration):
    times, concentrations = zip(*chlorine_concentration)
    plt.figure(figsize=(12, 6))
    plt.plot(times, concentrations, label='余氯浓度 (mg/L)')
    plt.axhline(y=0.3, color='r', linestyle='--', label='阈值
   (0.3 mg/L)')
    plt.xlabel('时间 (小时)')
    plt.ylabel('余氯浓度 (mg/L)')
    plt.title(f'{start} 余氯浓度随时间变化')
    plt.legend()
    plt.grid(True)
    plt.show()
    print(f'{start} 加氯时间点: {chlorine_times}')

for i, (start, end) in enumerate(time_slots):
    if i == 2 or i == 5 or i == 8:
        flag, chlorine_times, chlorine_concentration, cc =
```

```
    simulate_chlorine_concentration(start, end, 0, c, i)
69          print('最大人数 = 0')
70          #display(chlorine_times, chlorine_concentration)
71          continue;
72      if (i == 0 or i == 3 or i == 6 or i == 9):
73          c = 0.6
74      l = 0
75      r = 520
76      while l < r:
77          mid = int((l + r + 1) / 2)
78          flag , chlorine_times, chlorine_concentration ,cc =
    simulate_chlorine_concentration(start, end, mid ,c ,i)
79          if(flag <= 0):l = mid
80          else: r = mid - 1
81      c = cc
82      #display(chlorine_times, chlorine_concentration)
83      print(f'最大人数 = {l}')
```