游泳池余氯浓度分析和利益最优化效率研究

摘要

为了确保游泳者的身体健康，游泳馆需要对泳池进行定期消毒，目前常用的方法是使用氯消毒方法。本文针对于氯消毒优化进行了模型建立与求解，以助于管理者进行合理消毒规划。

针对问题 1，要求根据当日最高气温和最低气温数据建立气温模拟曲线，通过分析得出，可以使用分段的**余弦函数**近似对气温曲线进行拟合。因此，我们使用改进的 **DSTC 模型**进行气温函数的模拟，增加自然衰减函数模拟无太阳辐射的情况，之后使用题目所给参数进行带入求解，得出拟合结果。

针对问题 2，由于忽略游泳者对氯含量的影响，因此只需要考虑温度和余氯浓度对反应速率的影响，采用**有限差分法**对其进行求解。在前半问，温度为定值时，建立余氯关于时间的函数表达式，对于后半问，考虑温度会随着时间流逝而变化，引用第一问的温度表达式，分别建立余氯关于温度、温度关于时间的函数，之后进行变量替换建立完整模型。求解得出，在温度为 25° C 时，经过 **22 小时**，余氯浓度达到 0.05mg/L；当温度随时间变化时，在 **10 点 06 分**余氯浓度下降低至 0.4mg/L。

针对问题 3，需要考虑游泳者的存在，会使得氯浓度下降速度大大加快，因此需要及时补充氯含量。我们根据第二问所得余氯量变化函数曲线，在其中加入游泳人数变量的影响，重新进行函数拟合，得到不同情况下余氯浓度下降低至 0.4mg/L 的点，即加药时间点。由于加药时间与温度变化情况有很大关系，在这里我们使用 7 月 30 日气温变化情况作为基本曲线，进行拟合。之后，根据每小时在场人数，进行差分方程求解，得出一天内需要加药 **85 次**，并绘制了余氯变化曲线。

针对问题 4，需要尽可能的增加游泳馆内的游泳人数，以此提高营业量，增加收入。因此，我们建立了单目标最优化模型，以泳池中所能容纳的总人次为优化函数，根据约束条件进行求解。之后我们使用**模拟退火算法**对优化模型进行求解，得出在温度高于 **34.28°C** 时，需要限制人数，或者将温度降低，此外，也可提高加药效率，当加药浓度速率达到 **4.7 mg/L·h** 时，37°C 时仍然可以在有利于营业。并以此给出合理建议，帮助管理者进行规划。

最后，我们对模型进行了**优缺点分析以及推广**，期望得到模型的优化以及进一步应用。

关键词： 气温拟合；余氯变化；差分方程；最优化模型；退火算法

目录

一、问题重述

游泳池水中的余氯浓度的影响因素主要包含气温、消毒剂、游泳者以及泳池容积、营业 时间等。主要因素如气温则考虑日最低、最高气温，以及一天内的气温变化造成余氯浓度随 之变化的影响；消毒剂则包括溶解度、加药泵流速和加药时长等影响余氯浓度增量；游泳者 主要是指游泳人数和时长对余氯量下降的影响。

问题 1：在近似的认为室内外温度相同条件下，求解气温变化模型。要求由 2022 年 6 月 27 日最高气温和最低气温数据，以此建立模型来描述述白天（假设平均日长为 14 小时）的气温和晚上（平均夜长为 10 小时）气温随时间的变化规律，之后，再根据杭州历史最高气温和最低气温（附件 2 和网站）建立预测模型，预测 2022 年 7 月 29 日和 7 月 30 日杭州电子科技大学游泳馆内两天各自 24 小时的气温变化曲线。

问题 2：氯反应速率和游离态余氯浓度成正比，且受到气温影响，在此条件下。当气温控制在常温（25℃）情况下，余氯从起始浓度 0.6mg/L 反应降到 0.05mg/L 时，统计得平均反应速率$\bar{v}$= 0.025mg/(L·h)，建立合适模型，获得反应速率与余氯浓度的关系，求解余氯浓度达到 0.05mg/L 的时长，并描绘出余氯量随时 间的变化曲线。

问题 3：在有游泳者情况下，当池水余氯浓度小于 0.4mg/L 时必须立即开启加药泵，使得余氯量快速增至 0.6mg/L 时关闭。因为游泳者存在的原因，会对池中余氯浓度有明显影响，需要根据附件数据，建立在游泳者存在的情况下余氯量的下降规律，并以此来确定第一次加药时间。同学们建模分析余氯浓度下降规律，并确定第一次加药时间。之后，再根据人员变动数量，给出一天内加药的时间点，保障余氯量不低于 0.4mg/L，并以此绘制出完整的余氯浓度变化值。

问题 4：在加药需要时间的前提下，考虑到由于气温与游泳人数对余氯浓度下降的影响，可能会存在余氯量偏低而频繁加药的情况，从而会造成游泳人数进场的限制。以此，希望通过建立优化模型，优化加药时间点，从而减少加药次数，尽可能提高营业收入，最后对模型进行评价分析，提出建议帮助管理者进行合理规划。

二、问题分析

针对问题 1，要求根据当天的最高温和最低温，来拟合当日的变化曲线，考虑使用余弦函数模型，以此来近似表示当日的温度变化。因此，需要确定余弦函数的相关参数，进行图像拟合。此外，余弦函数的拟合效果可能不尽人意，因此在无太阳辐射时，引入自然衰减函数来使得温度自然衰减，增加气温拟合准确度。

针对问题 2，氯反应速率和游离态余氯浓度成正比，在不考虑温度影响下，可以建立其氯翻译与游离态余氯浓度的关系表达式，以此为基础，加入时间变量，

进行前半部分的函数拟合，考虑后半部分是，需要考虑当日温度的变化曲线，即不同时间的温度值不相同，因此，根据题目中公式，需要重新建立函数模型，进行氯浓度变化量拟合，最后，根据拟合模型，求解达到特定浓度的所需时间

针对问题 3，需要考虑游泳者的存在，会使得氯浓度下降速度大大加快，因此需要及时补充氯含量，当泳泳者数量变化时，氯浓度的下降速率也不尽相同，因此需要根据每个小时内游泳者的数量，来确定不同的氯浓度变化，当氯浓度降低到 0.4mg/L 时，立刻添加氯，并以此模型绘画氯含量变化图像，进行拟合。

针对问题 4，需要尽可能的增加游泳馆内的游泳人数，以此提高营业量，增加收入。因此，需要合理控制加氯的时间以及次数，尽可能的使泳池中的氯含量保持在合理范围内。考虑建立最优化模型，以泳池中所能容纳的总人次为优化函数，根据约束条件进行求解。在求解过程中使用退火算法，以求获取全局最优解。

三、问题假设

1. 在第一问，忽略天气对于当日内气温变化的影响，即默认当日气温为晴朗。
2. 近似认为室内气温与室外气温相同，水温和室内气温相差为固定值。
3. 假设氯的反应速率只与温度和反应物浓度有关。
4. 在第三问中，忽略加药时间对余氯浓度的影响。
5. 假设在每一小时的时间内，游泳池内无人员出入，即每小时游泳人数为定值。

四、符号说明

符号	含义
T_0	日出前后的残余温度
T_{a}	温度振幅
ω	余弦函数半周期宽度
t_{m}	温度到达最大值的时间
t_{s}	温度开始自由衰减的时间
v	氯的反应速率
y	游离态余氯浓度
h	差分发设定的步长
T	温度
N	游泳者人数
α	退火算法的降温系数

五、模型的建立与求解

5.1 问题 1 模型的建立与求解

5.1.1 模拟地表温度日变化模型建立

由题目要求，需要用当日之中的最高温度和最低温度来确定当日的日温变化曲线，是有一定难度的，因为当日的气温变化与许多因素有关，如天气情况，云层厚度，风力风向等。若只知晓当日气温极值，则需要建立合理的参数输入模型，输入最高气温和最低气温，以及必要的常量参数后，进行模型的拟合。

而根据经验公式和观察发现，日气温变化曲线与余弦函数 cos 的变化非常相似，因此考虑建立以余弦函数为基本函数的日气温变化曲线。在此基础上，只需要确认余弦函数的基本参数值，便可得到当日的气温变换函数，进行拟合求解。在这里我们考虑使用地表温度日变化模型。

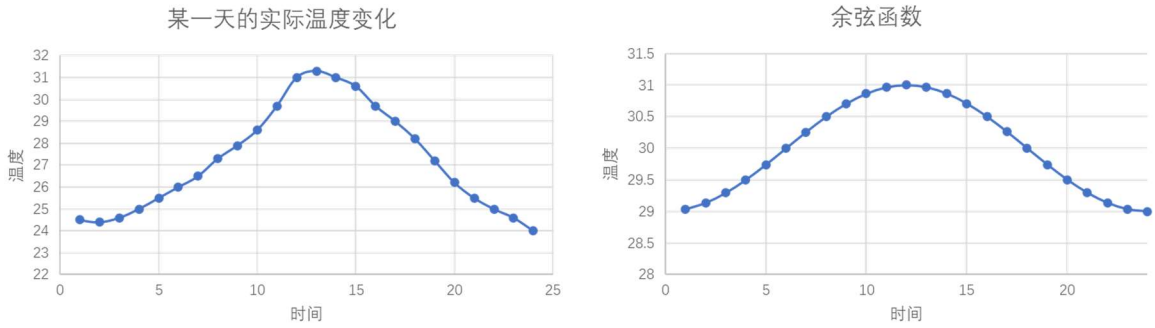

图 1: 实际温度曲线和余弦函数的对比

地表温度日变化模型 DSTC 是一种简化的气温模拟模型，其作为非常重要的输入参数模型在气候学和气象学研究中有广泛应用，可以用以计算热惯量和模拟地表温度的变化。

为了描述太阳照射的影响和夜间地表温度的自然下降趋势，Göttsche 和 Olesen[1]使用热扩散方程构建谐波方程拟合白天的温度变化曲线，选择指数函数来模拟自然地表夜晚的温度变化曲线。

该模型属于半经验模型，建立该模型有 3 个要求：

(1)晴空条件且风速没有明显变化；

(2)模型建立于两次日出之间；

(3)在定义的时间 ts 之后，地表温度自由降低。模型描述如下：

$$\begin{cases} T_1(t) = T_0 + T_{\mathrm{a}}\cos\left(\dfrac{\pi}{\omega}(t - t_{\mathrm{m}})\right) \; t < t_{\mathrm{s}} \\ T_2(t) = (T_0 + \delta T) + \left(T_{\mathrm{a}}\cos\left(\dfrac{\pi}{\omega}(t_{\mathrm{s}} - t_{\mathrm{m}})\right) - \delta T\right)\mathrm{e}^{\frac{-(t,t\ \mathrm{s})}{k}t} t \geqslant t_{\mathrm{s}} \end{cases} \tag{1}$$

Ts 指的是日落时，当有太阳辐射是，温度将遵循公式二的变化，当日落后，太阳辐射消失，因此温度将自由下降。

式中， T_1/T_2 是模型中分别代表白天/夜晚温度的两个部分。其中 t 是时间， T_0 是日出前后的残余温度， T_{a} 是温度振幅， ω 是余弦函数半周期宽度， t_{m} 是温度到达最大值的时间， t_{s} 是温度开始自由衰减的时 间， δT 是 T_0 和 $T(t \to \infty)$ 的差值， k 是衰减常数。模型中有 5 个自由变量$T_0, T_{\mathrm{a}}, t_{\mathrm{m}}, t_{\mathrm{s}}$和 δT 。模型 中的 ω 定义为有太阳能量输入的时长，由纬度 ϕ 和太阳倾斜角 δ 决定[2]，其表示为

$$\begin{aligned} \omega &= \frac{2}{15}\cos^{-1}(-\tan\ \phi\tan\ \delta) \\ \delta &= 23.45\sin\left(360\frac{284+n}{365}\right) \end{aligned} \tag{2}$$

由于 DSTC 模型曲线连续，白天和夜晚的温度模拟函数在 t_{s} 处的导数应相同，由此可获得衰减常数[3]

$$k = \frac{\omega}{\pi}\left(\tan^{-1}\left(\frac{\pi}{\omega}(t_{\mathrm{s}} - t_{\mathrm{m}})\right) - \frac{\delta T}{T_{\mathrm{a}}}\sin^{-1}\left(\frac{\pi}{\omega}(t_{\mathrm{s}} - t_{\mathrm{m}})\right)\right) \tag{3}$$

对于问题一，需要给出白天温度上升和夜晚温度下降的分段函数，因此需要对模型进行一定的改进，采用两个余弦函数分别描述白天温度上升过程和温度下降过程的地表温度的变化情况，采用指数衰减模型描述晚上气温变化，新模型包含 6 个自由变量 $T0, Ta, tm, ts, \omega 1, \omega 2$ ，新模型描述如下：

$$\begin{cases} T_1(t) = T_0 + T_{\mathrm{a}}\cos\left(\dfrac{\pi}{\omega_1}(t - t_{\mathrm{m}})\right) & t < t_{\mathrm{m}} \\ T_2(t) = T_0 + T_{\mathrm{a}}\cos\left(\dfrac{\pi}{\omega_2}(t - t_{\mathrm{m}})\right) & t_{\mathrm{m}} \leqslant t < t_{\mathrm{s}} \\ k = \dfrac{\omega_2}{\pi}\tan^{-1}\left(\dfrac{\pi}{\omega_2}(t_{\mathrm{s}} - t_{\mathrm{m}})\right) & \\ T_3(t) = T_0 + T_{\mathrm{a}}\cos\left(\dfrac{\pi}{\omega_2}(t_{\mathrm{s}} - t_{\mathrm{m}})\right)e^{\frac{-(t-\mathrm{s})}{k}} & t \geqslant t_{\mathrm{s}} \end{cases} \tag{4}$$

式中， T_1，T_2，T_3 分别表示白天温度上升过程、温度下降过程、夜晚 3 个时间段，以温度达到最高值 t_{m} 和温度自由下降的 t_{s} 两个时间点为分界。$\omega 1, \omega 2$ 可由往年同期温度数据拟合得出。

5.1.2 当日气温变化模型求解

当我们使用原始未改进的模型进行求解后，与 6 月 27 日实际逐小时温度变化进行比较，发现在白天的拟合效果较好，但在夜晚的气温差异较大，可能是由于夜晚温度为自由衰减，而白天有太阳辐射的原因，因此，考虑使用改进后的模型重新求解[4]。

首先我们对 6 月 27 日温度变化进行求解，根据杭州气象局官方数据可得，2022 年 6 月 27 日的最高气温为：29.8℃，最低气温为：36.8℃。

并求得其他参数为：

参数	取值
$\omega 1$	-19.5615
$\omega 2$	13.5006
t_s	19
t_m	15
T_0	29.8
T_a	7

表 1：6 月 27 日温度变化模拟参数表

带入求解得本日温度变化函数图像为

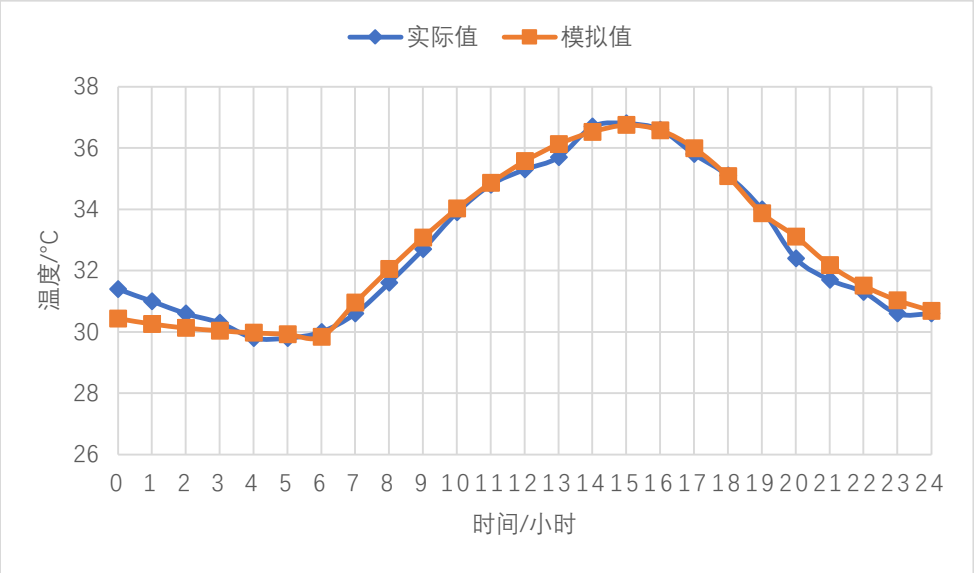

图 2： 6 月 27 日温度变化模拟曲线

我们将模拟值与实际值对比可以看出，两条曲线的整体变化趋势相对较拟合，说明拟合效果较好。通过拟合曲线可以看出，最低温在日出前后，而为高温在下午 15 点左右，符合常理，因此可以接受拟合曲线。

5.1.3 气温变化预测模型求解

在后半问中，我们需要更具以往的温度最高值和最低值数据需要对 2022 年 7 月 29 日和 7 月 30 日的温度变化进行预测。

在这里，我们使用以往 10 年的温度峰值，来预测近年当天的气温值，其中，我们只考虑天气晴朗的情况，因此，将以往 10 年的雨天等特殊天气情况去除，进行数据的优化和整理，最后求得相关的参数如下表。

参数	取值
$\omega 1$	-20.7092
$\omega 2$	9.4003
t_s	19
t_m	16.134
T_0	28
T_a	9.5

表 2： 7 月 29 日温度变化模拟参数表

带入求解得本日温度变化函数图像为：

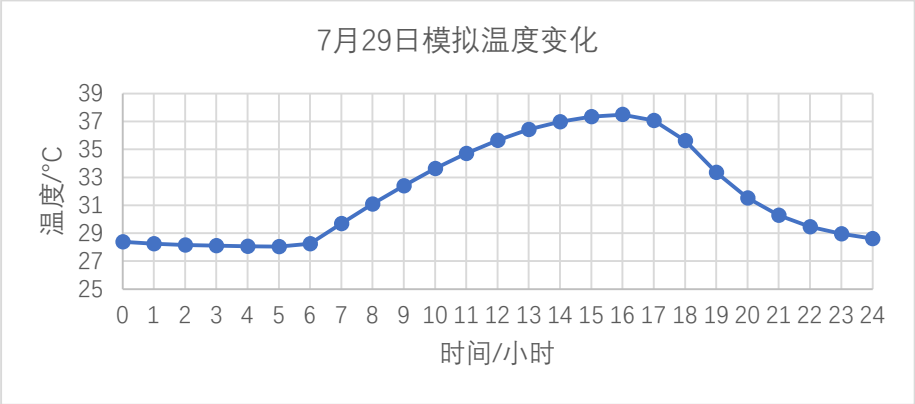

图 3： 7 月 29 日温度变化模拟曲线

根据拟合曲线可以看出，最低温度出现在 6 点左右，即日出前后，最高气温出现在下午 4 点左右。最高气温为 37.2°C,最低气温为 28.5°C。通过与往年数据进行分析比较，得到拟合数据较为可信。

相同方法求得 7 月 30 日的各项参数如下表所示：

参数	取值
$\omega 1$	-18.24386
$\omega 2$	12.5865
t_s	19
t_m	14
T_0	27.71
T_a	9.86

表 3： 7 月 30 日温度变化模拟参数表

带入求解得本日温度变化函数图像为：

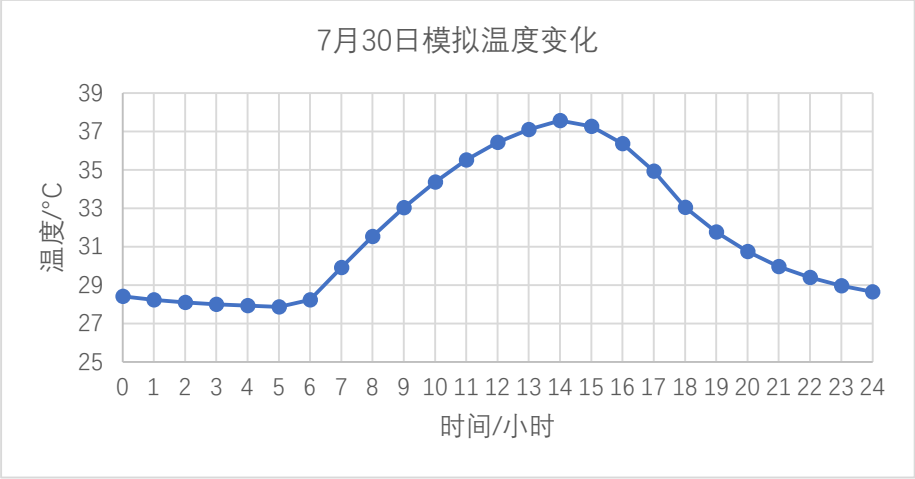

图 4： 7 月 30 日温度变化模拟曲线

根据拟合曲线可以看出，最低温度出现在 6 点左右，即日出前后，最高气温出现在下午 4 点左右。最高气温为 37.3°C,最低气温为 28.1°C。通过与往年数据进行分析比较，得到拟合数据较为可信。

5.2 问题二模型的建立与求解

5.2.1 恒温速率模型的建立：

根据题意，池水中氯的反应速率v (mg/(L · h))跟游离态余氯浓度y(mg/L) 成正比，且受到气温T(℃)影响。前半部分给定气温为常温（25℃），而后半部分需要建立合适模型，获得反应速率与余氯浓度的关系。

由题目中所给关系可建立微分方程

设定反应速率与含氯浓度的比率α，则其关系为：

$$\begin{cases} v = \alpha \cdot y \\ \dfrac{dy}{dt} = \alpha \cdot y \end{cases} \tag{5}$$

设定初始浓度为c_1，浓度随时间的变化关系为：

$$y_1 = e^{\alpha t + c_0} = c_1 e^{\alpha t} \tag{6}$$

因为平均反应速率 v=0.025 mg/(L*h)，初始浓度为 0.6 mg/L，可以求解得

$$\begin{cases} \alpha = -0.1129 \dfrac{mg}{L^2 \cdot h} \\ c_1 = 0.6 \dfrac{mg}{L} \end{cases} \tag{7}$$

5.2.2 温度变化时速率模型的建立：

考虑到气温因素时，余氯反应速率不单单只和游离态余氯浓度有关，还和当时的气温变化有关，因此需要进行新的拟合求解

温度变化函数由第一问的结果得出，第一问的温度变化函数分三段，我们将其分别带入，进行温度随时间变化的求解。

之后将此变量带入到速率模型中，进行多参数的模型建立。

由题目可设应速率与含氯浓度和温度的关系为：

$$\begin{cases} v = \beta \cdot 10^{\frac{T-25}{5}} y \\ \dfrac{dy}{dt} = \beta \cdot 10^{\frac{T-25}{5}} y \end{cases} \tag{8}$$

设定初始浓度为c_2，浓度随时间的变化关系为：

$$y_2 = c_2 \cdot e^{\int \beta \cdot 10^{\frac{T-25}{5}}\, dt} \tag{9}$$

由于当 T=25℃时

$$y_2 = y_1$$

故可以解得

$$\begin{cases} \beta = \alpha = -0.1129\, mg/(L^2 \cdot h) \\ c_2 = c_1 = 0.6\, mg/L \end{cases} \tag{10}$$

5.2.3 温度不变时速率模型的求解

利用差分的方法求解

$$\begin{cases} y' = \alpha \cdot y \\ y(t=0) = 0.6 \end{cases} \tag{11}$$

设定步长为 h，方程转变为：

$$(y_{i+1} - y_i)/h \ = \ \alpha \cdot y_i \tag{12}$$

得到差分方程为：

$$y_{i+1} = y_i + h \cdot \alpha \cdot y_i \tag{13}$$

求解出不同时间的浓度值的大小，然后将函数模型可视化，得到余氯量随时间变化曲线如下：

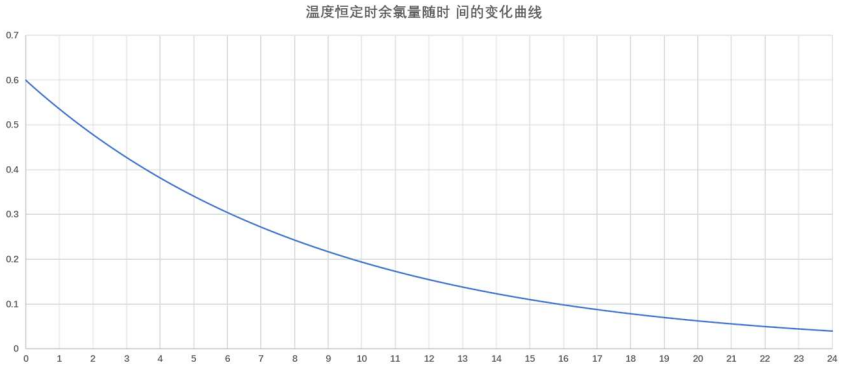

图 5： 温度恒定时余氯量随时间的变化曲线

可以看出，当温度恒为 25 度时，余氯量随时间下降的较慢，有题，需要计算求解余氯浓度达到 0.05mg/L 的时长，可得此时长为 22 小时。

5.2.4 温度变化时速率模型的求解

根据生活经验以及题目要求，室内气温和水温有一定的差值，因此，计算水温对于余氯下降速度的变化影响时，需要对此差值进行考虑，本文中将其取为定值 3°C，即认为水温恒定比室温低 3°C[5]。

利用差分的方法求解

$$\begin{cases} y' = B * 10^{\frac{T-28}{5}} * y \\ y(t=0) = 0.6 \end{cases} \qquad (14)$$

设定步长为 h，方程转变为：

$$(y_{i+1} - y_i)/h \ = \ B * 10^{\frac{T-28}{5}} * y_i \qquad (15)$$

得到差分方程为：

$$y_{i+1} = y_i + h * B * 10^{\frac{T-28}{5}} * y_i \qquad (16)$$

求解出不同时间的浓度值的大小，然后将函数模型可视化，得到余氯量随时间变化曲线如下：

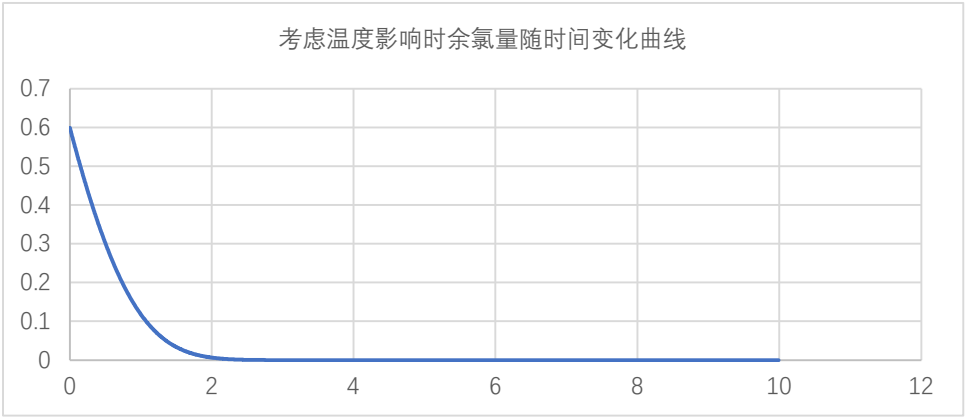

图 6： 考虑温度影响时余氯量随时间变化曲线

可以看到，当考虑温度影响后，余氯量的下降速率受到很大影响，当温度上升 10 度时，下降速率将变为原来的 10 倍。因此整体下降速率将变得很快。

由图像可以得到，当三小时过后，余氯量已经几乎变为 0。

问题 3 中要求求解下降至 0.4mg/L 的时刻，根据函数和图像得知 1.099 小时后到达此值，即 10.点 06 分的时候，氯浓度下降至 0.4mg/L。

5.3 问题三模型的建立与求解

5.3.1 模型的建立

在解答过程中，只考虑气温变化、氯浓度以及游泳者的存在才会使氯的反应速率发生变化，并假设气温和泳池中人的数量对氯的浓度影响相互独立。因此，余氯浓度的下降曲线与气温，浓度，以及游泳者数量有关，再经过变量替换后，建立与时间的函数关系式[6]。

此外，当氯浓度下降至 0.4mg/L 的时刻时，需要进行加氯操作，以保证氯的消毒杀菌效果。在分析过程中，忽略添加氯的加药时间，当氯浓度低于阈值时，立刻加药，使得浓度变为 0.6mg/L，之后在根据下降曲线进行下降，最终建立完整的氯浓度变化曲线[7]。

根据附件三的数据，当游泳人数为 0 时，只有余氯反应，在 1 小时内，其消耗的氯浓度为：

$$C_1 = 0.6 - 0.5359 = 0.461mg/L$$

假设由游泳者消耗的氯浓度为 C2，一小时后氯浓度余量为 Ci，（i 为游泳者的数量）

$$C_2 = C_i - C_1$$

利用 spss 软件对游泳者消耗的氯浓度 C2 和游泳者人数 N 进行线行回归分析，采用最小二乘法进行参数得到：

$$C_i = K * N + B1$$

解得，K=0.0004，B1=0.00015，R2=0.99，说明回归效果非常显著。

如果经过 t 个小时，游泳者为 N，则游泳者消耗的氯浓度 y1 为：

$$y_1 = K * N * t$$

氯在水中反应剩余的氯浓度为

$$y_2 = A * e^{\int_0^t B*10^{\frac{T-28}{5}}dt} \quad (17)$$

与氯在水中反应消耗的氯浓度相加得到总的消耗量

$$y = y_1 + y_2 = A * e^{\int_0^t B*10^{\frac{T-28}{5}}dt} - K * N * t \quad (18)$$

因此，只需要进行消耗量曲线的拟合，就可以进行余氯浓度的求解，在方程中，影响因素有人数以及当日气温变化，在求解第一次加药时间时，需要以

游泳人数为自变量，来获取不同游泳人数状况下，第一次加药时间的不同。在求解余氯浓度变化曲线时，带入不同时刻的游泳人数，以此来获得游泳人数对余氯浓度的影响，进行拟合。

5.3.2 第一次加药时间的求解

为了获取余氯变化曲线函数，采用差分的方法进行求解

$$\begin{cases} y' = B * 10^{\frac{T-28}{5}} * y - K * N \\ y(t=0) = 0.6 \end{cases} \tag{19}$$

设定步长为 h，方程转变为：

$$(y_{i+1} - y_i)/h = B * 10^{\frac{T-28}{5}} * y_i - K * N \tag{20}$$

得到差分方程为：

$$y_{i+1} = y_i + h * B * 10^{\frac{T-28}{5}} * y_i - h * K * N \tag{21}$$

利用模型一求解出的 7 月 30 日的温度数据，得到不同游泳者数量下，第一次加药的时间为：

游泳人数	加药时间	游泳人数	加药时间
0	9.321	260	9.278
10	9.319	280	9.275
20	9.317	300	9.273
40	9.313	320	9.27
60	9.31	340	9.267
80	9.306	360	9.265
100	9.303	380	9.262
120	9.3	400	9.26
140	9.296	420	9.257
160	9.293	440	9.255
180	9.29	460	9.252
200	9.287	480	9.25
220	9.284	500	9.248
240	9.281	520	9.245

表 4：不同游泳人数时的第一次加药时间

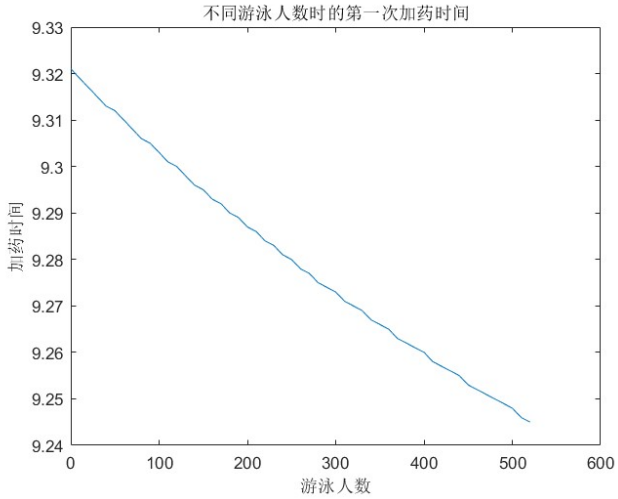

图 7：不同游泳人数时的第一次加药时间

从图中可以看出，随着游泳人数的增加，第一次加药时间近似呈线性减小趋势，说明游泳人数越多，第一次加药的时间越早，即余氯浓度的下降速率越快，这在逻辑上也是符合变化规律的。

5.3.3 余氯浓度变化求解的求解

考虑到整天泳池人数不断变化，我们在求解加药时间时，随不同的时间段来改变游泳人数这一参数[8]。在营业时间，每当氯浓度低于 0.4mg/L 时，就立即加药使其浓度变为 0.6mg/L，在非营业时间，不进行加药处理。因为 9 点开始营业，21 点停止营业，所以我们只考虑 9 点到 21 点的氯浓度变化规律，得到结果如下：

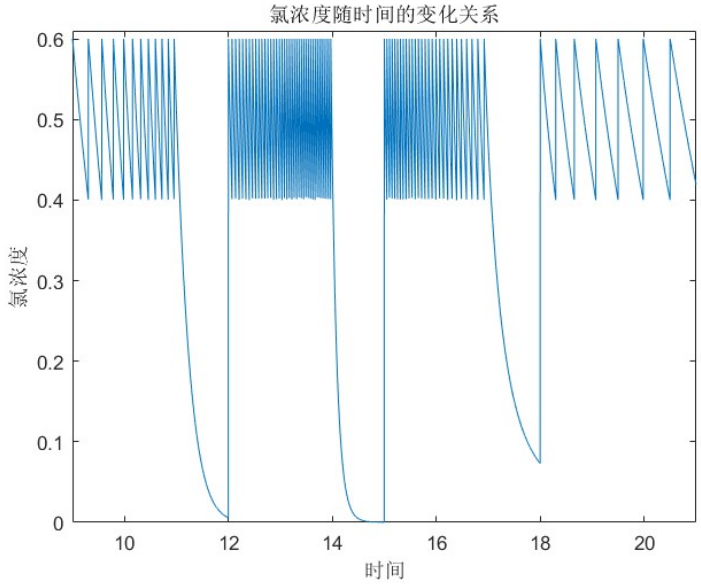

图 8：9 点到 21 点内的氯浓度变化

通过分析氯浓度变化规律可以发现，在营业时间氯浓度可以有效的控制在 0.4~0.6mg/L，在非营业时间，氯含量自由下降，只需要营业开始前进行一次加药

即可，不需要考虑在非营业情况下氯浓度的降低情况。

通过余氯变化曲线我们可以得到，当浓度低于 0.4 时，需要立刻进行加药，得出在不同时间段的加药次数分布图像，并得到总的加药次数为 85 次。

时间段	9-10	10-11	11-12	12-13	13-14	14-15	总计 85 次
次数	4	7	0	17	22	0	
时间段	15-16	16-17	17-18	18-19	19-20	20-21	
次数	18	10	0	3	3	1	

表 7：各时间段添加氯的次数

我们将表格数据进行可视化展示，得到加药频率直方图如下：

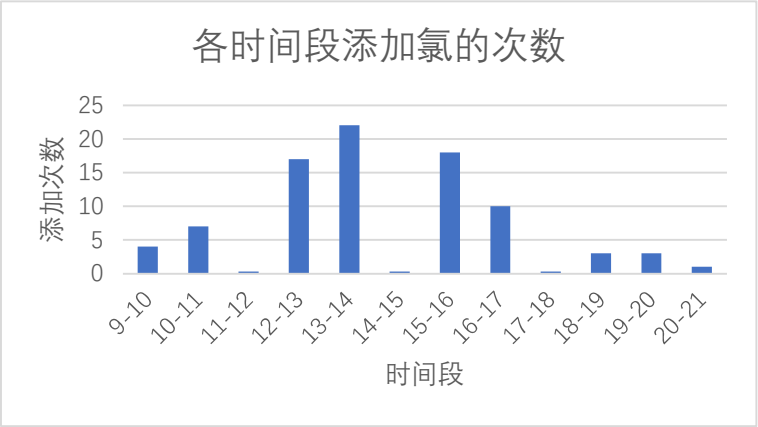

图 9：各时间段添加氯的次数

通过分析加药时间规律规律可以发现，不同营业时间加药的次数明显不同。随着一天当中温度的变化，每个时间段所需要加药的次数也不尽相同，时间段内温度越高，所需加药次数越多。在 13 点到 14 点加药最为频繁，这是因为这一时间段温度最高，氯的下降速率相对较快，因此需要频繁加药。

5.4 问题四模型的建立与求解

5.4.1 最优化模型的建立

根据上述分析可以得出，当日的温度变化和每个时刻游泳的人数是影响余氯浓度变化的两个重要因素。为了增加营业收入，希望尽可能的增加游泳人数，而温度上升又使得余氯浓度下降速率增快，因此需要综合考虑两个影响因素，建立最优化模型，进行综合效益最优化的求解。

此外，由附件 4 可得，加药对氯浓度上升的速率影响为恒定的，因此希望加药使余氯浓度增加，气温与人数使余氯浓度减小的变化尽量平衡，从而使得氯浓度在合理的范围内变化。

因此，为了保证游泳池中率的浓度在安全范围内，就要使得给药速度大于消耗氯的速度。所以建议可以从三个方向考虑，一是限制游泳池内人数，二是降低场馆内温度，三是要加大给药速率。

利用 spss 软件对附件 4 中加药时间 t 和余氯浓度增量△y 进行线行回归分析，采用最小二乘法进行参数得到：

$$y_3 = k1 \cdot t + b1$$

解得，K1=1.475, $b1$=*0.0002*，R^2 =0.99，说明回归效果非常显著。

故游泳池中氯浓度消耗量 y 为

$$y = y1 + y2 - y3 = A \cdot e^{\int_0^t B*10^{\frac{T-28}{5}}dt} + K \cdot N \cdot t - k1 \cdot t \qquad (22)$$

氯浓度的总消耗速率

$$v = y' = A \cdot 10^{\frac{T-28}{5}} \cdot y + K \cdot N - K1 \qquad (23)$$

当游泳池中氯浓度为 0.04mg/L 时，氯浓度的消耗速率最小

我们要保证$v \le 0$恒成立，这样才能控制氯浓度不会低于 0.04mg/L，有

当游泳池中氯浓度为 0.04mg/L 时，氯浓度的消耗速率最小，我们要保证$v \le 0$恒成立，这样才能控制氯浓度不会低于 0.04mg/L，同时，由于人数 N 对速率的影响太小，我们的 N 都选取为 521 人。

我们的目标是使游泳池内能够容纳的人数最多，即 N 最大。

$$\max \quad N = N1 + N2 + \cdots + N9 \qquad (24)$$

N1,N2,…N9 分别代表营业期间每小时在场人数,每场的人数要满足

$$0 \le N_i \le 521 \qquad (25)$$

同时要保证游泳池中氯浓度

$$0.04 \le y \le 0.06 \qquad (26)$$

5.4.2 加大给药速率模型的求解

只要当天最高温度下，能够保证氯浓度的消耗速率小于加药速率，就能保证全天都能控制氯的浓度$v \le 0$，取 N=521，A=0.06。

$$v = A \cdot 10^{\frac{T-28}{5}} \cdot y + K \cdot N - k_1 \qquad (27)$$

对于不同的最高温度，要保证$v \le 0$，可以解得所需的最小加药速率 v1，将温度和加药速率 v1 做图如下：

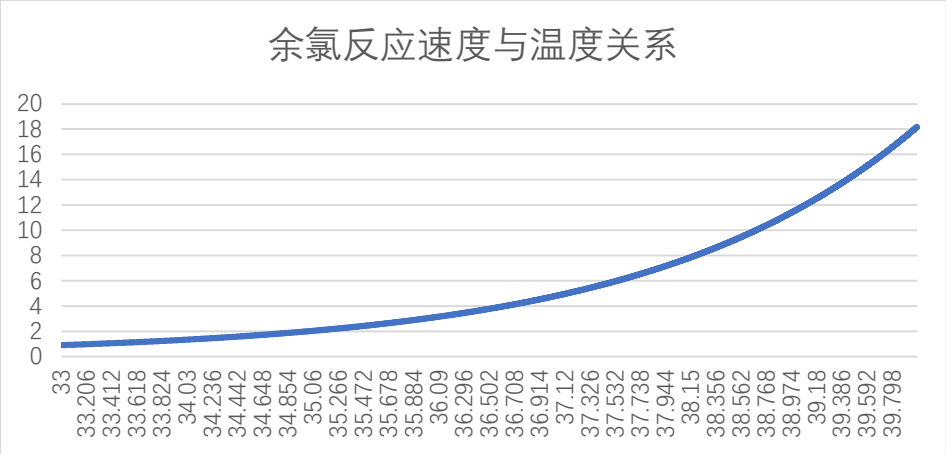

图 10：最高温度与所需加药的关系

5.4.3 减小游泳人数模型的求解

利用退化算法求解减小游泳人数模型。

模拟退火算法最早的思想是由 N.Metropolis 等人于 1953 年提出[6]。它是基于迭代求解策略的一种随机寻优算法，其出发点是基于物理中固体物质的退火过程与一般组合优化问题之间的相似性。该算法的优点为求解速度快，可达到全局最优解。

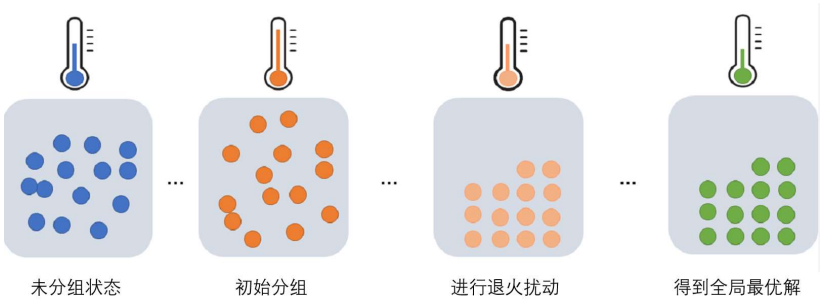

图 11： 退火算法示意图

模拟退火算法包括三函数两准则，即状态产生函数，状态接受函数，温度更新函数；内循环终止和外循环终止准则[6]。这些环节之间影响模拟退火算法的优化性能。

求解模拟退火算法描述如下：

(1)解空间。解空间 S 可表示为个场次游泳人数的分布。

(2)目标函数。目标函数为：

$$\max \quad N = N1 + N2 + \cdots + N9$$

(3)新解的产生。随机挑选一个场次，更改该场次人数的数目。

(4)代价函数差。假设原先的分组数据隐匿量为 T1，新的分组数据隐匿量为 T2。隐匿量差可表示为：

$$\Delta N = N2 - N1$$

(5)接受准则

$$P=\begin{cases}1,\Delta N<0\\ exp(-\Delta N/T),\Delta N\geq 0\end{cases}$$

如果$\Delta N<0$，则接受新的分组；否则，以概率$exp(-\Delta N/T)$接受新的分组，即用计算机产生一个[0,1]区间上均匀分布的随机数 rand，若rand <= $exp(-\Delta N/T)$则接受。

(6)降温。利用选定的降温系数 α 进行降温，取新的温度 T 为 AT(这里 T 为上一步迭代的温度)，这里选 α=0.999。

(7)结束条件。用选定的终止温度$e=10^{-3}$，判断退火过程是否结束。若 T<e，则算发结束，输出当前状态。

假设每一天的最高温度时刻为下午 15 点，如果此时温度过高，使得无法保证氯浓度的总消耗速率$v\leq 0$，那么下午 15 到 16 点这个时间段就不能营业。

取 N=521，A=0.06，k1=1.475。

要使$v\leq 0$，求得最高温度$T\leq 34.286℃$，故此时任何时间都可以满人数营业。

由第一问温度模型可知，在 15 点的温度 T>34.576℃ 时，14 点和 16 点的温度非常相近，其温度$T>34.286℃$，则此时 13~14 点与 16~17 点也无法控制氯浓度的总消耗速率$v\leq 0$，故不能营业，白天和晚上仍然可以营业；

在 15 点温度$34.286℃<T<34.576℃$时，15~16 点无法控制氯浓度的总消耗速率$v\leq 0$，不能营业，而要 13~14 点与 16~17 点要限制游泳人数。

故不同最高温度下的营业人数为。

$$\begin{cases}521*6\text{ 人} & T>34.576℃\\ 521*7\text{ 人} & 34.286℃<T<34.576℃\\ 521*9\text{ 人} & T\leq 34.286℃\end{cases}\qquad(28)$$

基于以上求解，我们给出如下建议：

尊敬的游泳馆管理者：

首先祝贺游泳馆开业大吉。为了保障水质安全，需要定期进行氯消毒。经过我们讨论研究，得出一些有益于您更好地进行经营研究的建议，希望可以提供一定的参考价值。

- **降低室内温度。**当室内温度过高时，氯浓度的下降速率将极快，因此我们建议当温度**34.5℃**时，采用开启空调等方法降低室内温度，尽可能减少加药时间和次数。
- **优化加药设备。**加药效率提高到**4.71mg/g·h**时，可以有效减缓氯浓度的下降速度。可以增加当前时刻的最大承受人数，增加收益效率。
- **合理控制人数。**当游泳馆人数过多时，首先会加快氯浓度的下降速度，此外,过多的游泳人数也不利于游泳者的体验。

007数模团队

六 模型评价与推广

6.1 模型的优点分析

在第一问中，我们使用改进的 DSTC 模型进行气温函数的模拟，为了使拟合变化曲线更符合，将原来的余弦曲线分为三段，将最低温度时间(日出时间)，最高温度时间设置为临界点，并且增加自然衰减函数模拟无太阳辐射的情况，适合拟合曲线更精确。

而对于泳池中余氯曲线的变化情况，我们结合第一问中的温度变化曲线对其进行分析，考虑了温差，浓度等关键因素，建模较为全面，合理。之后使用差分方程以及最小二乘法进行求解，解法合理。

考虑游泳者的存在时，会使得氯浓度下降速度大大加快，我们根据题目要求，进行了曲线的拟合和求解，考虑到了营业时间，温度影响等重要条件，求解结果也较为合理，正确。

在最后一问中，通过建立单目标优化模型，来使得营销效率最优化，我们采用退火算法进行求解，使得求解过程科学合理，期望获得全局最优解。

6.2 模型的缺点分析

1. 在考虑气温变化曲线时，没有考虑天气对于气温变化的影响，可能对于某些特殊天气下温度的变化情况有所缺失，造成结果的片面性与局部性。
2. 在考虑气温与水温的温差时，将差值设置为定值，可能与实际情况不符，使得求解结果存在一定差值。
3. 在进行第四问的求解时，我们使用了 7 月 30 日的温度变化为前提条件，可能不具有普适性和代表性。

6.3 模型的改进与推广

对温度变化曲线的拟合进行优化后，考虑天气对于气温变化的影响，可以进行天气的预测和拟合，可以对气象的日常监测提供一定的参考帮助。

本模型的建立对于游泳馆的日常规划提出了一些合理建议，有助于管理者的日常经营，有利于游泳馆的日常效益最大化，增加收益效率。

参考文献

[1] Aires F, Prigent C and Rossow W B. 2004. Temporal interpolation of global surface skin temperature diurnal cycle over land under clear and cloudy conditions. Journal of Geophysical Research: Atmospheres, 109 (D4): D04313 [DOI: 10.1029/2003JD003527]

[2] Chen Y, Duan S B, Leng P, Chen Y Y and Han X J. 2016. Modeling of diurnal cycle of land surface temperature based on polar orbiting satellite thermal infrared data. Remote Sensing Information, 31 (6): 7–14. [DOI: 10.3969/j.issn.1000-3177.2016.06.002] (陈颖, 段四波, 冷佩, 陈媛媛, 韩晓静. 2016. 极轨卫星热红外地表温度日变化模拟. 遥感信息, 31 (6): 7–14. [DOI: 10.3969/j.issn.1000-3177.2016.06.002])

[3] Cheng J, Liang S L, Wang J D and Li X W. 2010. A stepwise refining algorithm of temperature and emissivity separation for hyperspectral thermal infrared data. IEEE Transactions on Geoscience and Remote Sensing, 48 (3): 1588–1597. [DOI: 10.1109/TGRS.2009.2029852]

[4] 谢昕, 郭鹏飞, 詹小丽. 基于 RBF 神经网络的余氯浓度预测模型研究[J]. 传感器与微系统, 2012, 31(8): 64-65.

[5] 刘兰芬, 郝红, 鲁光四. 电厂温排水中余氯衰减规律及其影响因素的实验研究[J]. 水利学报, 2004 (5): 94-98.

[6] 陈静文, 丁敏. 余氯检测方法与余氯标准溶液的概况[J]. 化工管理, 2013 (2): 80-82.

[7] 房康, 王根宝. 某市游泳场所卫生及消毒情况调查[J]. 中国消毒学杂志, 2012, 29(2): 151-152.

[8] 杨轶戬, 施洁, 蒋琴琴, 等. 广州市游泳池水氯化消毒副产物含量及其影响因素分析[J]. 预防医学论坛, 2015, 21(5): 324.

附录

代码附录：

```
x=0:10:520;
h=0.001;
k=-0.11295;
y=zeros(1,54);
y(1)=0.6;
for j=1:1:53
    for i=2:1:1000
        t=h*i;
        y(i) = y(i - 1) + h * k * 10^((27.71 + 9.86*cos(pi / (-
18.24386) * (9 + t - 14.8139)) - 3 - 25) / 5) * y(i - 1) -
(x(j)*0.0004-0.0015) * h;
        if(y(i)<0.4)
     fprintf('第%.4f 次结果为%.6f\n',j,t) ;
        v(j)=t+9;
        break
        end
    end
end

t=9:0.001:21;
plot(x,v)
xlim([9,21]);

title("不同游泳人数时的第一次加药时间"); %设置标题
xlabel("游泳人数"); %设置 X 轴标签
ylabel("加药时间"); %设置 Y 轴标签
```

```
h=0.001;
k=-0.11295;
y=zeros(1,12000);
y(1)=0.6;
p=3.4195;
j=1;
for i=2:1:12000
    if(y(i-1)>=0.4)
        t=9+h*i;
        if(t<=10)
```

```
            m=80;
            y(i) = y(i - 1) + h * k * 10^((27.71 + 9.86*cos(pi / (-18.24386) * (t - 14.8139)) - 3 - 25) / 5) * y(i - 1) - (m*0.0004-0.0015) * h;
        end
        if(10<t&&t<=11)
            m=190;
            y(i) = y(i - 1) + h * k * 10^((27.71 + 9.86*cos(pi / (-18.24386) * (t - 14.8139)) - 3 - 25) / 5) * y(i - 1) - (m*0.0004-0.0015) * h;
        end
        if(11<t&&t<=12)
            m=0;
            y(i) = y(i - 1) + h * k * 10^((27.71 + 9.86*cos(pi / (-18.24386) * (t - 14.8139)) - 3 - 25) / 5) * y(i - 1) - (m*0.0004-0.0015) * h;
        end
        if(12<t&&t<=13)
            m=220;
            y(i) = y(i - 1) + h * k * 10^((27.71 + 9.86*cos(pi / (-18.24386) * (t - 14.8139)) - 3 - 25) / 5) * y(i - 1) - (m*0.0004-0.0015) * h;
        end
        if(13<t&&t<=14)
            m=330;
            y(i) = y(i - 1) + h * k * 10^((27.71 + 9.86*cos(pi / (-18.24386) * (t - 14.8139)) - 3 - 25) / 5) * y(i - 1) - (m*0.0004-0.0015) * h;
        end
        if(13<t&&t<=14)
            m=340;
            y(i) = y(i - 1) + h * k * 10^((27.71 + 9.86*cos(pi / (12.5865) * (t - 14)) - 3 - 25) / 5) * y(i - 1) - (m*0.0004-0.0015) * h;
        end
        if(14<t&&t<=15)
            m=0;
```

```
            y(i) = y(i - 1) + h * k * 10^((27.71 + 9.86*cos
(pi / (12.5865) * (t - 14)) - 3 - 25) / 5) * y(i - 1) - (m*
0.0004-0.0015) * h;
        end
        if(15<t&&t<=16)
            m=280;
            y(i) = y(i - 1) + h * k * 10^((27.71 + 9.86*cos
(pi / (12.5865) * (t - 14)) - 3 - 25) / 5) * y(i - 1) - (m*
0.0004-0.0015) * h;
        end
        if(16<t&&t<=17)
            m=380;
            y(i) = y(i - 1) + h * k * 10^((27.71 + 9.86*cos
(pi / (12.5865) * (t - 14)) - 3 - 25) / 5) * y(i - 1) - (m*
0.0004-0.0015) * h;
        end
        if(17<t&&t<=18)
            m=0;
            y(i) = y(i - 1) + h * k * 10^((27.71 + 9.86*cos
(pi / (12.5865) * (t - 14)) - 3 - 25) / 5) * y(i - 1) - (m*
0.0004-0.0015) * h;
        end
        if(18<t&&t<=19)
            m=320;
            y(i) = y(i - 1) + h * k * 10^((27.71 + 9.86*cos
(pi / (12.5865) * (19-14.4096))*exp(-(t-
19)/p) - 3 - 25) / 5) * y(i - 1) - (m*0.0004-0.0015) * h;
        end
        if(19<t&&t<=20)
            m=480;
            y(i) = y(i - 1) + h * k * 10^((27.71 + 9.86*cos
(pi / (12.5865) * (19-14.4096))*exp(-(t-
19)/p) - 3 - 25) / 5) * y(i - 1) - (m*0.0004-0.0015) * h;
        end
        if(20<t&&t<=21)
            m=520;
            y(i) = y(i - 1) + h * k * 10^((27.71 + 9.86*cos
(pi / (12.5865) * (19-14.4096))*exp(-(t-
19)/p) - 3 - 25) / 5) * y(i - 1) - (m*0.0004-0.0015) * h;
        end
    end
    if(y(i)<0.4)
        y(i)=0.6;
        s(j)=t;
```

```
        j=j+1;
    end
end


for i in range(1, 12000, 1):
    if (y[i - 1] < 0.4):
        y[i - 1] = 0.6
    pnum=Swpepple[i//1000]
    ploss=0.0004*pnum+0.0015
    if(i//1000<9):
        t = h * i
        y[i] = y[i - 1] + h * k * 10 ** ((T0 + Ta * math.cos(pi / 14 * (t0 + t - tm)) - m - 25) / 5) * y[i - 1] - ploss * h
        concent.append(y[i])
    else:
        t = h * i
        y[i] = y[i - 1] + h * k * 10 ** ((T0 + Ta * math.cos(pi / 14 * (t0 + t - tm))* pow(e, -(t0+t - ts) / km) - m - 25) / 5) * y[i - 1] - ploss * h
        concent.append(y[i])
```

```
import math
import random
import numpy as np
import copy
from scipy import integrate

step = 0.001
step0 = 3

stp = 0.01

t0 = 0
t1 = 5
t2 = 15
t3 = 19
t4 = 24

Ta = 7
T0 = 29.8
tm0 = 15
tm00 = 15.33195
```

```
tm1 = 15.7447
tm2 = 14.979209
w1 = -19.561598
w2 = 13.500613

k = math.atan(math.pi *(t3 - tm00)/w2) * w2 / math.pi
mannumber = [521,521,521,521,521,521,521,521,521]

T = 1
AT = 0.999
resu = []

r = []

resulist = copy.deepcopy(mannumber)

def temperature(t):
    t = t
    temperature = 0
    if t >= t0 and t <= t1:
        t = t + 24
        temperature = T0 + Ta * math.cos(math.pi * (t3 - tm00) / w2) * math.exp((t3 - t) / k)
    elif t < t2:
        temperature = T0 + Ta * math.cos(math.pi * (t - tm1) / w1)
    elif t < t3:
        temperature = T0 + Ta * math.cos(math.pi * (t - tm2) / w2)
    elif t <= t4:
        temperature = T0 + Ta * math.cos(math.pi * (t3 - tm00) / w2) * math.exp((t3 - t) / k)
    return temperature

def mannumbercount(t):
    result = 0
    if t >= 9.0 and t < 10.0:
        result = mannumber[0]
    elif t >= 10.0 and t <= 11.0:
        result = mannumber[1]
    elif t >= 12.0 and t < 13.0:
        result = mannumber[2]
    elif t >= 13.0 and t <= 14.0:
        result = mannumber[3]
```

```
62.     elif t >= 15.0 and t < 16.0:
63.         result = mannumber[4]
64.     elif t >= 16.0 and t <= 17.0:
65.         result = mannumber[5]
66.     elif t >= 18.0 and t < 19.0:
67.         result = mannumber[6]
68.     elif t >= 19.0 and t < 20.0:
69.         result = mannumber[7]
70.     elif t >= 20.0 and t <= 21.0:
71.         result = mannumber[8]
72.     else:
73.         result = 0
74.     return result
75.
76.def func(t):
77.     return math.pow(10.0,(t/5 - 5.4))
78.
79.def jifen():
80.
81.     num1 = 0
82.     i = 0
83.     while i <= t4:
84.         mesg[round(i/step)].append(num1)
85.         num1 += func(mesg[round(i/step)][0]) * step
86.         i += step
87.
88.
89.def qiujifen(x1,x2):
90.     fArea, err = integrate.quad(func, x1, x2)
91.     return fArea
92.
93.def day():
94.     tt = 0
95.     A = 0.6
96.     a = -0.1129503
97.     b = -0.1129503
98.     ttt = 0
99.     B = -0.0004
100.      v = 0
101.      state = 0
102.      tempy = 0.6
103.
104.      tempN = 0
105.      concentration = []
```

```
106.     while tt <= t4:
107.         if state == 1:
108.             tempy = A * math.exp(b * qiujifen(tt,ttt)) +
    B * mesg[round(tt/step)][1] * (tt - ttt) + 1.4754 * (tt - t
    tt)
109.             if tempy > 0.6:
110.                 tempy = 0.6
111.         elif state == 0:
112.             tempy = 0.6
113.
114.
115.         vtemp = A * b * func(mesg[round(tt/step)][0]) * t
    empy + B * mesg[round(tt/step)][1]
116.
117.
118.         if vtemp + 1.4754 >= 0 and tempy >= 0.6:
119.             v = 0
120.             if state == 1:
121.                 state = 0
122.                 A = tempy
123.                 ttt = tt
124.         else:
125.             v = vtemp + 1.4754
126.             if state == 0:
127.                 state = 1
128.                 A = tempy
129.                 ttt = tt
130.         if mesg[round(tt/step)][1] != tempN:
131.             A = tempy
132.             ttt = tt
133.         tempN = mesg[round(tt/step)][1]
134.         concentration.append([tt,tempy])
135.         tt += stp
136.     return concentration
137.
138. def newans():
139.     first = random.randint(2,5)
140.     second = random.randint(1,10)
141.     number = random.randint(1,20)
142.     if second >= 1 and second <= 3:
143.         second = 1
144.     else :
145.         second = 0
146.     return [first,second,number]
```

```
147.
148. def gai(list):
149.     global mannumber
150.     global resulist
151.     copy.deepcopy(mannumber)
152.     if list[1] == 1:
153.
154.         resulist[list[0]] = resulist[list[0]]+list[2]
155.         if resulist[list[0]] > 521:
156.             resulist[list[0]] = 521
157.     elif list[1] == 0:
158.         resulist[list[0]] = resulist[list[0]]-list[2]
159.         if resulist[list[0]] < 0:
160.             resulist[list[0]] = 0
161.     return resulist
162.
163. def count(list):
164.     global r
165.     r = [1,1,1,1,1,1,1,1,1]
166.     for x in list:
167.         if x[1] < 0.4:
168.             if x[0] >= 9.0 and x[0] < 10.0:
169.                 r[0] = 0
170.             elif x[0] >= 10.0 and x[0] <= 11.0:
171.                 r[1] = 0
172.             elif x[0] >= 12.0 and x[0] < 13.0:
173.                 r[2] = 0
174.             elif x[0] >= 13.0 and x[0] <= 14.0:
175.                 r[3] = 0
176.             elif x[0] >= 15.0 and x[0] < 16.0:
177.                 r[4] = 0
178.             elif x[0] >= 16.0 and x[0] <= 17.0:
179.                 r[5] = 0
180.             elif x[0] >= 18.0 and x[0] < 19.0:
181.                 r[6] = 0
182.             elif x[0] >= 19.0 and x[0] < 20.0:
183.                 r[7] = 0
184.             elif x[0] >= 20.0 and x[0] <= 21.0:
185.                 r[8] = 0
186.     num = 0
187.     for x in range(9):
188.         num += r[x] * mannumber[x]
189.
190.     return num
```

```
191.
192. def accept(df):
193.     global mannumber
194.     global resulist
195.     global T
196.     if df < 0:
197.         mannumber = copy.deepcopy(resulist)
198.     else:
199.         # a = random.randint(1,100000000)/100000000
200.         # if a <= math.exp(-1 * df / T):
201.         #     mannumber = copy.deepcopy(resulist)
202.         # else:
203.         resulist = copy.deepcopy(mannumber)
204.     T = AT * T
205.
206. mesg = []
207. tt = 0
208. while(tt <= t4):
209.     mesg.append([temperature(tt),mannumbercount(tt)])
210.     tt += step
211. jifen()
212. list = day()
213. re = count(list)
214. resu.append([re,mannumber])
215. while T > np.power(10.0,-20):
216.     list = day()
217.     re = count(list)
218.     accept(resu[-1][0] - re)
219.     resu.append([re,mannumber,r])
220.     print([re,mannumber ,r])
221.     li = newans()
222.     gai(li)
```