泳池余氯浓度动态控制与加氯策略的综合研究

摘　要

在杭州电子科技大学游泳馆的水质管理中，余氯浓度的稳定控制对保障泳池水质安全至关重要。为了更好控制游泳馆运营中的余氯浓度，本文通过建立余氯浓度的指数衰减模型、线性回归模型、泳池余氯浓度动态控制模型、基于随机森林的气温预测模型、温差模型等模型解决池水再次加氯的时刻等问题。

对于问题一，基于余氯浓度的动态变化特性，我们假设余氯浓度随时间的变化遵循一级反应动力学，即余氯浓度的衰减速率与其当前浓度成比，建立了**余氯浓度的指数衰减模型**。首先，通过已知条件计算出衰减速率常数 k 为 0.462。接着，为了保证余氯浓度不得低于 0.05 **mg/L**，我们将最低余氯浓度带入指数衰减模型求解得再次加氯的时刻 t 为 5 **小时** 22 **分钟**后，即在第二天的 3 点 22 分需要再次加氯。

对于问题二，我们通过数据拟合建立了游泳人数与半小时后余氯浓度的关系模型。经过比较两者**线性回归模型**的拟合效果，相比于线性拟合函数、三次多项式拟合函数和指数拟合函数，二次多项式拟合函数的拟合结果更优，结果为 $V_2 = 0.481 + 0.0003N - 1.606 \times 10^{-6}N^2$。结合问题一中的余氯衰减，我们计算得出 255 人同时游泳时，余氯浓度首次降至安全阈值所需的时间，从而确定了首次加氯的时间点为 **29 min**。

对于问题三，基于游泳人数与余氯浓度的关系模型，以及余氯的指数衰减模型，我们建立了**余氯浓度的动态控制模型**。模型分为运营期和闭馆期两部分，通过计算游泳者活动对余氯衰减的影响，以及闭馆期间的自然衰减速度，确定了每个时段内何时需要加氯。然后，我们对全天的余氯浓度进行了逐分钟模拟，得出了 **19 个加氯时刻**，并绘制了从上午 9 点到晚上 21 点的余氯浓度变化曲线。

对于问题四，通过天气史网站收集近五年杭州在 9 月初的气温数据，我们利用**随机森林模型**预测了 2024 年 9 月 8 日至 10 日的气温变化，并通过 **Cubic Spline** 进行平滑处理后得出预测的气温曲线。接着，基于气温预测结果，建立了**池水温度与馆内气温的温差模型**，并构建了**余氯浓度的动态温度衰减模型**，假设水温越高，余氯衰减速度越快。通过模型，我们模拟了在不同温度条件下的余氯浓度变化，并据此调整加氯策略：2024 年 9 月 9 日上午 9 点到晚上 21 点的加氯次数为 **5 次**，加氯时刻分别为：**12:00，13:07，13:54，15:00，16:43**。

对于问题五，综合考虑水温和人员数量对余氯浓度的影响，为了优化泳池在不同开放时段内的入场人数管理，我们在前几问的基础上，建立了**余氯浓度的综合衰减模型**，结合游泳人数、水温和余氯浓度之间的关系，模拟了不同人数情况下的余氯变化过程。针对每个开放时段，在确保余氯浓度不低于 0.3 mg/L 的前提下，计算得出泳池在各个时段的最佳入场人数，第一场人数分别为 **458，450**；第二场人数为 **399，345**；第三场人数为 **422，447**；第四场人数为 **458，520**，520。

关键词:　**指数衰减 随机森林 温差模型　线性回归 余氯浓度**

一　问题的背景和重述

1.1　问题背景

在游泳池管理中，水质安全至关重要，特别是余氯浓度的控制。余氯浓度过低会导致水质不洁净，增加细菌和藻类的滋生风险；过高则可能刺激皮肤和黏膜。为了保障杭州电子科技大学游泳馆的水质安全，尤其是在开学季，需要通过合理的加氯措施，确保余氯浓度始终维持在 0.3-0.6mg/L 的安全范围内。这个过程涉及对泳池关闭维护、游泳者活动以及水温变化等多种因素下余氯浓度的动态管理。

1.2　问题重述

为了确保游泳池水质的安全性，避免对游泳者的健康造成潜在威胁，本文根据附件的数据建立数学模型，解决以下几个关键问题：

1. 在闭馆维护时，确定余氯从 0.6mg/L 降至 0.3mg/L 所需的时间，并确定何时需要再次加氯以保持水质不低于安全阈值。

2. 分析游泳者的活动对池水余氯浓度的影响，特别是 255 人同时游泳时，首次加氯的时间点。

3. 在游泳馆的四个营业时段中，确定加氯的次数，并绘制余氯浓度随时间变化的曲线。

4. 考虑到水温对余氯浓度的影响，根据气温预测来调整加氯策略，确保在不同温度条件下的水质安全。

5. 在综合考虑水温和人员数量对余氯浓度影响的情况下，优化入场人数的管理方案，确保在安全范围内最大化泳池使用率。

二　问题分析

2.1　问题一的分析

问题一要求在确定泳池闭馆维护期间，余氯浓度从 0.6mg/L 降至 0.3mg/L 所需的时间，以及在此之后何时需要再次加氯以保持水质安全。基于余氯浓度的动态变化特性，我们采用了指数衰减模型，假设余氯浓度随时间的变化遵循一级反应动力学，即余氯浓度的衰减速率与其当前浓度成正比。首先，通过已知条件计算出衰减速率常数 k，然后利用该模型预测余氯浓度降至安全阈值 0.05mg/L 所需的时间，从而确定再次加氯的最佳时机。

2.2　问题二的分析

问题二要求分析游泳者活动对泳池余氯浓度的影响，尤其是在 255 人同时游泳时首次加氯的时间点。余氯浓度不仅会自然衰减，还会因游泳者的活动而加速降低。为此，我们通过数据拟合建立了游泳人数与半小时后余氯浓度的关系模型。经过比较多种线性回归模型的拟合效果，最终选取了合适函数关系来描述游泳人数与余氯浓度的关系。结合问题一中的余氯衰减模型，计算得出 255 人同时游泳时，余氯浓度首次降至安全阈值所需的时间，从而确定了首次加氯的时间点。这一分析为泳池在高峰期的水质管理提供了科学依据，确保在大量游泳者活动时水质依然安全。

2.3　问题三的分析

问题三要求于确定泳池在不同营业时段内的加氯次数，并绘制余氯浓度随时间变化的曲线。泳池的营业时段从上午 9 点到晚上 21 点，期间余氯浓度会因游泳者的活动而降低，同时也会在闭馆维护期间自然衰减。为了确保水质安全，我们首先基于游泳人数与余氯浓度的关系模型，以及余氯的指数衰减模

型，建立了余氯浓度的动态控制模型。该模型分为运营期和闭馆期两部分，通过计算游泳者活动对余氯衰减的影响，以及闭馆期间的自然衰减速度，确定了每个时段内何时需要加氯。最后，我们对全天的余氯浓度进行了逐分钟模拟，得出了每个加氯时刻，并绘制了从上午 9 点到晚上 21 点的余氯浓度变化曲线。这一分析为泳池日常运营中的水质管理提供了科学的加氯策略，确保各时段内余氯浓度始终保持在安全范围内。

2.4 问题四的分析

问题四要求考虑水温对余氯浓度的影响，并根据未来的气温预测调整加氯策略，确保泳池在不同温度条件下的水质安全。首先，我们通过收集近五年杭州在 9 月初的气温数据，利用随机森林模型预测了 2024 年 9 月 8 日至 10 日的气温变化。接着，基于气温预测结果，建立了池水温度与馆内气温的温差模型，并构建了余氯浓度的动态温度衰减模型，假设水温越高，余氯衰减速度越快。通过这些模型，我们模拟了在不同温度条件下的余氯浓度变化，并据此调整加氯策略，以确保在未来三天内，泳池的水质能够在各个时段内保持在安全范围内。这一分析为在气温变化条件下的泳池水质管理提供了科学依据，确保在不同时段的水质安全。

2.5 问题五的分析

问题五要求综合考虑水温和人员数量对余氯浓度的影响，优化泳池在不同开放时段内的入场人数管理，以确保水质安全的前提下最大化泳池的使用率。我们首先在前几问的基础上，建立了余氯浓度的综合衰减模型，结合游泳人数、水温和余氯浓度之间的关系，模拟了不同人数情况下的余氯变化过程。然后，针对每个开放时段，考虑在满足水质安全（余氯浓度不低于 0.3mg/L）的前提下，计算得出泳池在各个时段的最佳入场人数。通过这一优化模型，我们提出了不同时段的入场人数控制方案，以确保泳池在高效利用的同时，保持水质始终在安全范围内。这一分析为泳池运营提供了优化管理策略，既保障了水质安全，又提高了泳池的使用效率。

三 模型假设

1. 余氯浓度符合一级反应动力学。
2. 泳池水温与馆内气温存在稳定的温差关系。
3. 假设近五年杭州气温与杭电校园气温一致。
4. 假设忽略加氯所需的时间。

四　符号说明

符号	含义
$C(t)$	时间 t 时的余氯浓度（mg/L）
C_0	初始余氯浓度（mg/L）
k	余氯衰减速率常数
T	水温（摄氏度）
N	游泳人数
t	时间（小时）
C_1	半小时后的余氯浓度（mg/L）
α	标准化常数，$\alpha = \frac{1}{\ln k}$
Δt	水温与气温的温差（摄氏度）
$f(N)$	游泳人数与余氯衰减速率的函数关系

五　模型的建立与求解

5.1　问题一模型的建立与求解

5.1.1　余氯浓度的指数衰减模型

对于确定池水需要再次加氯的时刻，参考余氯衰减相关研究 [1]，余氯衰减过程具有指数衰减的特性，因此我们假设余氯浓度符合一级反应动力学，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，余氯数学模型为微分方程：

$$\frac{d[C]}{dt} = -K[C] \tag{1}$$

其中，t 为时间，$x = x(t)$ 为 t 时刻的余氯含量（浓度），一阶导数 $\frac{dx}{dt}$ 是反应速率，比例系数 k 是反应速率常数，$k > 0$，且不随着余氯含量（浓度）的变化而变化，符号 $-$ 表示余氯含量（浓度）在减少。

其特解方程，初始时刻为 t_0，浓度为 x_0：

$$x(t) = x_0 e^{-K(t-t_0)} \tag{2}$$

余氯浓度随时间 t 的变化服从指数衰减模型, 公式如下：

$$C(t) = C_0 \cdot e^{-kt} \tag{3}$$

其中，$C(t)$ 是时间 t 的余氯浓度（mg/L），C_0 是初始余氯浓度（mg/L），k 是衰减系数。

5.1.2　模型求解

根据问题一的要求我们对余氯浓度的指数衰减模型进行求解，具体如下：

1. 计算衰减系数 k

已知初始余氯浓度 0.6 mg/L 降到 0.3 mg/L 平均用时为 1.5 小时，根据公式：

$$k = -\frac{\ln(C_1/C_0)}{1.5} \tag{4}$$

其中，$C_1 = C_0 \cdot e^{-k \cdot 1.5}$，可以解得衰减系数为：$k = 0.462$。

2. 计算再次加氯时间

为了保证池水余氯浓度不低于 0.05 mg/L，我们需要计算时间 t，使得：

$$C(t) = 0.05 \tag{5}$$

带入指数衰减模型得：

$$0.05 = 0.6 \cdot e^{-kt} \tag{6}$$

解得：

$$t = -\frac{\ln(0.05/0.6)}{k} \tag{7}$$

求解得再次加氯的时刻 t 为 5 小时 22 分钟后，即在第二天的 3 点 22 分需要再次加氯。

5.2 问题二模型的建立与求解

5.2.1 模型建立

对于人数对余氯浓度的影响，我们将常温下游泳人数与 0.5 小时后余氯值的数据可视化，结果如下：

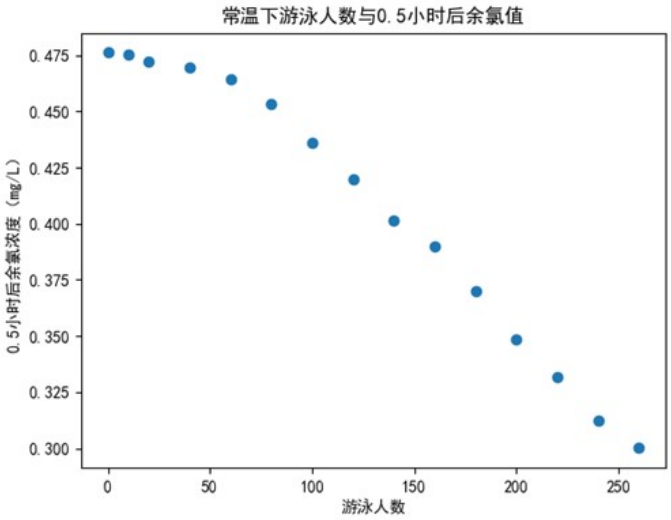

图 1: 常温下游泳人数与 0.5 小时后余氯值

由图观察可得, 常温下游泳人数与 0.5h 后余氯浓度呈多元线性关系，对此我们分别建立线性回归模

型，二次多项式线性模型，三次多项式线性模型及指数模型对数据进行拟合，通过对比 R2 值来寻求最优的模型，最终建立游泳人数与半小时余氯浓度的关系的数学模型。

根据问题一的余氯指数衰减模型, 通过游泳人数与半小时余氯浓度的关系的数学模型求解余氯浓度, 求解得泳池常温下 255 人游泳（初始余氯 0.6mg/L）的第一次加氯时间。

5.2.2 线性回归模型

为了探讨游泳人数对 0.5 小时后余氯浓度的影响，我们使用线性回归模型 [2] 进行数据拟合, 通过建立自变量（游泳人数）与因变量（0.5 小时后余氯浓度）之间的线性关系.

假设游泳人数为 x，0.5 小时后余氯浓度为 y。公式如下：

$$y = \beta_0 + \beta_1 x + \epsilon \tag{8}$$

其中，y 是因变量，即 0.5 小时后余氯浓度,x 是自变量，即游泳人数,β_0 是模型的截距项,β_1 是模型的斜率项，反映了游泳人数对余氯浓度的影响程度,ϵ 是误差项。

通过最小二乘法（OLS）对模型参数 β_0 和 β_1 进行估计，得到的估计值分别为 $\hat{\beta_0}$ 和 $\hat{\beta_1}$。在完成参数估计后，可以通过以下方程进行预测：

$$\hat{y} = \hat{\beta_0} + \hat{\beta_1} x \tag{9}$$

为了评估模型的拟合效果，我们计算决定系数 R^2 ，其公式如下：

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y_i})^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \tag{10}$$

其中，y_i 是观测值,$\hat{y_i}$ 是预测值,$\bar{y}$ 是观测值的平均值,n 是观测值的数量。决定系数 R^2 的取值范围为 0 到 1，值越接近 1，说明模型的拟合效果越好。

5.2.3 模型求解

对于游泳人数与半小时余氯浓度的关系, 我们利用 SPSS 分别求出游泳人数与半小时余氯浓度的一次、二次、三次函数关系和指数函数关系, 以一次函数关系为例, 原理如下:

$$V_2 = \mu N + \sigma \tag{11}$$

其中，μ 和 σ 是线性关系的相关参数。通过最小二乘法（OLS）可以估计出这些参数的值。

μ 和 σ 的取值可以通过以下公式计算得出：

$$\mu = \frac{\sum_{i=1}^{n}(N - \bar{N})(V_{2i} - \bar{V_2})}{\sum_{i=1}^{n}(N - \bar{N})^2} \tag{12}$$

$$\sigma = \bar{V_2} - \mu \bar{N} \tag{13}$$

具体结果如下:

- 线性拟合函数：

$$V_2 = -0.495 - 0.000716N \tag{14}$$

- 二次多项式拟合函数：

$$V_2 = 0.481 + 0.0003N - 1.606 \times 10^{-6}N^2 \tag{15}$$

- 三次多项式拟合函数：

$$V_2 = 0.475 + 9.207 \times 10^{-5}N - 5.747 \times 10^{-6}N^2 + 1.073 \times 10^{-8}N^3 \tag{16}$$

- 指数拟合函数：

$$V_2 = 0.504 \times e^{-0.002N} \tag{17}$$

四种函数的拟合优度分别为 1,1,1,0.996,F 值为 0,0,0,0.000026，拟合曲线如下所示：

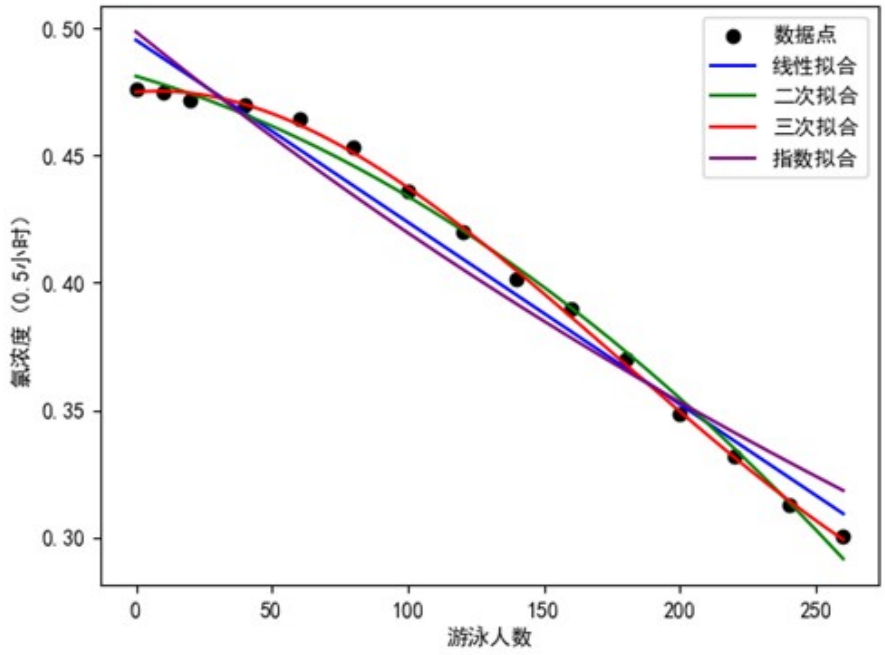

图 2: 游泳人数与半小时后余氯浓度的关系

其中三次的 R 方最大，拟合效果最好，但是三次在 357 人数左右出现拐点，人数超过 352 人余氯值随游泳人数的增加而增加，显然不符合实际情况。拟合曲线如下所示：

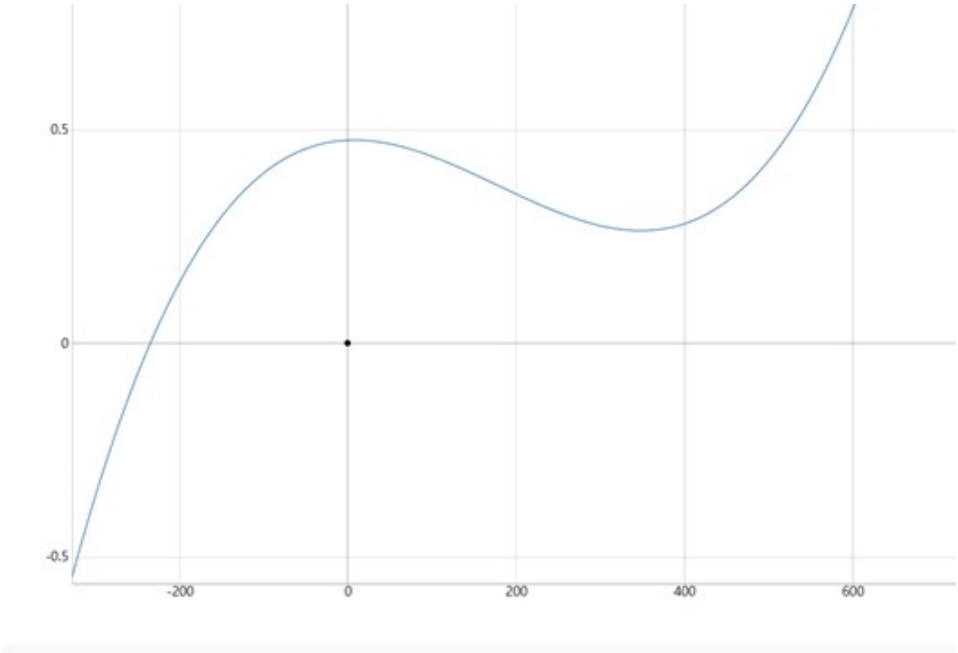

图 3: 三次函数关系拟合曲线

而 R 方值次大的二次拟合的图像没有拐点，在大于 0 时为单独递减函数，因此该曲线是符合实际情况的。结果如下所示：

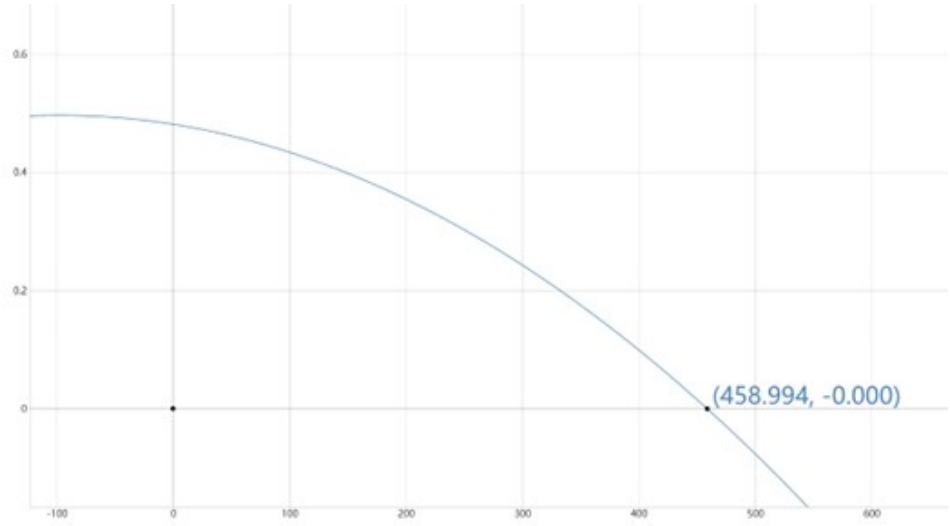

图 4: 二次函数关系拟合曲线

因此我们最终选取二次函数关系，游泳人数与半小时余氯浓度的关系的数学模型为：

$$V_2 = 0.481 + 0.0003N - 1.606 \times 10^{-6} N^2 \tag{18}$$

根据拟合出的游泳人数与余氯浓度关系模型，带入数据得出在常温下 255 人游泳（初始余氯浓度为 0.6 mg/L）0.5 小时后的余氯浓度值为 0.297 mg/L，基于这个余氯值，利用问题一的指数衰减模型求出

衰减系数 k 的值：

$$k = 1.406$$

接着，利用问题一的解法将数据带入指数衰减模型，解出时间 t 为：

$$t = 29min$$

5.2.4 模型检验

R^2 是线性回归的拟合优度或决定系数，反映了回归模型拟合数据的优良程度：R^2 越接近 1，拟合效果越好。F 值是估计回归总显著性的一个度量，也是调整后的 R^2 的一个显著性检验指标，用于验证模型整体显著性水平：F 值越大，越拒绝零假设，意味着模型整体显著性水平越高。

对于二次回归模型，R^2 的值为 0.990，F 值为 0，表明该模型的拟合度与显著性都非常优秀。

5.3 问题三模型的建立与求解

5.3.1 泳池余氯浓度动态控制模型

对于问题三提出的求解 9 点到 21 点这段时间内的加氯次数，我们在问题二的基础上建立泳池余氯浓度动态控制模型，对运营期和闭馆期分别建立余氯浓度的衰减模型，根据游泳者数量对氯浓度的影响求解余氯衰减系数，最终设定阈值确定加氯时刻，具体模型建立步骤如下：

1. 运营期与闭馆期的余氯浓度衰减模型

在游泳池的运营期，假设余氯浓度随时间 t 的变化符合指数衰减模型。余氯的衰减系数 k_p 与游泳人数 N 有关。由此，我们建立运营期的余氯浓度衰减模型如下：

$$C(t) = C(t_0) \cdot e^{-k_p(t-t_0)}, \tag{19}$$

其中 $C(t_0)$ 为起始时间 t_0 时的余氯浓度，k_p 为游泳人数为 N 时的余氯衰减系数。

在闭馆期，同样假设余氯浓度随时间 t 的变化符合指数衰减模型，闭馆期的余氯衰减系数 k_0 在第一问中已经求出。由此可得闭馆期的余氯浓度衰减模型为：

$$C(t) = C(t_0) \cdot e^{-k_0(t-t_0)}, \tag{20}$$

其中 $C(t_0)$ 为闭馆期起始时间 t_0 时的余氯浓度，k_0 为闭馆期的余氯衰减系数。

2. 游泳者对余氯浓度的影响

游泳者的数量 N 会加速氯的消耗。通过第二问得出氯的衰减速率 k_p 与游泳者数量的函数关系 $f(N)$。设 $N(t)$ 是时间 t 时的游泳者数量，则氯的有效衰减系数可以表示为：

$$C_1 = f(N) \tag{21}$$

$$k_p = -\frac{\ln(C_1/C_0)}{t_1} = -\frac{\ln(C_1/0.6)}{0.5}, \tag{22}$$

其中 C_0 是初始的余氯浓度，k_p 为大量人数 N 时的衰减系数。

3. 加氯操作策略

在游泳池运营期间（9:00-21:00），如果氯浓度 $C(t)$ 低于运营加氯阈值 thr_1 （即 0.3 mg/L），我们立即进行加氯操作。每次加氯后，氯浓度立即恢复到 0.6 mg/L。

在闭馆维护期间（11:00-12:00，14:00-15:00，17:00-18:00），如果氯浓度低于维护加氯阈值 thr_2 （即 0.05 mg/L），也需要进行加氯操作，使氯浓度恢复到 0.6 mg/L。

5.3.2 模型求解

根据附件 2 的数据，我们使用泳池余氯浓度动态控制模型对加氯时刻进行求解，以 9:00 至 10:00 时间段为例，游泳人数 $N = 20$。我们利用函数 $V_2(N)$ 计算出此时的衰减系数 k，并根据公式对每分钟的余氯浓度进行更新。通过逐分钟计算，发现当时间到达 10:22 时，余氯浓度首次下降至运营加氯阈值 $\text{thr}_1 = 0.3\,\text{mg/L}$ 以下：

$$C(10{:}22) < 0.3\,\text{mg/L} \tag{23}$$

因此，系统在 10:22 触发了首次加氯操作，将余氯浓度恢复到 0.6 mg/L。此时，加氯时刻被记录为 10:22。完整结果如下表所示：

表 1: 加氯时刻

加氯时刻	加氯时刻	加氯时刻
10:22	12:00	12:51
13:36	15:00	15:38
16:07	16:26	16:45
18:00	18:26	18:52
19:06	19:16	19:26
19:36	19:46	19:56
20:28		

此外，我们求得了 9 点到 21 点时段泳池中余氯浓度值变化曲线，结果如下：

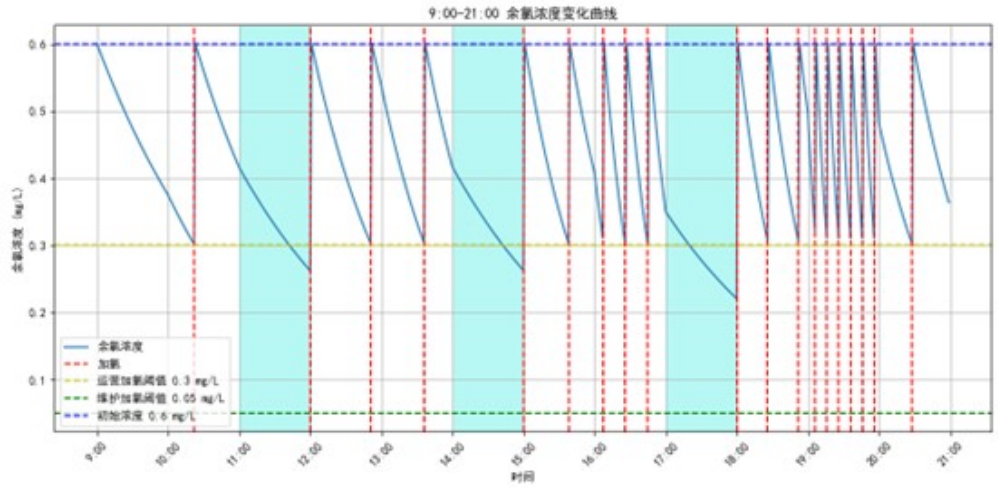

图 5: 9 点到 21 点时段泳池中余氯浓度值变化曲线

5.3.3 模型检验

为了验证所建立的泳池余氯浓度动态控制模型的合理性和准确性，我们将模型预测的余氯浓度与各时间段的衰减系数进行了对比分析。结果表明，模型预测的余氯浓度在不同时间段内与衰减系数的变化趋势相吻合，模型能够准确反映余氯浓度随时间的衰减情况。具体的检验结果如下表所示：

表 2: 0.5 小时后余氯浓度和衰减系数 k 对比

时间段	衰减系数 k	0.5 小时后余氯浓度 (mg/L)
9:00-10:00	0.469904	0.474365
10:00-11:00	0.593254	0.445993
11:00-12:00	0.462098	0.476220
12:00-13:00	0.819551	0.398280
13:00-14:00	0.950628	0.373014
14:00-15:00	0.462098	0.476220
15:00-16:00	1.107594	0.344858
16:00-17:00	2.305603	0.189451
17:00-18:00	0.462098	0.476220
18:00-19:00	1.612475	0.267921
19:00-20:00	4.389272	0.066839
20:00-21:00	0.999828	0.363950

5.4 问题四模型的建立与求解

5.4.1 模型建立

问题四要求预测出 2024 年 9 月 8 日-10 日杭电校园气温和馆内气温，因此我们通过收集天气史网站 [3] 提供的近五年 9 月 8 日-10 日每小时杭州气温的数据，使用随机森林回归 [4] 对 2024 年 9 月 8-10 日的气温进行预测，使用 Cubic Spline [5] 进行平滑处理后得出预测的气温曲线。根据获得的气温曲线，建立池中水温与馆内气温之间的温差模型获得 2024 年 9 月 8 日-10 日池水温度曲线。

5.4.2 基于随机森林的气温预测模型

根据收集到的近五年（2019 年-2023 年）9 月 8 日-10 日每小时杭州气温的数据，我们使用随机森林模型对 2024 年 9 月 8 日-10 日的气温进行预测。

随机森林作为一种基于决策树的机器学习算法，广泛应用于分类和回归任务。其核心思想是通过构建多个相互独立的决策树，并将这些决策树的预测结果结合起来，从而提升模型的准确性和鲁棒性。随机森林模型的建模过程主要包括以下步骤：

1. 自助采样：

从原始数据集中使用自助采样法（Bootstrap Sampling）抽取样本，形成多个子数据集。假设我们有 $m = 24 \times 3 \times 5$ 条数据（即 5 年 9 月 8 日-10 日，每天 24 小时的数据），我们通过自助采样法生成 n 个子数据集，每个子数据集的大小与原始数据集相同。

2. 构建决策树：

对每个子数据集，构建一棵决策树。在每个节点处，随机选择部分特征进行分裂。当每个样本有 M 个特征（如时间、前几小时的气温等）时，在决策树的每个节点需要分裂时，随机从这 M 个特征中选

取 m 个特征，满足条件 $m \ll M$。然后从这 m 个特征中采用如基尼指数、信息增益等策略来选择一个特征作为该节点的分裂属性。

3. 重复分裂：

决策树形成过程中，每个节点都按照上述方式进行分裂，直到达到预设的停止条件（如最小节点样本数或最大树深度）为止。注意在整个决策树形成过程中，没有进行剪枝操作。

4. 生成随机森林：

重复上述步骤，直到生成 N 棵决策树，构成最终的随机森林。通常 N 的取值较大，以确保模型的稳定性和泛化能力。

最终的预测模型表示为：

$$\hat{T}(X) = \frac{1}{N} \sum_{i=1}^{N} T_i(X) \tag{24}$$

其中，$T_i(X)$ 是第 i 棵决策树对输入特征 X（如时间和前几小时的气温）的预测值，N 是决策树的数量，$\hat{T}(X)$ 是通过随机森林预测的气温。

对于预测后的气温数据，我们利用三次样条插值（Cubic Spline Interpolation）进行平滑处理，以生成连续的气温预测曲线。三次样条插值是一种常用于数据平滑和插值的方法，它通过在每个分段上构建三次多项式函数，并使得这些多项式函数在区间节点处具有相同的一阶和二阶导数，从而生成一条平滑且连续的曲线。具体而言，我们对每小时的预测气温进行插值，以生成整个时间段内的平滑气温变化曲线。

通过以上步骤，我们可以获得 2024 年 9 月 8 日-10 日杭州的气温预测结果，并且这些结果能够更好地反映气温的实际变化趋势。

5.4.3 池中水温与馆内气温之间的温差模型

对于游泳馆的池水温度变化规律, 我们建立池中水温与馆内气温之间的温差模型，具体如下：

1. 当馆内气温 $t < 35°\mathrm{C}$ 时，假设水温比馆内气温低 $2°\mathrm{C}$，即：

$$\Delta t = 2°\mathrm{C} \tag{25}$$

2. 当馆内气温 $35°\mathrm{C} \leq t \leq 40°\mathrm{C}$ 时，水温与馆内气温的温差与馆内气温呈线性关系，线性模型为：

$$\Delta t = 3.5°\mathrm{C} + 0.1(t - 35°\mathrm{C}) \tag{26}$$

3. 当馆内气温 $t > 40°\mathrm{C}$ 时，假设水温比馆内气温低 $4°\mathrm{C}$，即：

$$\Delta t = 4°\mathrm{C} \tag{27}$$

综上所述，温差模型可以用分段函数表示为：

$$\Delta t = \begin{cases} 2°\mathrm{C}, & t < 35°\mathrm{C} \\ 3.5°\mathrm{C} + 0.1(t - 35°\mathrm{C}), & 35°\mathrm{C} \leq t \leq 40°\mathrm{C} \\ 4°\mathrm{C}, & t > 40°\mathrm{C} \end{cases} \tag{28}$$

其中，Δt 为馆内气温与水温的温差，t 为馆内气温。

5.4.4 游泳池余氯浓度的动态温度衰减模型

在游泳池中，余氯（用于消毒）会随着时间的推移逐渐减少，而这一过程受到多种因素的影响，其中水温是一个关键因素。通常情况下，水温越高，余氯分解的速度越快。为了研究水温对余氯浓度变化的影响，我们建立了游泳池余氯浓度的动态温度衰减模型，具体如下：

1. 模型建立

假设时间为 t 时，余氯浓度为 $C(t)$。在给定水温 $T_{水}$ 下，余氯浓度的变化率 $\frac{dC(t)}{dt}$ 与浓度 $C(t)$ 和衰减速率 $r(T_{水})$ 的乘积成正比：

$$\frac{dC(t)}{dt} = -r(T_{水}) \cdot C(t) \tag{29}$$

其中，余氯浓度衰减速率 $r(T_{水})$ 与水温 $T_{水}$ 的关系为：

$$r(T_{水}) = k \cdot 10^{\frac{T_{水}-25}{5}} \tag{30}$$

上式中，常数 k 表示在基准水温 $25°C$ 下的衰减速率，单位为 小时$^{-1}$；$T_{水}$ 表示当前的水温，单位为摄氏度（°C）。

2. 构建微分方程

余氯浓度随时间的变化可以表示为以下微分方程：

$$\frac{dC(t)}{dt} = -k \cdot 10^{\frac{T_{水}-25}{5}} \cdot C(t) \tag{31}$$

这是一个一阶线性微分方程，使用分离变量法解此方程：

$$\frac{dC(t)}{C(t)} = -k \cdot 10^{\frac{T_{水}-25}{5}} \cdot dt \tag{32}$$

两边积分得到：

$$\int \frac{1}{C(t)} dC(t) = -k \cdot 10^{\frac{T_{水}-25}{5}} \cdot \int dt \tag{33}$$

对左边积分，得到 $\ln C(t)$；对右边积分，得到 $-k \cdot 10^{\frac{T_{水}-25}{5}} \cdot t + C_1$：

$$\ln C(t) = -k \cdot 10^{\frac{T_{水}-25}{5}} \cdot t + C_1 \tag{34}$$

通过指数化简，得到余氯浓度随时间的变化公式：

$$C(t) = C_0 \cdot e^{-k \cdot 10^{\frac{T_{水}-25}{5}} \cdot t} \tag{35}$$

其中，$C_0 = e^{C_1}$ 是初始余氯浓度。

3. 常数 k 的确定

假设在水温 $T_{水} = 25°C$ 时，余氯从 0.6 mg/L 降到 0.3 mg/L 用时 1.5 小时。将已知条件代入模型公式：

$$0.3 = 0.6 \cdot e^{-k \cdot 1.5} \tag{36}$$

解此方程可得：

$$k = \frac{\ln(2)}{1.5} \approx 0.462 \text{ 小时}^{-1} \tag{37}$$

基于以上推导，水温 $T_{水}$ 下的余氯衰减速率为：

$$r(T_{水}) = k \cdot 10^{\frac{T_{水}-25}{5}} \approx 0.462 \cdot 10^{\frac{T_{水}-25}{5}} \text{ 小时}^{-1} \tag{38}$$

余氯浓度随时间变化的最终表达式为：

$$C(t) = C_0 \cdot e^{-k \cdot 10^{\frac{T_{水}-25}{5}} \cdot t} \tag{39}$$

5.4.5 模型求解

题目假设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℃。因此认为馆内气温 T_{in} 比室外温度 T_{out} 低 3 度，即：

$$T_{\text{in}} = T_{\text{out}} - 3\,^{\circ}\text{C} \tag{40}$$

通过基于随机森林的气温预测模型，我们求的 2024 年 9 月 8 日-10 日杭电校园气温和馆内气温的预测结果如下：

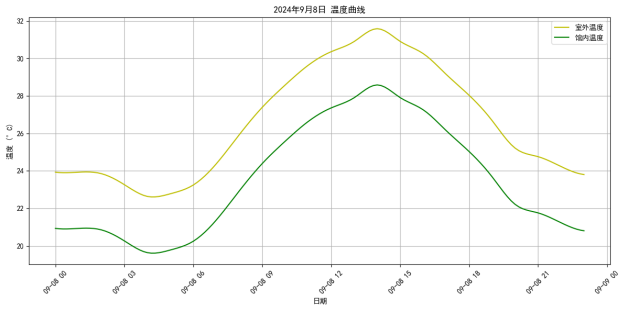

图 6: 2024 年 9 月 8 日杭电校园气温和馆内气温

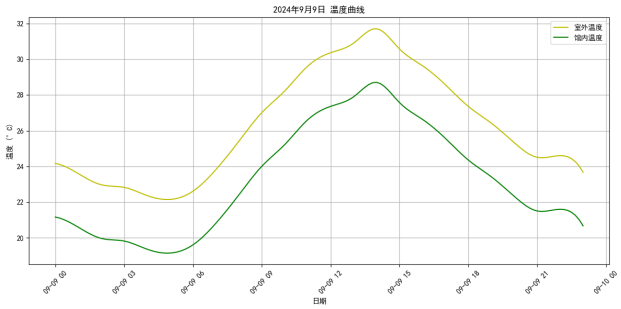

图 7: 2024 年 9 月 9 日杭电校园气温和馆内气温

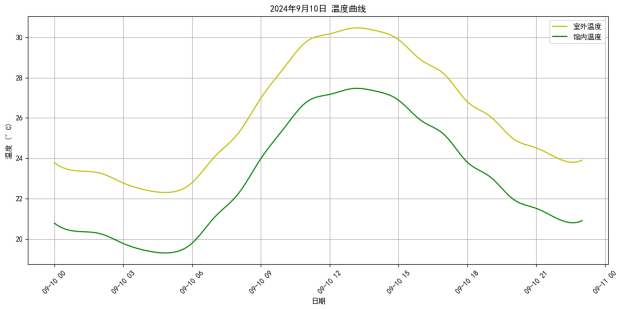

图 8: 2024 年 9 月 10 日杭电校园气温和馆内气温

此外，我们根据温差模型与馆内 72h 气温预测得池中水温预测曲线如下：

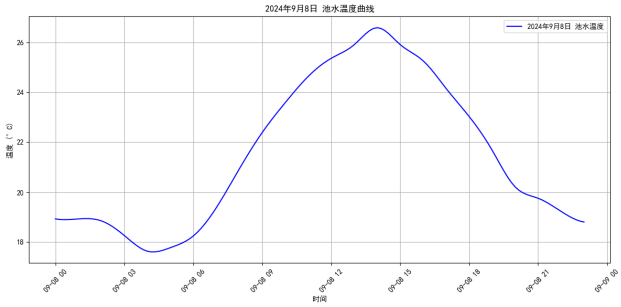

图 9: 2024 年 9 月 8 日池水温度曲线

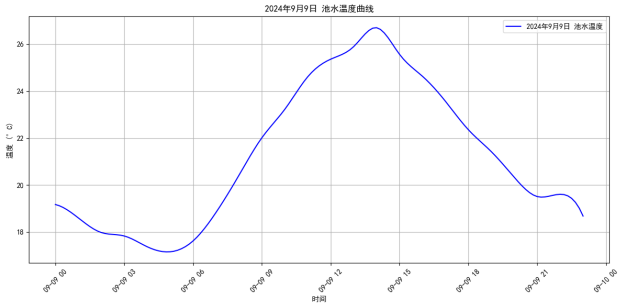

图 10: 2024 年 9 月 9 日池水温度曲线

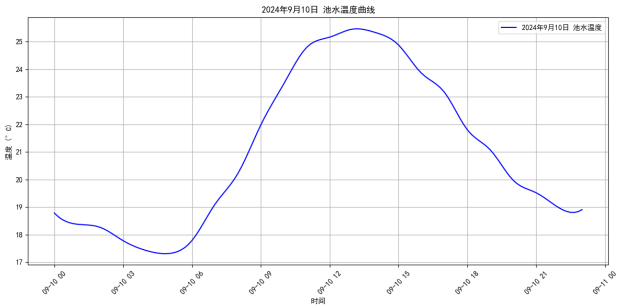

图 11: 2024 年 9 月 10 日池水温度曲线

预测得到的每天池水最高温度和最低温度，以及 10 点，12 点，14 点，16 点和 20 点的池水温度值结果如下：

表 3: 杭电游泳馆三天相应的池水温度预测值

时间	2024 年 9 月 8 日	2024 年 9 月 9 日	2024 年 9 月 10 日
最高温度	26.58℃	26.70℃	25.47℃
最低温度	17.61℃	17.15℃	17.31℃
上午 10:00	23.58℃	23.22℃	23.47℃
中午 12:00	25.36℃	25.36℃	25.17℃
下午 14:00	26.58℃	26.70℃	25.33℃
下午 16:00	25.26℃	24.63℃	23.86℃
晚上 20:00	20.21℃	20.33℃	19.97℃

对于问题四的第二个任务，为简化计算，假设池水温度在每个 1 小时的时间段内基本保持不变。因此，我们将每天从 9:00 到 21:00 划分为 12 个时间段，并取各时间段的中间时间作为该时间段的池水温度。由此，我们得到池水温度在 8 日、9 日和 10 日各时间段的温度如下：

表 4: 池水 9 月 8 日-10 日温度（单位：摄氏度）

时间段	8 日温度	9 日温度	10 日温度
9:00-10:00	23.01	22.62	22.76
10:00-11:00	24.13	23.94	24.22
11:00-12:00	25.05	25.08	25.06
12:00-13:00	25.60	25.56	25.33
13:00-14:00	26.34	26.43	25.45
14:00-15:00	26.35	26.28	25.18
15:00-16:00	25.69	25.05	24.37
16:00-17:00	24.75	24.15	23.55
17:00-18:00	23.59	22.94	22.49
18:00-19:00	22.41	21.88	21.42
19:00-20:00	20.87	20.89	20.52
20:00-21:00	19.91	19.81	19.69

假定早上 9 点池中余氯浓度 0.6mg/L，该时段泳池中余氯浓度值变化曲线以及加氯操作如下

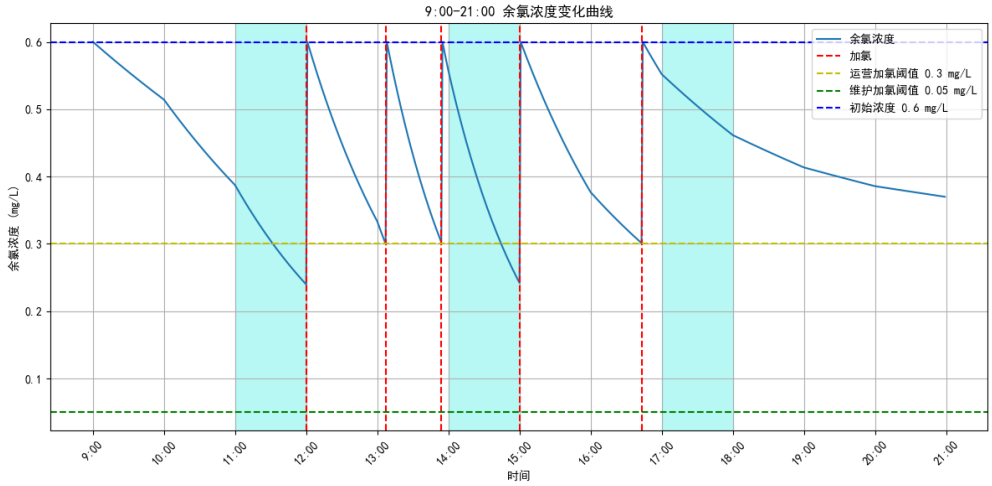

图 12: 余氯浓度值变化曲线以及加氯操作

由图可得 2024 年 9 月 9 日上午 9 点到晚上 21 点的加氯次数为 5 次, 加氯时刻分别为：12:00, 13:07, 13:54, 15:00, 16:43。

5.5 问题五模型的建立与求解

5.5.1 余氯浓度综合衰减模型

为了给出 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，在前几问的基础上我们建立余氯浓度综合衰减模型，具体如下：

1. 确定余氯浓度变化

考虑到泳池水温和游泳人数的影响，余氯浓度随时间的变化可以表示为：

$$C(t-t_N) = C_0 \cdot e^{-k_p N \cdot 10 \cdot \frac{T_k-25}{5} \cdot t} \tag{41}$$

其中，t_N 为第 n 次加氯的时间，t_0 的初始值为起始时间，k_{pN} 表示当人数为 N 且基准水温为 25°C 时的衰减速率。

2. 考虑游泳者的影响

游泳者的数量会加速余氯的消耗。设 $N(t)$ 为时间 t 时的游泳者数量，氯气的有效衰减系数 k_p 可以表示为：

$$C_1 = f(N) \tag{42}$$

$$k_p = \frac{\ln\left(\frac{C_1}{C_0}\right)}{t_1} = \frac{-\ln\left(\frac{C_1}{0.6}\right)}{0.5} \tag{43}$$

其中 C_0 是初始余氯浓度，k_p 表示当人数为 N 时的衰减速率。

3. 约束条件

a. 为保证泳池水质始终安全（控制在 0.3-0.6 mg/L 范围内），当余氯浓度降至阈值时需要进行加氯，将余氯浓度加至 0.6 mg/L。假设加氯时间可以忽略不计。

$$C(t) = 0.6 \quad n = n + 1 \quad \text{if} \quad C(t) = 0.3 \tag{44}$$

b. 对于两个游泳开放场次间的闭馆 1 小时内，泳池不需加氯。但为保证水质不变质，闭馆期间余氯浓度不应低于 0.05 mg/L。此时，闭馆开始时间为 t_{close}。

$$C(t_{\text{close}}) = C_0 \cdot e^{-k_{pN} \cdot 10^{\frac{T_k-25}{5}} \cdot (t_{\text{close}} - t_n)} \tag{45}$$

$$C(t_{\text{close}} + 1) = C(t_{\text{close}}) \cdot e^{-k_{pN} \cdot 10^{\frac{T_k-25}{5}}} \tag{46}$$

c. 游泳期间两次“加氯”间隔不应低于 10 分钟：

$$t_n - t_{n-1} \geq \frac{1}{6} \quad \forall n \tag{47}$$

d. 根据规定，泳池人均需 3.5 平方米空间，因此杭电游泳馆池面饱和人数为 520 人：

$$N \leq 520 \tag{48}$$

5.5.2 模型求解

根据游泳池余氯浓度的衰减模型及游泳人数、池水温度对余氯浓度的影响，制定了不同时段的入场人数控制方案。余氯是游泳池水质管理中的关键因素，过低的余氯浓度可能导致水质恶化，影响游泳者的健康；而过高的余氯浓度又会造成对人体皮肤和呼吸道的刺激。因此，合理控制游泳人数可以帮助保持池水中的余氯浓度在安全范围内。

我们运用了二分法来确定每个时间段内适宜的入场人数，以确保游泳池内的余氯浓度维持在合理的范围内。本模型中，目标是找到每个时段内的游泳人数 $n(t)$，使得游泳池内的余氯浓度 $C(t)$ 维持在设定的安全区间内。

首先，我们确定了每个时段的初始区间，假设游泳池在某一时段内的最小入场人数为 0，最大入场人数为 520。因此，初始区间为 $[n_{\min}, n_{\max}] = [0, 520]$。在每一次迭代中，我们计算当前区间的中点人

数 n_{mid}，公式如下：

$$n_{\text{mid}} = \frac{n_{\min} + n_{\max}}{2} \tag{49}$$

然后，将 n_{mid} 代入模型中，计算该人数下的余氯浓度 $C(n_{\text{mid}})$。根据 $C(n_{\text{mid}})$ 与目标浓度 C_{target} 的比较结果，来决定更新区间的方式。

如果 $C(n_{\text{mid}}) > C_{\text{target}}$，说明当前人数过多，需要减少人数，因此更新区间为 $[n_{\min}, n_{\text{mid}}]$。若 $C(n_{\text{mid}}) < C_{\text{target}}$，则说明当前人数过少，需要增加人数，更新区间为 $[n_{\text{mid}}, n_{\max}]$。这个过程会一直持续，直到区间的宽度 $|n_{\max} - n_{\min}|$ 小于设定的精度阈值 ϵ，此时，n_{mid} 即为满足条件的最优入场人数。

通过上述二分法，我们计算得出了 9 月 9 日不同时段的游泳人数控制方案。最终结果如下表所示:

表 5: 池水 9 月 9 日入场人数控制方案

时间段	人数	时间段	人数
9:00-10:00	458	15:00-16:00	422
10:00-11:00	450	16:00-17:00	447
12:00-13:00	399	18:00-19:00	458
13:00-14:00	345	19:00-20:00	520
		20:00-21:00	520

不同时段的最优入场人数分布，能够有效控制游泳池内的余氯浓度在安全范围内，保障游泳者的健康安全。

六　模型的优缺点及推广

6.1　模型的优点

1. 模型基于余氯浓度的一级反应动力学、温度与衰减速率的关系等科学原理，确保了分析的准确性和可靠性。特别是在余氯浓度的动态变化模拟中，模型能精确预测余氯浓度随时间的衰减情况。

2. 模型不仅适用于游泳池闭馆期间的余氯浓度管理，还能扩展应用到泳池的日常运营中，通过合理调整加氯策略，保障水质安全。

3. 结合气温预测和水温模型，模型能够在不同温度条件下动态调整加氯策略，从而适应气温变化，确保在高温或低温条件下水质仍保持在安全范围内。

4. 模型同时考虑了游泳人数、水温等多个因素对余氯浓度的影响，通过综合衰减模型有效模拟实际运营中的复杂情况，使得模型结果更符合实际操作需求。

5. 通过余氯浓度变化的曲线图，直观展示了泳池在不同时段的水质变化情况，并给出具体的加氯时间点，使得管理者能够轻松制定并执行加氯计划。

6.2　模型的缺点

1. 模型高度依赖于输入数据的准确性和完整性，特别是气温、水温、游泳人数等数据。如果这些数据存在误差或不准确，可能会导致模型预测结果偏差较大。

2. 模型假设加氯过程是瞬时完成的，并且忽略了加氯设备的响应时间和实际操作中的不确定性，这可能导致实际操作与模型预测存在差异。

3. 模型中涉及的多个参数（如衰减速率常数 k、温度衰减系数等）通常基于历史数据或理论推导，可能无法完全反映实际情况，尤其是在环境条件或泳池使用情况变化较大时。

4. 模型主要适用于常规运营条件，对于突发的水质异常、设备故障或极端天气等情况，缺乏快速响应机制和应急调整策略。

6.3 模型的推广

模型具备良好的通用性，能够推广应用于其他类型的泳池水质管理，甚至扩展到类似的水处理系统中。通过调整模型中的参数，如衰减速率常数、加氯策略和气温预测方法，可以适应不同地区和不同环境条件下的水质管理需求。此外，模型的框架可与实时监测数据结合，开发出自动化的水质调控系统，实现水质的实时监控与智能化管理。在其他领域，如饮用水处理、工业水循环系统等，模型也可用于监测和调节水中化学物质的浓度，保障水质安全和系统稳定运行。通过进一步优化和扩展，该模型能够为各类水处理系统提供科学、精确的解决方案。

参考文献

[1] 郝艳萍. 余氯衰减一级模型参数确定方法试验研究 [D]. 哈尔滨工业大学 [2024-08-08].

[2] 林彬. 多元线性回归分析及其应用 [J]. 中国科技信息, 2010(9):2.

[3] https://www.tianqishi.com/lishi/hangzhou.html.

[4] 吕红燕, 冯倩. 随机森林算法研究综述 [J]. 河北省科学院学报, 2019, 36(3):5.

[5] Dyer S A , Dyer J S .Cubic-spline interpolation. 1[J].IEEE Instrumentation & Measurement Magazine, 2001, 4(1):44-46.

附录 1 问题一代码

```python
import math

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

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


hours = int(t_min)
minutes = int((t_min - hours) * 60)

print(f"k:{k}")
print(f"t: {hours}小时{minutes}分钟")
```

附录 2 问题二代码

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit
from sklearn.metrics import r2_score
from scipy.stats import f_oneway
plt.rcParams['font.sans-serif'] = ['SimHei']

# 数据
swimmers = np.array([0, 10, 20, 40, 60, 80, 100, 120, 140, 160, 180, 200, 220, 240, 260])
cl_concen = np.array([0.4762, 0.4752, 0.4721, 0.4698, 0.4645, 0.4535, 0.4358, 0.4199, 0.4014, 0.3901, 0.3699,
     0.3485, 0.3316, 0.3125, 0.3001])

# 拟合函数
def exponential(x, a, b):
    return a * np.exp(b * x)

# 线性拟合
linear_params = np.polyfit(swimmers, cl_concen, 1)
linear_fit = np.poly1d(linear_params)

# 二次拟合
quadratic_params = np.polyfit(swimmers, cl_concen, 2)
quadratic_fit = np.poly1d(quadratic_params)

# 三次拟合
cubic_params = np.polyfit(swimmers, cl_concen, 3)
cubic_fit = np.poly1d(cubic_params)

# 指数拟合
exp_params, _ = curve_fit(exponential, swimmers, cl_concen, p0=(1, -0.01))
exp_fit = lambda x: exponential(x, *exp_params)

# 计算R2值
r2_linear = r2_score(cl_concen, linear_fit(swimmers))
r2_quadratic = r2_score(cl_concen, quadratic_fit(swimmers))
r2_cubic = r2_score(cl_concen, cubic_fit(swimmers))
r2_exp = r2_score(cl_concen, exp_fit(swimmers))

# 计算F值和p值
f_linear, p_linear = f_oneway(cl_concen, linear_fit(swimmers))
f_quadratic, p_quadratic = f_oneway(cl_concen, quadratic_fit(swimmers))
f_cubic, p_cubic = f_oneway(cl_concen, cubic_fit(swimmers))
f_exp, p_exp = f_oneway(cl_concen, exp_fit(swimmers))

```

```python
# 打印拟合函数、R2值、F值和p值
print(f"线性拟合函数: y = {linear_params[0]:.10f}x + {linear_params[1]:.10f}")
print(f"线性拟合 R2: {r2_linear:.10f}, F值: {f_linear:.10f}, p值: {p_linear:.10f}")

print(f"二次拟合函数: y = {quadratic_params[0]:.10f}x^2 + {quadratic_params[1]:.10f}x + {quadratic_params[2]:.10f}")
print(f"二次拟合 R2: {r2_quadratic:.10f}, F值: {f_quadratic:.10f}, p值: {p_quadratic:.10f}")

print(f"三次拟合函数: y = {cubic_params[0]:.10f}x^3 + {cubic_params[1]:.10f}x^2 + {cubic_params[2]:.10f}x + {cubic_params[3]:.10f}")
print(f"三次拟合 R2: {r2_cubic:.10f}, F值: {f_cubic:.10f}, p值: {p_cubic:.10f}")

print(f"指数拟合函数: y = {exp_params[0]:.10f} * e^({exp_params[1]:.10f}x)")
print(f"指数拟合 R2: {r2_exp:.10f}, F值: {f_exp:.10f}, p值: {p_exp:.10f}")

# 绘图
plt.scatter(swimmers, cl_concen, label='数据点', color='black')
x = np.linspace(0, 260, 1000)
plt.plot(x, linear_fit(x), label='线性拟合', color='blue')
plt.plot(x, quadratic_fit(x), label='二次拟合', color='green')
plt.plot(x, cubic_fit(x), label='三次拟合', color='red')
plt.plot(x, exp_fit(x), label='指数拟合', color='purple')

plt.xlabel('游泳人数')
plt.ylabel('氯浓度（0.5小时）')
plt.legend()
plt.show()
```

附录 3 问题三代码

```
import numpy as np
import matplotlib.pyplot as plt
from datetime import datetime, timedelta
import math
plt.rcParams['font.sans-serif'] = ['SimHei']

# 初始化参数
C0 = 0.6
k_pause = 0.462098
opera_thr = 0.3
pause_thr = 0.05
cl_value = 0.6

# 定义营业时间段和人数
time_slots = [("9:00", "10:00"), ("10:00", "11:00"), ("11:00", "12:00"),
              ("12:00", "13:00"), ("13:00", "14:00"), ("14:00", "15:00"),
              ("15:00", "16:00"), ("16:00", "17:00"), ("17:00", "18:00"),
              ("18:00", "19:00"), ("19:00", "20:00"), ("20:00", "21:00")]

swimmers = [20, 80, 0, 150, 180, 0, 210, 340, 0, 280, 420, 190]

# 存储浓度变化和加氯时间
times = []
concen = []
cl_times = []

# 存储每个时间段的0.5小时后的浓度和k值
half_hour_concen = []
decay_rates = []

current_time = datetime.strptime("9:00", "%H:%M")
current_concen= C0

def v2(N):
    return 0.4812537698 + -0.0003123067 * N - 0.0000016057 * N ** 2

# 开始计算每分钟的浓度变化
for i, (start_time, end_time) in enumerate(time_slots):
    slot_start = datetime.strptime(start_time, "%H:%M")
    slot_end = datetime.strptime(end_time, "%H:%M")
    N = swimmers[i]

    # 获取当前时间段开始时的k值
    if N > 0:
```

```
        C1 = v2(N)
        k = -math.log(C1 / C0) / 0.5
    else:
        k = k_pause

    half_hour_concen.append(C1)
    decay_rates.append(k)

    while current_time < slot_end:
        times.append(current_time)
        concen.append(current_concen)

        # 更新浓度
        current_concen *= np.exp(-k * 1 / 60)

        # 检查是否需要加氯
        if (N > 0 and current_concen < opera_thr) or (
                N == 0 and current_concen < pause_thr):
            cl_times.append(current_time)
            current_concen = cl_value

        current_time += timedelta(minutes=1)

# 自定义刻度标签
hour_labels = [f"{hour}:00" for hour in range(9, 22)]

# 绘制浓度变化曲线
plt.figure(figsize=(12, 6))
plt.plot(times, concen, label='余氯浓度')

# 标记加氯时间
for cl_time in cl_times:
    plt.axvline(x=cl_time, color='r', linestyle='--',
                label='加氯' if cl_time == cl_times[0] else "")

# 标记阈值线
plt.axhline(opera_thr, color='y', linestyle='--', label=f'运营加氯阈值 {opera_thr} mg/L')
plt.axhline(pause_thr, color='g', linestyle='--', label=f'维护加氯阈值 {pause_thr} mg/L')
plt.axhline(0.6, color='b', linestyle='--', label='初始浓度 0.6 mg/L')

# 添加运营和闭馆时间段的背景颜色
plt.axvspan(datetime.strptime("11:00", "%H:%M"), datetime.strptime("12:00", "%H:%M"), color='#72F2EB', alpha
    =0.5)
plt.axvspan(datetime.strptime("14:00", "%H:%M"), datetime.strptime("15:00", "%H:%M"), color='#72F2EB', alpha
    =0.5)
plt.axvspan(datetime.strptime("17:00", "%H:%M"), datetime.strptime("18:00", "%H:%M"), color='#72F2EB', alpha
```

```
    =0.5)

plt.xlabel('时间')
plt.ylabel('余氯浓度 (mg/L)')
plt.title('9:00-21:00 余氯浓度变化曲线')
plt.xticks([datetime.strptime(f"{hour}:00", "%H:%M") for hour in range(9, 22)], labels=hour_labels)
plt.xticks(rotation=45)
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()

# 输出加氯时间
for ct in cl_times:
    print(f"加氯时刻: {ct.strftime('%H:%M')}")

# 输出每个运营时间段0.5小时后的浓度及k值
for i, (start_time, end_time) in enumerate(time_slots):
    print(f"{start_time}-{end_time} 时间段的 k 值: {decay_rates[i]:.6f}, 0.5小时后的余氯浓度: {half_hour_concen
        [i]:.6f} mg/L")
```

附录 4 问题四代码

```
import pandas as pd
from sklearn.ensemble import RandomForestRegressor
import matplotlib.pyplot as plt
from scipy.interpolate import CubicSpline

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

file_paths = ['2019.xlsx', '2020.xlsx', '2021.xlsx', '2022.xlsx', '2023.xlsx']
data_frames = []


for path in file_paths:
    df = pd.read_excel(path, usecols=[0, 1])
    df.columns = ['Date', 'Temperature']
    df['Temperature'] = df['Temperature'].str.replace('℃', '').astype(float)
    df['Date'] = pd.to_datetime(df['Date'], format='%d/%m/%Y %I:%M:%S %p')
    df.set_index('Date', inplace=True)
    data_frames.append(df)

def prepare(date_str):
    X = []
    y = []
    for df in data_frames:
        year = df.index[0].year
        mask = (df.index.month == int(date_str.split('-')[1])) & (df.index.day == int(date_str.split('-')[0]))
        day_data = df[mask]
        X.extend(pd.DataFrame({
            'Month': day_data.index.month,
            'Day': day_data.index.day,
            'Hour': day_data.index.hour,
        }).values)
        y.extend(day_data['Temperature'].values)
    return pd.DataFrame(X, columns=['Month', 'Day', 'Hour']), y


def forecast(date_str, forecast_start_date, forecast_end_date):

    X_train, y_train = prepare(date_str)


    model = RandomForestRegressor(n_estimators=100, random_state=42)
    model.fit(X_train, y_train)

```

```
    forecast_index = pd.date_range(start=forecast_start_date, end=forecast_end_date, freq='H')
    forecast_X = pd.DataFrame({
        'Month': forecast_index.month,
        'Day': forecast_index.day,
        'Hour': forecast_index.hour,
    }).values
    forecast_temperature = model.predict(forecast_X)

    cs = CubicSpline(forecast_index, forecast_temperature)
    x_smooth = pd.date_range(start=forecast_start_date, end=forecast_end_date, freq='10T') # 10分钟间隔
    forecast_smooth = cs(x_smooth)

    forecast_df_smooth = pd.DataFrame({
        'Forecasted Temperature': forecast_smooth,
    }, index=x_smooth)

    return forecast_df_smooth

import numpy as np
import matplotlib.pyplot as plt
from datetime import datetime, timedelta

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

C0 = 0.6
k_base = 0.462098
cl_value = 0.6

# 定义营业时间段
time_slots = [("9:00", "10:00"), ("10:00", "11:00"), ("11:00", "12:00"),
              ("12:00", "13:00"), ("13:00", "14:00"), ("14:00", "15:00"),
              ("15:00", "16:00"), ("16:00", "17:00"), ("17:00", "18:00"),
              ("18:00", "19:00"), ("19:00", "20:00"), ("20:00", "21:00")]

# 每小时的温度数据
temperatures = [22.62, 23.94, 25.08, 25.56, 26.43, 26.28, 25.05, 24.15, 22.94, 21.88, 20.89, 19.81]

#存储浓度变化和加氯时间
times = []
concen = []
cl_times = []

#存储每小时衰减率及其他信息
hourly_decay_rates = []
half_hour_concen = []
```

```
T_factor_values = []

#计算每小时浓度的变化
current_concen = C0
current_time = datetime.strptime("9:00", "%H:%M")

for i ,(start_time, end_time) in enumerate(time_slots):
    slot_start = datetime.strptime(start_time, "%H:%M")
    slot_end = datetime.strptime(end_time, "%H:%M")

    T = temperatures[i]
    T_factor = 10 ** ((T-25)/5)
    k = k_base * T_factor

    # 记录当前小时的T_factor和k值及半小时后的浓度
    T_factor_values.append(T_factor)
    hourly_decay_rates.append(k)
    half_hour_concen_value = current_concen * np.exp(-k * 0.5)
    half_hour_concen.append(half_hour_concen_value)

    if start_time in ["11:00","14:00","17:00"]:
        thr = 0.05
    else:
        thr = 0.3

    while current_time <= slot_end:
        times.append(current_time)
        concen.append(current_concen)

        current_concen *= np.exp(-k * 1/3600)

        if current_concen < thr:
            cl_times.append(current_time)
            current_concen = cl_value

        current_time += timedelta(seconds=1)

# 输出加氯时间
for ct in cl_times:
    print(f"加氯时刻: {ct.strftime('%H:%M')}")
```

附录 5 问题四代码

```python
import numpy as np
import matplotlib.pyplot as plt
from datetime import datetime, timedelta
import math

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

C0 = 0.6
k_base = 0.462098
cl_value = 0.6


time_slots = [("9:00", "10:00"), ("10:00", "11:00"), ("11:00", "12:00"),
              ("12:00", "13:00"), ("13:00", "14:00"), ("14:00", "15:00"),
              ("15:00", "16:00"), ("16:00", "17:00"), ("17:00", "18:00"),
              ("18:00", "19:00"), ("19:00", "20:00"), ("20:00", "21:00")]
temperatures = [22.62, 23.94, 25.08, 25.56, 26.43, 26.28, 25.05, 24.15, 22.94, 21.88, 20.89, 19.81]



def v2(N):
    return 0.4812537698 + -0.0003123067 * N - 0.0000016057 * N ** 2



max_swimmers = []
chlorine_concentration_over_time = []

for i, (start_time, end_time) in enumerate(time_slots):
    slot_start = datetime.strptime(start_time, "%H:%M")
    slot_end = datetime.strptime(end_time, "%H:%M")
    T = temperatures[i]
    T_factor = 10 ** ((T - 25) / 5)

    if start_time in ["11:00", "14:00", "17:00"]:
        max_swimmers.append(0)
        chlorine_concentration_over_time.append(
            ([(slot_start + timedelta(minutes=j)).strftime("%H:%M") for j in range(60)], [0] * 60))
        continue

    if start_time in ["11:00", "14:00", "17:00"]:
        thr = 0.05
    else:
```

```
        thr = 0.3

    low, high = 0, 520
    best_N = 0
    best_concentration = []

    while low <= high:
        N = (low + high) // 2
        if N > 0:
            C1 = v2(N)
            if C1 <= 0:
                C1 = 1e-6 # 防止计算中出现负值或零

            kp = -math.log(C1 / C0) / 0.5
        else:
            kp = k_base

        k = kp * T_factor
        current_concen = C0
        current_time = slot_start
        last_cl_time = None
        valid = True
        concentration = []

        while current_time < slot_end:
            concentration.append(current_concen)
            current_concen *= np.exp(-k * 1 / 60)

            if current_time.strftime("%H:%M") in ["12:00", "15:00", "18:00"]:
                if current_concen < 0.05:
                    valid = False
                    break
                current_concen = cl_value

            if current_concen < thr:
                if last_cl_time and (current_time - last_cl_time).seconds < 600:
                    valid = False
                    break
                last_cl_time = current_time
                current_concen = cl_value

            current_time += timedelta(minutes=1)

        if valid:
            best_N = N
            best_concentration = concentration
```

```
            low = N + 1
        else:
            high = N - 1

    max_swimmers.append(best_N)
print("各时间段可容纳的最大游泳者数量:", max_swimmers)
```