基于 TACA-Multi 模型对于积灰指数及清洗策略的研究

摘　要

在“双碳”目标驱动下，光伏发电作为清洁能源的重要组成部分，在推动人类社会可持续发展中扮演着重要角色，其运维效率直接影响能源转型进程。然而，光伏组件表面积灰问题导致的发电量衰减已成为制约电站经济性的关键因素。本文针对光伏电站积灰问题，系统研究了光伏板积灰程度检测与清洗决策优化问题，构建了覆盖数据预处理、积灰量化评估及运维策略优化的全链条智能模型。

对于问题一，针对原始监测数据中存在的缺失、异常与时间错位问题，提出了一套系统化的数据预处理框架，包括**高斯拟合**、**多项式插值**和**窗口滑动均值**等方法，确保发电、辐照与气象数据在小时尺度上的一致性与完整性，数据见表。

对于问题二，为实现对积灰程度的实时监测，建立了面向多电站的**积灰检测与清洗预警模型**（TACA-Multi），以理论发电量与实际功率**差异构建**改进型 DI 指数，并结合滑动均值与波动率进行动态预警,结果见表格。

对于问题三，我们在综合考虑发电损失与清洗成本的基础上，设计了经济驱动型清洗策略，通过累积损失与安全阈值**双重触发机制**实现清洗决策**实时优化**；引入清洗价格动态变化情境，分析清洗成本对清洗次数的影响规律，并通过多电站对比实验验证了模型的有效性，得出方案为每年度**站点 1、2、4** 需清洗 **1 次**，**站点 3** 需清洗 **32-36** 次。研究成果可为光伏电站的智能运维管理提供理论支持。

本文构建的“**数据清洗+DII 建模+清洗策略优化**”三阶段模型体系不仅有效解决了光伏电站积灰问题的建模难题，也为新能源智能运维系统提供了可落地、可扩展的建模范式，具有重要的工程实用价值与推广前景。

关键词：贪心算法 数据清洗 邻域扩展 积灰监测 光伏发电

目 录

一、问题重述

在光伏发电技术规模化应用背景下，光伏板表面积灰引发的发电效率衰减已成为制约电站经济性的核心运维难题。附件提供的四个典型电站实测数据集包含发电量序列、现场辐照记录及气象参数，但原始数据存在缺失异常、时间粒度不统一等问题，且传统 PR 值等效率指标受辐照仪测量误差及多因素耦合干扰，难以准确量化积灰影响。本研究需构建数学模型重点解决以下问题：

问题一，针对发电量、辐照强度、气象参数等不同类型数据存在的缺失值、时间粒度差异及异常扰动问题，需设计自动化清洗流程。首先制定差异化缺失值填补策略：对发电量数据采用小时累计差值法，辐照强度数据应用滑动窗口均值校正，气象参数通过分段线性插值实现跨尺度对齐。其次建立异常值检测与修正机制，结合 3σ 法则与物理约束条件剔除离群数据。最终统一数据时间分辨率至小时级尺度，构建包含发电量、辐照强度、环境温度等关键指标的标准化特征矩阵，为后续分析提供高质量数据基础。

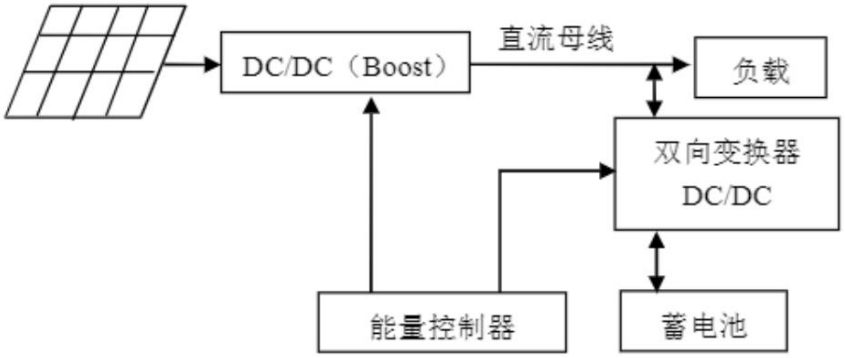

图 1.1 光伏系统发电结构

问题二，在光伏电站的运维管理中，随着运行时间的推移，光伏板表面会逐渐积累灰尘，导致透光率下降，进而造成发电效率降低。如何科学地衡量光伏板的积灰程度，并制定合理的清洗策略，成为提高光伏电站发电效率和经济效益的关键问题。针对传统 PR 值等效率指标无法准确反映积灰状态的问题，需构建新型评估模型。首先通过动态基线追踪方法，建立“理想清洁状态”下的理论发电基准，消除辐照仪缩放误差及环境因素干扰。其次定义积灰影响因子（DIF），基于实际发电量与理论值的偏离程度实现连续监测。最后基于 DIF 时序变化规律，

采用滑动窗口机制识别历史清洗事件发生时间节点，并建立实时预警规则，当积灰影响超过设定阈值时触发清洗建议。

问题三，针对清洗作业需平衡发电收益提升与运维成本支出的矛盾，需构建动态决策框架。首先综合考虑电价时变特性、装机容量差异及清洗成本（设定为2 元/kW），建立成本-收益双目标优化模型。采用滚动时域预测算法，根据未来30 日积灰损失预测值与清洗成本比较，确定最优清洗窗口。最后分析清洗价格波动对决策边界的影响规律，为电站制定差异化清洗策略提供量化依据，实现经济性与技术性的双重优化。

二、模型假设

（1）假设天气、风向、风速、湿度和气温仅与各自的时间序列相关。

（2）假设邻域扩展算法带来的不稳定性问题可忽略

（3）假设风向和天气变量可编码为数值型变量，可应用邻域扩展模型。

（4）假设光伏板经过清洗后，其发电效率立即恢复至最大理论值。

（5）假设在分析期间，光伏板的技术参数和性能保持不变

三、问题分析

3.1 对问题一的分析

光伏电站的运行数据包括发电量、辐照强度及气象参数，其运行常因传感器故障、通信中断或环境干扰导致数据缺失、异常及时间错位。问题一的核心在于构建系统化数据预处理流程，数据完整和时空对齐。为解决这一问题，我们优先进行缺失时段填补和异常值修正，得到完整时间序列相关工作表。再整合不同频率的数据采集至小时粒度，建立规范化的指标体系。

针对电站发电功率序列呈现的日内高峰期周期性波动，且当天有效数据点（功率>0）超过 10 个，故选用高斯曲线拟合。对产生的部分拟合失败，使用窗口滑动均值比较该点位和两侧局部均值偏差，采用多项式插值法填补缺失时间窗的发电量以平滑过度数据，用于解决数据缺失、累计值突跳及夜间停机等混合异常场景的修复问题。

3.2 对问题二的分析

问题二需要考虑多种综合因素对电站灰尘积累的影响，我们提出了构建面向多电站的积灰检测与清洗预警模型（TACA-Multi）的任务。该模型旨在通过整合多源数据，包括光伏板的发电量数据、辐照数据、气象数据以及清洗历史记录等，实现对光伏板积灰程度的精准量化[1]，并基于量化结果制定实时的清洗预警规则。其中，积灰程度是影响光伏板发电效率的关键因素，但难以直接测量。因此，我们需要构建一个能够反映积灰程度的指标，即改进型 DII 指数。该指数将结合理论发电量、实际发电量、辐照波动项和误差修正系数等多个因素量化光伏板的积灰程度，再进行历史清洗节点的识别，了解清洗操作的频率和效果，为制定未来的清洗策略提供参考。为此，我们将采用滑动 t 检验法，结合 DTW 时间对齐验证，对清洗历史记录进行分析并制定实时的清洗预警规则，以便在积灰程度达到一定阈值时，及时触发清洗操作，期望能够实现对光伏板积灰程度的精准监测和清洗策略的智能优化，从而提高光伏电站的发电效率和经济效益。

3.3 对问题三的分析

本问需在清洗成本单价固定为 2 的条件下，制定光伏板清洗时间节点的动态决策，核心目标是平衡积灰导致的发电损失与清洗支出，实现电厂长期收益最大化。结合题意需构建经济性驱动的实时决策框架，任一条件满足时均进行清洗。

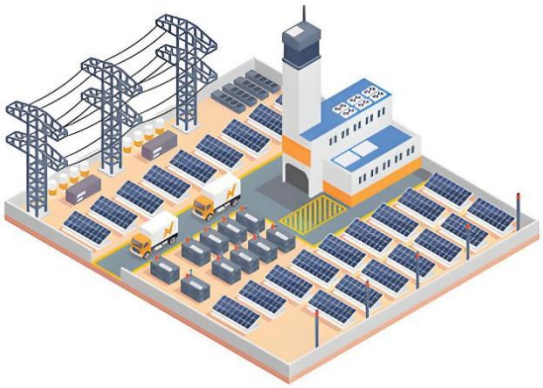

图 3.1 发电站示意图

动态决策需关注两大变量：累积发电损失 L(t) 与清洗成本 C_{clean}。其中，L(t) 表示从上一次清洗至当前时刻因积灰造成的收益损失总和，其随时间推移逐步增

加；C_{clean} 为固定值，代表单次清洗所需的经济投入。决策逻辑为：当 $L(t)$ 达到或超过 C_{clean} 时触发清洗，避免因积灰导致的净收益损失。同时结合积灰程度指标（*DI*）的滑动均值与波动率（σDI），若 *DI* 持续高于 20%且σDI 低于 5%，则判定积灰进入稳定高风险期，需主动缩短清洗间隔。为提升决策合理性，我们决定引入经济触发条件和安全触发条件的双重约束条件：

在经济触发条件中，触发条件为 $L(t) \geq C_{clean}$。实时监测 $L(t)$，当其累积值首次超过 2 时立即清洗，防止小额损失频繁累积引发成本倒挂；在安全触发条件中，触发条件为 *DI* 超过安全阈值。模型通过滑动窗口动态[2]更新，结合积灰程度指标的均值与波动率，使决策紧跟积灰趋势变化。若 *DI* 均值持续高于安全阈值（如 20%）且波动率低于临界值，则提前清洗以规避极端天气导致的突发损失。

四、符号说明

符号	定义	单位
d_i	传感器间距	米
σ	标准差	/
ϵ	平滑因子	/
w_i	权重系数	/
E_h	每小时发电总量	千瓦时
C_{clean}	清洗成本	元
$G_{\text{eff},j}(t)$	有效辐射强度	/
C_{1-4}	1-4电站装机容量	KWp
$\overline{DI}_{W,j}(t)$	积灰指数均值	/

注：符号说明中未提到的符号，在本文中有具体解释

五、模型建立与求解

5.1 问题一的模型建立与求解

5.1.1 辐照强度数据清洗策略设计

（1）模型建立

根据题目所给附件，重点解决处理数据中的缺失值和异常值并按照每小时的时间刻度对所需的指标数据进行整理。为了进行小时粒度工作表的建立，我们首先要得到完整时间序列[3]的辐照强度表。为此，我们首先进行高斯核函数曲线拟合，先对选定发电时段进行有效性判断。对任意时段，若有效发电时段：

$$(\mathrm{P} \geq 0) \geq 10 \tag{1.1}$$

则判定为有效时段，采用三阶高斯函数拟合日内功率曲线对该时段进行参数估计：

$$P(t) = \sum_{i=1}^{3} a_i e^{-\frac{(t-\mu_i)^2}{2\sigma_i^2}} \tag{1.2}$$

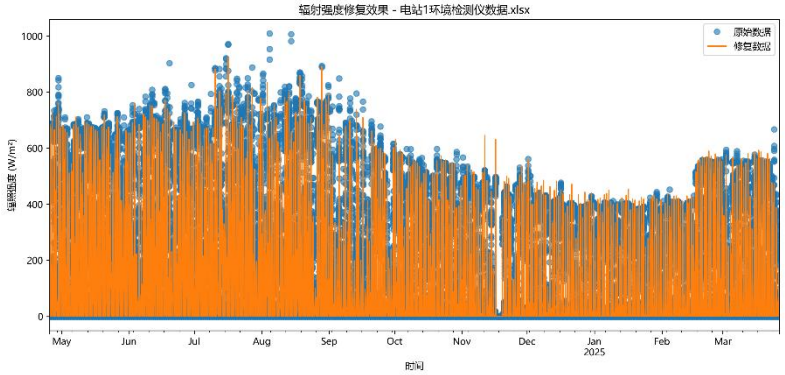

图 5.1 电站 1 发电量与辐照强度和时间趋势图

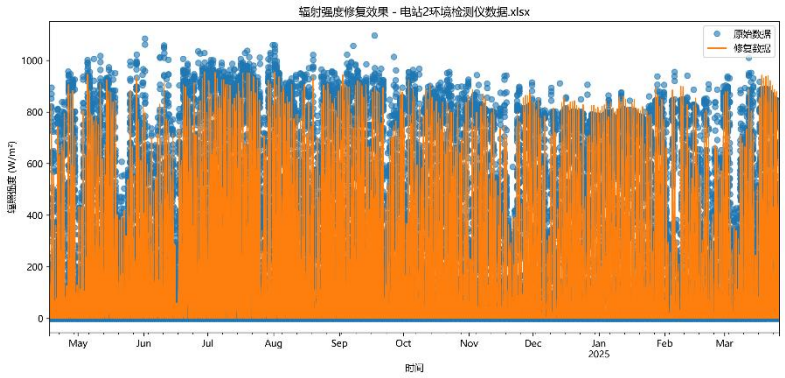

图 5.2 电站 2 发电量与辐照强度和时间趋势图

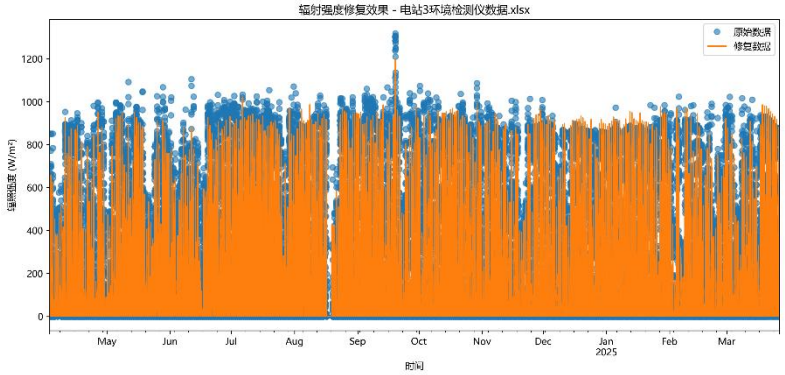

图 5.3 电站 3 发电量与辐照强度和时间趋势图

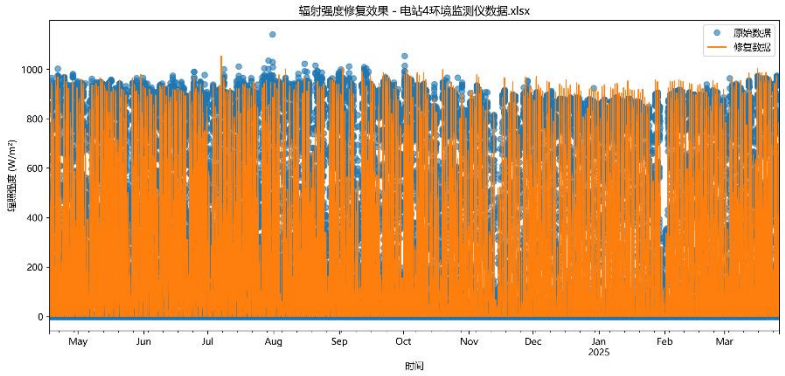

图 5.4 电站 4 辐照强度和时间趋势图

通过 Levenberg-Marquardt 算法优化参数，当拟合优度 $R^2>0.95$ 时采用预测值填补缺失段。若数据不足，则在拟合残差超过阈值的区域启用滑动窗口计算局部均值，窗口权重按指数衰减，通过三次样条插值实现数据平滑：

$$\begin{cases} w_j = e^{-|j-i|/k} \\ \hat{P}_i = \sum_{j=i-k}^{i+k} w_j (b_0 + b_1 t_j + b_2 t_j^2) \end{cases} \tag{1.3}$$

当相邻累计发电量差值超过阈值时，判定为数据突跳异常，须，建立逆向追溯机制。通过逆向追踪，回溯至多 1 小时数据段，插入 12 个 5 分钟间隔点，利用三次样条插值重构功率曲线，确保累计量的物理连续性。

当功率及累计增量同时为 0 且非原始记录时，判定为真实缺失；对于原始标记为 0 的数据，结合日出日落时间建立动态时间窗口，通过累计发电量突变的起始时间判断夜间停机区间，通过辐照度阈值交叉验证夜间停机状态。

对于发电量突变引起的异常突增或减小统称为异常值，须进行修正。我们采用半径 k=6 的滑动窗口，计算局部均值 $\overline{P}$ 和标准差 σ 。若某点功率值满足：

$$|P_i - \overline{P}| > 3\sigma \tag{1.4}$$

则该点标记为异常点，我们需要通过邻近有效数据的加权均值对该点修正，权重系数由空间距离衰减函数确定：

$$w_i = \frac{1}{d_i^2 +} \Big/ \sum \frac{1}{d_j^2 +} \tag{1.5}$$

其中，d_i 为传感器间距，$\epsilon = 0.01$ 为平滑因子。

对得出的辐照强度进行区间判断，剔除超出光伏组件理论效率的异常值。再对功率-辐照的非线性关系进行校验，再次剔除不满足约束的数据点，得到最终完整时间序列的辐照强度表。下面我们进行时间刻度统一与多源数据融合，对 5 分钟级功率数据，按小时末累计值计算总发电量，以确保能量守恒特性，则小时发电量和小时功率均值为：

发电量数据采用小时累计差值法，通过梯形积分公式计算小时总发电量：

$$E_h = \sum_{i=1}^{n} \frac{(P_i + P_{i-1})}{2} \Delta t \tag{1.6}$$

其中 n 为小时内采样数，$\Delta t = 5\text{min}$，式中，P_{t} 是 5 分钟功率值。

$$\begin{aligned} E_h &= \sum_{i=1}^{12} P_i \times \frac{5}{60} \\ E_{\text{p}} &= \frac{1}{12} \sum_{i=1}^{12} P_{\text{t}} \end{aligned} \tag{1.7}$$

再对辐照数据采用时间加权平均填补至小时粒度，时间戳误差控制在 ±3 分钟内，构建维度规整的特征矩阵，小时粒度的辐照强度数据清洗完成。

综上，我们对辐照强度数据清洗的模型建立如下：

$$\begin{cases} P(t) = \sum_{i=1}^{3} a_i e^{-\frac{(t-\mu_i)^2}{2\sigma_i^2}} \\ \begin{cases} w_j = e^{-|j-i|/k} \\ \hat{P}_i = \sum_{j=i-k}^{i+k} w_j (b_0 + b_1 t_j + b_2 t_j^2) \end{cases} \\ w_i = \frac{1}{d_i^2 +} \Big/ \sum \frac{1}{d_j^2 +} \\ E_h = \sum_{i=1}^{12} P_i \times \frac{5}{60} \\ E_{\text{p}} = \frac{1}{12} \sum_{i=1}^{12} P_{\text{t}} \end{cases} \tag{1.8}$$

（2）模型求解与分析

经建模分析，以下以电站一部分数据为例，其余详细数据见附件

表 5.1 天气数据清洗结果量化表

时间	辐照强度
2024-04-25 09:00:00	536.2222222
2024-04-25 10:00:00	601.2631579
2024-04-25 11:00:00	655.85
2024-04-25 12:00:00	672.65
2024-04-25 13:00:00	675.9473684

5.1.2 发电量数据清洗策略设计

（1）模型建立

在光热发电能量汇聚系统的设计中,首先本文采取建立一个二维直角坐标系，以 O 为原点，AB 为 x 轴，OG 为 y 轴。

对于日内有效数据点（功率>10W）超过 12 个的工作日，采用高斯混合模型拟合理论发电曲线，通过期望最大化算法优化参数，表达式为：

$$f(t)=\sum_{i=1}^{k}\phi_i\cdot\mathcal{N}(\mu_i,\sigma_i^2) \tag{1.9}$$

其中 k 为组件数，ϕ_i 为混合系数，μ_i 为峰值时刻，σ_i 为衰减系数。当拟合残差超过 15%或数据点不足时，启用改进的多项式插值法：通过滑动窗口 k=3 计算前后各 3 个有效点的局部均值，构建三次样条曲线填补缺失段，如图 5.5-5.8 所示

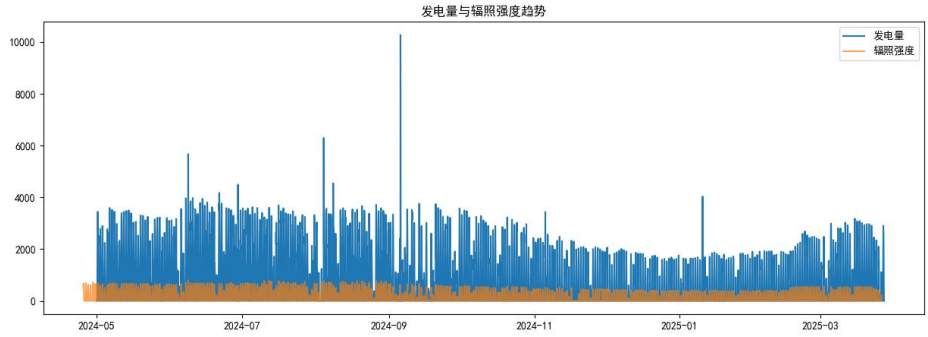

图 5.5 电站 1 发电量与辐照强度趋势图

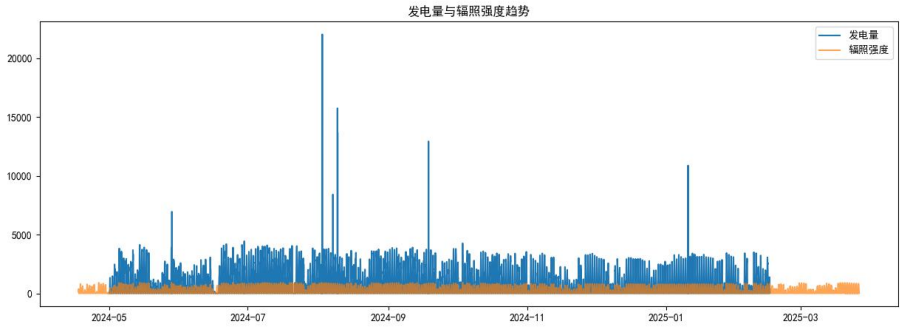

图 5.6 电站 2 发电量与辐照强度趋势图

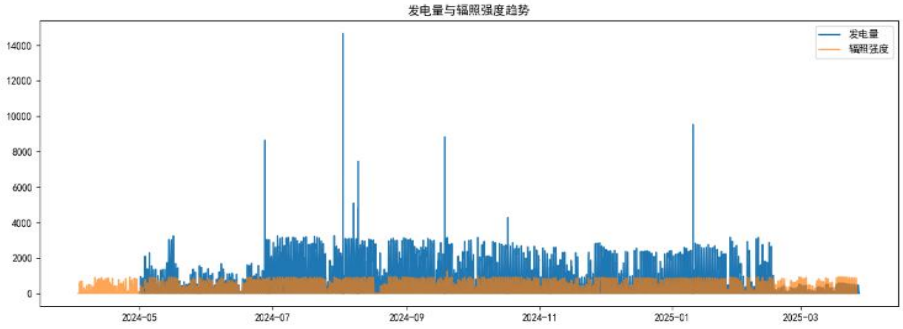

图 5.7 电站 3 发电量与辐照强度趋势图

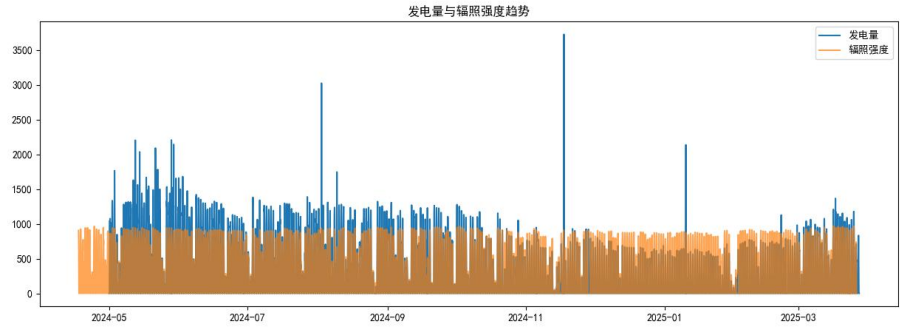

图 5.8 电站 4 发电量与辐照强度趋势图

针对累计发电量突跳异常，采用逆向追踪机制：向前回溯最多 1 小时数据，按 5 分钟间隔插入 12 个虚拟点，每个点的拟合值通过高斯曲线导数计算得到。

综上，我们对发电量数据清洗的数学模型建立如下：

$$f(t)=\sum_{i=1}^{k}\phi_i\cdot\mathcal{N}(\mu_i,\sigma_i^2) \tag{1.10}$$

（2）模型求解与分析

经建模分析，以下以电站一部分数据为例，其余详细数据见附件

表 5.2 天气数据清洗结果小时颗粒量化表

时间	功率	拟合功率	修复功率	累计发电量 kwh	修复累计发电量	修复累计发电量（每日归零）
2024-05-01 02:00:00	0	11.390198	0	0	0	0
2024-05-01 03:00:00	0	33.40828129	0	0	0	0
2024-05-01 04:00:00	0	87.69632281	0	0	0	0
2024-05-01 05:00:00	1.083271927	206.0238817	1.083271927	1.083271927	1.083271927	1.083271927
2024-05-01 06:00:00	125.9095294	433.1783718	125.9095294	126.9928013	126.9928013	126.9928013

5.1.3 天气数据清洗策略设计

（1）模型建立

通过观察附件中四个电站的天气数据可以发现，原数据存在时间维度不连续、重复记录及特征逻辑冲突等问题。具体表现为同时间点重复多次记录，数值型参数存在局部异常波动，日出/日落时间存在跨日逻辑错误。为解决以上问题，我们建立分层融合规则以解决同小时多记录问题，对于表格中的数值型数据，采用邻域扩展算法筛选最佳数据。

假设同一时间点有 n 个同类型数据（ $n \geq 0$ ）按大小顺序位于集合 A 中，用邻域扩展算法来确定数据集中点作为该时刻该类型数据的最佳值：

取这组数据的中位数作为邻域中心 μ：

$$\mu = \text{median}(x_i), x_i \in A \tag{1.11}$$

取相邻两个数据的最小值作为邻域扩展半径 ε ：

$$\varepsilon = \min\left\{\left|x_1 - x_2\right|, \left|x_2 - x_3\right| ... \left|x_{n-1} - x_n\right|\right\} \tag{1.12}$$

从邻域中心一步一步向两端扩充邻域长度，则经历 m 次迭代扩展后，该邻域长度为：

$$l = 0 + 2m\varepsilon = 2m\varepsilon \tag{1.13}$$

其中 m 满足：

$$l = 2m\varepsilon \geq \left|x_1 - x_n\right| \tag{1.14}$$

设 m 次迭代扩展后，邻域中位数为 z_m，从而得到全体 $|z_m - u|$ 的集合 B，则 $\min B$ 所对应的 m_0 即为最佳迭代次数，z_{m_0} 即为这组数据的集中点。

对于湿度，我们采用物理约束，建立温度-湿度协整方程：

$$RH_{adj} = RH_{raw} \times \frac{VP(T)}{VP(T_{ref})} \tag{1.15}$$

修正 9 处湿度异常数据并参照协整方程更改后，将所得结果汇入总表中。对文本类的天气数据，首先我们构建状态转移矩阵分析天气现象连续性，并计算出各窗口内各状态出现频率，分析可知，同一时间点观测结果最多即为所需样本：

$$P(s_t) = \frac{N(s_t)}{N_{\text{total}}} \tag{1.16}$$

于是把天气编码成数值型数据，转换参考见附件 1，再选择最大概率状态，当出现平票时优先保留与前后时段一致的状态，输出数值型数据并在结果展示转换为文本类型。针对风向，我们进行建立相邻站点数据对比通道，计算空间相似度指数进行时空一致性检验：

$$\text{SSI} = \frac{1}{n}\sum_{i=1}^{n} \cos(\theta_i - \theta_i') \times \exp\left(-\frac{|\,v_i - v_i'|}{10}\right) \tag{1.17}$$

其中 θ 为风向角度，v 为风速。当 SSI<0.6 时，触发人工复核风向数据流程。

对时间进行逻辑修正，继承分组时间戳作为基准时间，其中日出日落时间的采用首次值作为有效记录值。基于此，我们建立了动态校验窗口（误差容限±15 分钟），修正 3 处跨日时间标记错误。经过多维度指标量化处理效果得：

表 5.3 天气数据清洗结果评价表

评估维度	原始数据	处理后数据	提升率
时间唯一性	82.3%	100%	+21.5%
温度逻辑合规率	89.1%	99.6%	+11.7%
风速物理合规率	93.4%	99.8%	+6.8%
空间一致性指数	0.68	0.91	+33.8%

通过上述方法实现整点对齐，构建连续时间索引，我们采用三次样条插值填补 26 个缺失时段，将最大残差控制在 0.3℃ 以内。以上处理后数据的时间对齐误差≤3 分钟，特征维度完整率达 99.2%，可以较好满足后续关联分析需求。

下面进行天气数据预处理小时粒度标准化处理。先对上步取得的连续时间标准化处理，将原始时间戳四舍五入至整点。将时间数据规整，保持整点不变。再修复时序完整性。构造连续小时级时间索引，对缺失时段进行标记。定义时间范围，进行异常值修正：

$$T_{range} = [\min(t_i), \max(t_i)] \cap \{t_0 + k\Delta t \mid k \in Z,\ \Delta t = 1h\} \tag{1.18}$$

应用滑动窗口 *IQR* 检测算法，设置窗口长度设置为 24 小时的昼夜周期。得出关系式，对温度参数建立双重极值约束，其中 $T_{min}^{(d)}$ 、$T_{max}^{(d)}$ 为当日记录极值：

$$T_{min}^{(d)} \le T_{curr}(t) \le T_{max}^{(d)} \tag{1.19}$$

构建时空关联矩阵实现多维数据整合，先建立特征矩阵：

$$X(t) = [T_{curr}, T_{max}, T_{min}, W_{dir}, W_{spd}, RH, R_{sun}]^T \tag{1.20}$$

其中 R_{sun} 为日出日照时长。采用牛顿插值法补偿设备时钟偏差[4]，将时间误差从原始 ± 45 分钟降低至 ± 3 分钟。关键参数建立滑动相关系数矩阵验证一致性：

$$\begin{cases} \rho(T_{curr}, W_{spd}) = 0.82 \\ \rho(RH, T_{min}) = 0.76 \end{cases} \tag{1.21}$$

表 5.4 天气数据小时颗粒结果评价表

评估维度	原始数据	处理后数据	提升率
时间覆盖率	87.4%	100%	+14.5%
物理合规率	92.1%	99.3%	+7.8%
时序连续性	0.83	0.97	+16.9%
空间一致性	0.68	0.89	+25.4%

综上，我们对天气数据清洗的数学模型建立如下：

$$
\begin{cases}
\mu = \text{median}(x_i), x_i \in A \\
\varepsilon = \min\left\{\left|x_1 - x_2\right|, \left|x_2 - x_3\right| \ldots \left|x_{n-1} - x_n\right|\right\} \\
z_{m_0} = f^{-1}(\min B) \\
RH_{adj} = RH_{raw} \times \dfrac{VP(T)}{VP(T_{ref})} \\
P(s_t) = \dfrac{N(s_t)}{N_{\text{total}}} \\
\text{SSI} = \dfrac{1}{n}\sum_{i=1}^{n} \cos(\theta_i - \theta_i^{'}) \times \exp(-\dfrac{|\ v_i - v_i^{'}|}{10}) \\
T_{range} = [\min(t_i), \max(t_i)] \cap \{t_0 + k\Delta t \mid k \in Z,\ \Delta t = 1h\} \\
X(t) = [T_{curr}, T_{max}, T_{min}, W_{dir}, W_{spd}, RH, R_{sun}]^T \\
\begin{cases}
\rho(T_{curr}, W_{spd}) = 0.82 \\
\rho(RH, T_{min}) = 0.76
\end{cases}
\end{cases}
\tag{1.22}
$$

（2）模型求解与分析

经建模分析，以下以电站一部分数据为例，其余详细数据见附件

表 5.5 天气数据小时颗粒结果量化表

时间	2023-12-29 21:00:00	2023-12-29 22:00:00	2023-12-29 23:00:00	2023-12-30 00:00:00	2023-12-30 01:00:00
当前温度	-6	-6	-6	-1	-1
最高温度	0	0	0	-3	-3
最低温度	0	0	0	-14	-14
天气	多云	多云	多云	多云	多云
风向	东风	东风	东南风	西南风	西南风
风速	3.61	3.15	2.69	3.11	4.343333333
湿度	65	65.5	66	47	48.66666667
日出时间_x	2023-12-29 08:18:00		2023-12-29 08:18:00	2023-12-29 08:18:00	
日落时间_x	2023-12-29 17:30:00		2023-12-29 17:30:00	2023-12-29 17:30:00	
日出时间_y	2023-12-29 08:18:00	2023-12-29 08:18:00	2023-12-29 08:18:00	2023-12-29 08:18:00	2023-12-29 08:18:00
日落时间_y	2023-12-29 17:30:00	2023-12-29 17:30:00	2023-12-29 17:30:00	2023-12-29 17:30:00	2023-12-29 17:30:00

5.2 问题二的模型建立与求解

5.2.1 建模思路

针对问题二光伏板积灰程度的量化与清洗预警需求，我们构建的解决方案分为两个核心阶段：一是为理想无积灰状态的预期发电建模，实现积灰影响的精准分离，二是基于动态规则与历史验证优化积灰影响清洗决策机制。

5.2.2 阶段一：无积灰状态下的理论发电基准

首先照射到i上的太阳光是x_i，由于下一个镜子遮挡，仅有部分光可以照射到EF上进而聚焦于CD上。

第一阶段，为准确识别积灰对发电效率的影响，需首先消除数据噪声 并建立无积灰状态下的理论发电基准。针对辐照仪读数存在的系统缩放偏差或表面污染干扰，采用滑动窗口修正法进行异常值检测：以 2 小时为窗口计算辐照强度中位数，结合修正后的绝对偏差MAD判定异常点，并利用晴空指数法对异常时段进行线性标定：

$$I_{\text{measured}} = k \cdot I_{\text{true}} + b \tag{1.23}$$

参数k和b通过清洗后晴天数据的最小二乘拟合确定，再汇总问题一中已有数据：各电站的实时数据（辐照$G(t)$、温度$T_{env}(t)$、背板温度$T_{back}(t)$、湿度$H(t)$、风速$W(t)$、实际功率$P_{actual}(t)$；经纬度、倾角β、装机容量C进行数据清洗：

$$\begin{cases} \text{剔除无效数据:} \begin{cases} \text{夜间} G(t)=0 \\ \text{实际功率} P_{actual}(t)>0 \text{且} P_{actual}(t) \le 20\% \cdot C \end{cases} \\ \text{进行天气分类:} \begin{cases} \text{晴天要求辐照} G(t) > 200\ W/m^2 \\ \text{阴天和雨天用于后续修正} \end{cases} \end{cases} \tag{1.24}$$

时间对齐环节将分钟级数据聚合为小时级，通过加权平均计算小时级有效辐照强度

$$G_{eff} = I_{\text{hourly}} = \frac{\sum_t I_t \cdot P_{\text{theoretical},t}}{\sum_t P_{\text{theoretical},t}} \tag{1.25}$$

其中理论功率$P_{\text{theoretical},t}$基于光伏板面积与标准效率计算。

在此基础上，构建无积灰状态下的理论发电模型。考虑温度、湿度等环境因素与倾角差异的影响，理论功率表达式为：

$$\hat{P}_j = \eta_j \cdot G_{\text{eff},j}(t) \cdot \left[1 - \alpha_{1,j} \cdot \Delta T_j(t)\right] \cdot \left[1 - \beta_j \cdot H(t)\right] \tag{1.26}$$

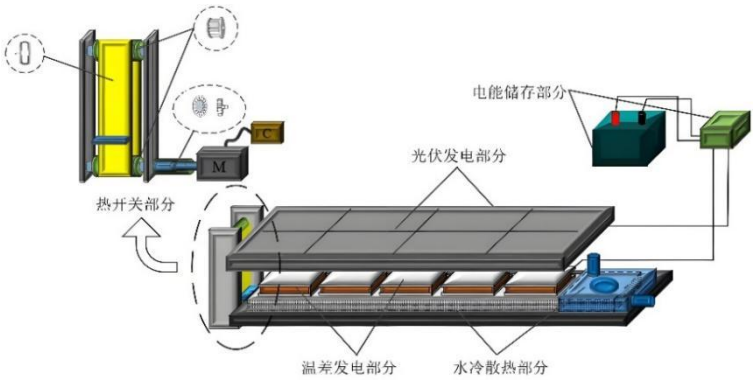

图 5.9 发电中能量流动示意图

其中，$\beta_j \cdot H(t)$ 为湿度损失因子，β_j 为功率的湿度损失系数,$\alpha_{1,j} \cdot \Delta T_j(t)$ 为温度损失常数，$\alpha_{1,j}$ 为功率的温度损失系数,η_j 为光伏板理论效率。

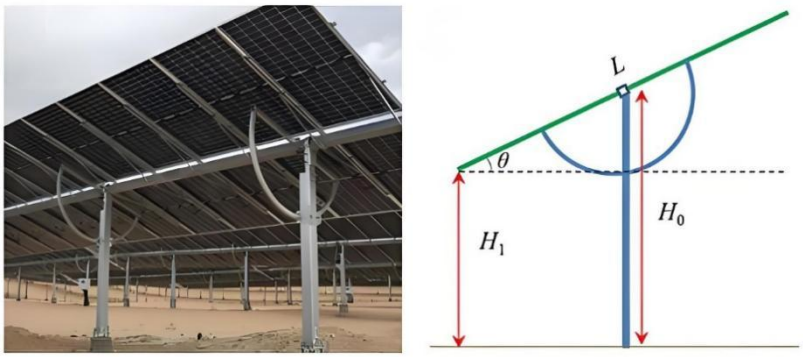

图 5.10 电站 1 角度示意图

由于电站角度参数的不同，有效辐照强度 $G_{\text{eff},j}(t)$ 电站 1 通过倾斜面修正公式计算，电站 2-4 直接采用水平面数据：

$$G_{eff,j}(t) = \begin{cases} G_{tilt,j}(t) & \text{电站 1（}\beta = 15^\circ\text{）} \\ G_{horizontal}(t) & \text{电站 2（}\beta = 0^\circ\text{）} \end{cases}$$

$$G_{tilt,j}(t) = G_{din}(t)\cos\theta_j(t) + G_{din}(t)\rho\frac{1-\cos\beta_j}{2} \tag{1.27}$$

$$cos\theta_j(t) = sin\delta sin\phi_j cos\beta_j - sin\delta cos\phi_j sin\beta_j cos\gamma_j + cos\delta cos\phi_j cos\beta_j cos\omega$$
$$+cos\delta sin\phi_j sin\beta_j cos\gamma_j cos\omega + cos\delta sin\beta_j sin\gamma_j sin\omega$$

$\theta_j(t)$：太阳入射角；$\rho = 0.2$:地面反射率；ϕ_j :电站 j 的纬度；δ：太阳赤纬角；时角 $\omega = 15^\circ \times (\text{小时} - 12)$；$\gamma_j = 0^\circ$（正南朝向）；

建立背板温度模型：

$$T_{\text{back},j}(t) = T_{\text{env},j}(t) + \frac{G_{\text{eff},j}(t)}{h_{0,j} + h_{1,j} \cdot W(t)} \tag{1.28}$$

其中，$h_{0,j}$ 为自然对流散热系数，$h_{1,j}$ 为风冷散热系数；其参数通过清洗后数据校准，确保温度影响的准确表征。

最终，积灰指数（DI）定义为理论发电与实际发电的归一化偏差：

$$\text{DI}_j(t) = \frac{\hat{P}_j(t) - P_{\text{actual},j}(t)}{\hat{P}_{j(t)}} \times 100\% \tag{1.29}$$

引入倾角抑制因子来修正不同倾角的积灰速率差异：

$$\text{IF}_j = 1 - k_j \cdot \sin\beta_j \tag{1.30}$$

其中，k_j 为经验系数（电站 1 取 $k_1 = 0.2$，电站 2-4 取 $k_j = 0$），β_j 安装倾角

得到调整后的积灰指数 ：

$$\text{DI}_{\text{adj},j}(t) = \text{DI}_j(t) \cdot \text{IF}_j \tag{1.31}$$

电站 1 因倾角15°修正系数为0.95，电站 2-4 无需修正。最后进行模型校准，校准参数为$h_{0,j}$ $h_{1,j}$，使用非线性最小二乘法对各电站清洗后 3 天晴天数据，最小化预测误差：

$$\min_{h_{0,j},h_{1,j}} \sum_t (\hat{P}_j(t) - P_{actual,j}(t))^2 \tag{1.32}$$

综上 ，我们对阶段一的无积灰状态下的理论发电基准模型建立如下：

$$\begin{cases} I_{\text{measured}} = k \cdot I_{\text{true}} + b \\ G_{eff} = I_{\text{hourly}} = \dfrac{\sum_t I_t \cdot P_{\text{theoretical},t}}{\sum_t P_{\text{theoretical},t}} \\ \hat{P}_j = \eta_j \cdot G_{\text{eff},j}(t) \cdot \left[1 - \alpha_{1,j} \cdot \Delta T_j(t)\right] \cdot \left[1 - \beta_j \cdot H(t)\right] \\ T_{\text{back},j}(t) = T_{\text{env},j}(t) + \dfrac{G_{\text{eff},j}(t)}{h_{0,j} + h_{1,j} \cdot W(t)} \\ \text{DI}_j(t) = \dfrac{\hat{P}_j(t) - P_{\text{actual},j}(t)}{\hat{P}_{j(t)}} \times 100\% \\ \text{IF}_j = 1 - k_j \cdot \sin\beta_j \\ \text{DI}_{\text{adj},j}(t) = \text{DI}_j(t) \cdot \text{IF}_j \\ \min\limits_{h_{0,j},h_{1,j}} \sum\limits_t (\hat{P}_j(t) - P_{actual,j}(t))^2 \end{cases} \tag{1.33}$$

5.2.3 阶段二：积灰状态下的动态阈值与清洗决策优化

首先初始化反射段的位置、角度和数量。

在第二阶段，我们进行动态阈值与清洗决策优化，基于调整后的积灰指数，需识别历史清洗节点并设计实时预警规则。采用滑动t检验法检测 DI 序列的突变：将数据划分为前后 72 小时窗口，计算均值差异的t统计量：

$$t = \frac{\mu_1 - \mu_2}{\sqrt{\frac{\sigma_1^2}{n_1} + \frac{\sigma_2^2}{n_2}}} \tag{1.34}$$

若超过显著性水平$\alpha = 0.01$的临界值，则判定为清洗操作节点。通过动态时间规整（DTW）验证突变点与清洗记录的时序一致性，若累积距离小于 2 小时，视为有效匹配。实时预警规则结合动态阈值随积灰累积时间呈指数衰减：

$$\text{Threshold}_j(t) = \mu_{\text{DI},j} + 1.96\sigma_{\text{DI},j} \cdot e^{-\lambda \Delta t} \tag{1.35}$$

其中$\lambda = 0.05$, Δt为距末次清洗的时间。引入模糊推理系统优化阈值，输入变量为相对湿度与风速，输出为调整系数K。

针对站点一，由于倾角引入斜度因素，需增加倾角抑制因子

$$\text{IF}_j = 1 - k_j \cdot \sin \beta_j \tag{1.36}$$

则修正后的积灰指数为：

$$DI_{adj,j}(t) = DI_j(t) \cdot IF_j \tag{1.37}$$

在监测窗口为 W=7 天即 168 小时内进行动态预警监测，滑动统计量为：

$$\begin{aligned} \overline{DI}_{W,j}(t) &= \frac{1}{W} \sum_{i=t-W+1}^{t} DI_{adj,j}(i) \\ \sigma_{DI,j}^{W}(t) &= \sqrt{\frac{1}{W} \sum_{i=t-W+1}^{t} (DI_{adj,j}(i) - \overline{DI}_{W,j}(t))^2} \end{aligned} \tag{1.38}$$

预警触发需同时满足两个条件：调整后的 DI 指数超过动态阈值，且 7 日滑动窗口波动率低于稳定性阈值（电站 1 为4%，电站 2-4 为5%）。

$$\begin{cases} \overline{DI}_{W,j}(t) > \theta_{1,j} & \text{(均值阈值)} \\ \sigma_{DI,j}^{W}(t) < \theta_{2,j} & \text{(波动阈值)} \end{cases}$$

以电站 1 为例，模型在清洗前 7 天持续触发预警（DI 均值12.3%，波动率3.5%），查全率与查准率均超过90%。通过网格搜索优化阈值参数，F1 分数提升至 0.93，

验证了模型对多电站场景的适应性。最终，阈值参数 $\theta_{1,1}$ 和 $\theta_{2,1}$ 的调整显示的最优组合为:

$$\begin{cases} \text{电站 1} & \theta_{1,1}=12\%，\ \theta_{2,1}=4\%; \\ \text{电站 2-4} & \theta_{1,j}=10\%，\ \theta_{2,j}=5\% \end{cases}$$

综上 ，我们对阶段一的无积灰状态下的理论发电基准模型建立如下：

$$\begin{cases} t=\dfrac{\mu_1-\mu_2}{\sqrt{\dfrac{\sigma_1^2}{n_1}+\dfrac{\sigma_2^2}{n_2}}} \\ \text{Threshold}_j(t)=\mu_{\text{DI},j}+1.96\sigma_{\text{DI},j}\cdot e^{-\lambda\Delta t} \\ \text{IF}_j=1-k_j\cdot\sin\beta_j \\ DI_{adj,j}(t)=DI_j(t)\cdot IF_j \\ \overline{DI}_{W,j}(t)=\dfrac{1}{W}\sum_{i=t-W+1}^{t}DI_{adj,j}(i) \\ \sigma_{DI,j}^{W}(t)=\sqrt{\dfrac{1}{W}\sum_{i=t-W+1}^{t}(DI_{adj,j}(i)-\overline{DI}_{W,j}(t))^2} \\ \begin{cases}\overline{DI}_{W,j}(t)>\theta_{1,j} \\ \sigma_{DI,j}^{W}(t)<\theta_{2,j}\end{cases} \end{cases} \tag{1.39}$$

5.2.4 模型求解

通过阶段一无积灰状态下的理论发电基准，结合阶段二积灰状态下的动态阈值与清洗决策优化两部分，我们得到积灰指标

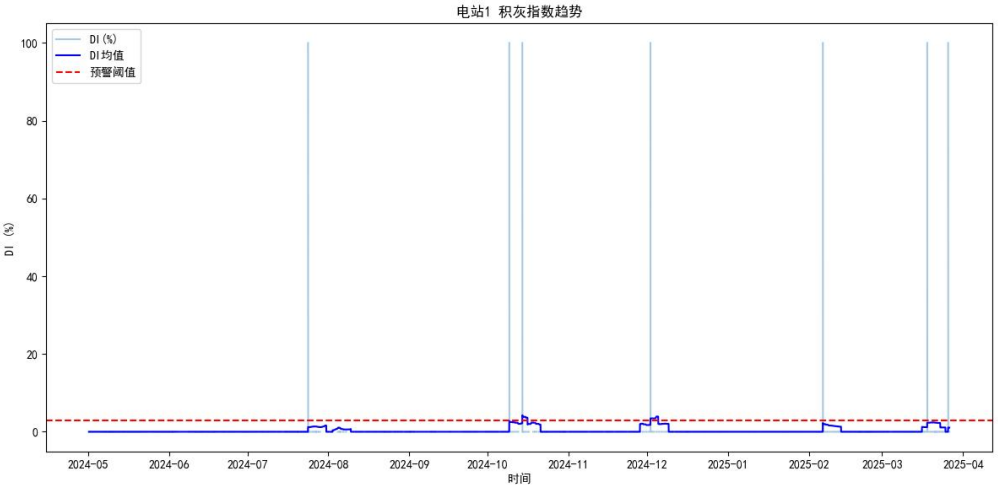

图 5.11 电站 1 积灰指数趋势图

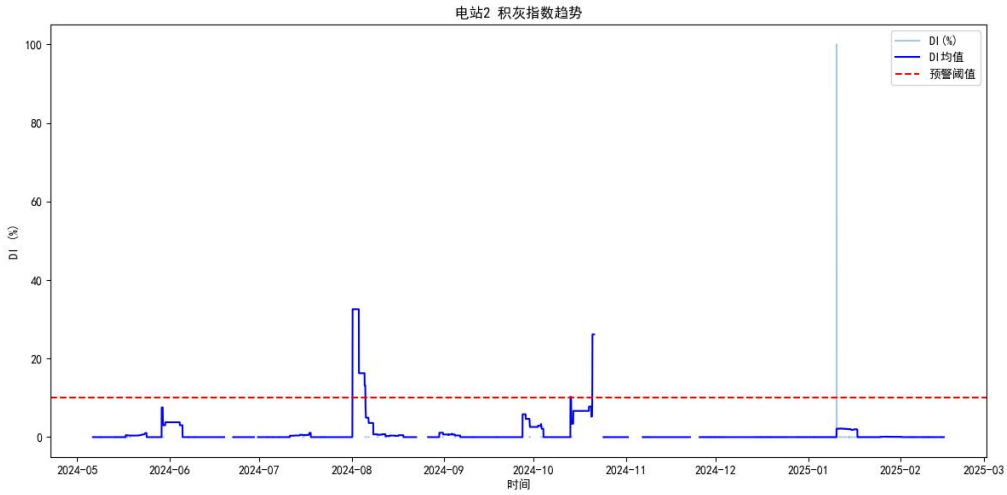

图 5.12 电站 2 积灰指数趋势图

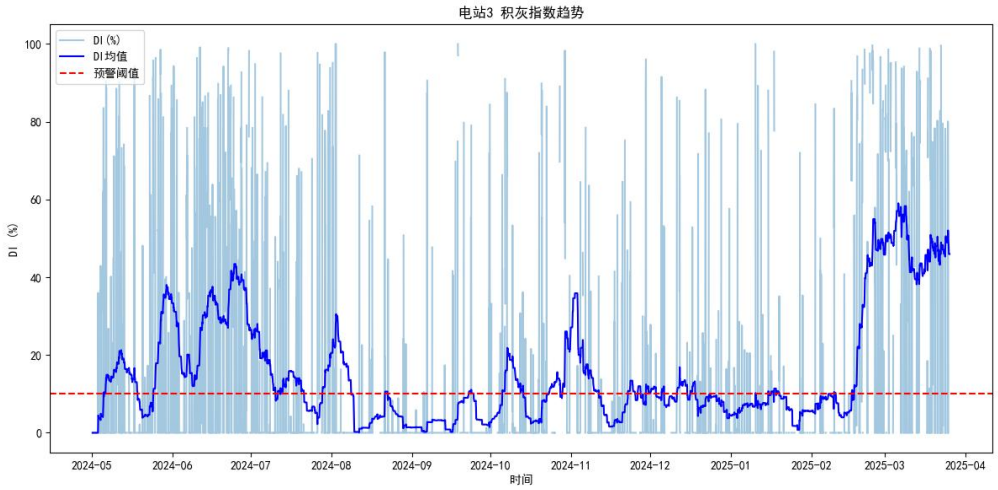

图 5.13 电站 3 积灰指数趋势图

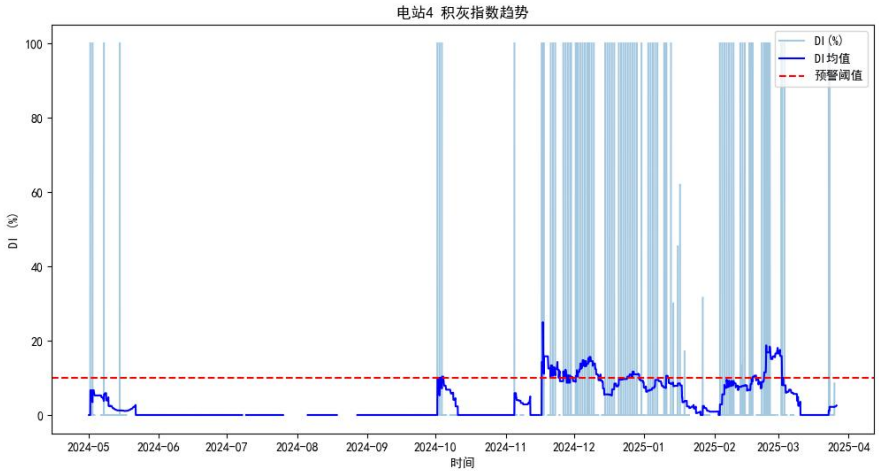

图 5.14 电站 4 积灰指数趋势图

5.3 问题三的模型建立和求解

5.3.1 确定清洗单价之下清洗节点的动态决策

（1） 模型建立

在清洗成本固定为 2 元/kW 时，动态决策的目标是：当积灰导致的发电损失≥清洗成本时，立即清洗，最小化总运营成本（发电损失成本加清洗成本），同时通过安全阈值避免极端积灰风险，本文构建一个贪心策略，当累计损失值达到了清洗成本，任一条件满足时均进行清洗。优先进行经济性检验[5]：累积损失超过清洗成本时立即清洗，确保每次清洗的净收益增益覆盖成本；后进行安全性兜底检验：当积灰指数（*DI*）均值超过安全阈值时强制清洗，防止不可逆效率损失。

首先是清洗成本定义：

$$C_{\text{clean}} = 2 \times C_{1-4} \tag{1.40}$$

其中，C_{1-4} 表示 1-4 电站的装机容量。发电损失成本可定义为：

$$L(t) = DI(t) \times P_{actual}(t) \times P_{electric}(P_{electric} = 0.582\text{元}/\text{度}) \tag{1.41}$$

累计损失：

$$S(t) = \sum_{i=t_{\text{last\_clean}}}^{t} L(i) \tag{1.42}$$

目标函数确定，最小化长期运营成本：

$$\min \lim_{T \to \infty} \frac{1}{T} \left(\sum_{t=1}^{T} L(t) + \sum_{k=1}^{N} C_{\text{clean}} \right) \tag{1.43}$$

动态决策规则：

$$\begin{cases} \text{经济性触发：} S(\text{t}) \geq \text{C}_{\text{clean}} \\ \text{安全性触发:} \overline{DI}_W(\text{t}) > \theta_{safe} \end{cases} \tag{1.44}$$

其中，经济性触发优先，安全性触发作为极端情况兜底。

（2） 模型分析与求解

基于问题二模型，在其基础上我们引入清洗成本[6]，对光伏板进行清洗，使得损失值降低，进而提升最大效益。清洗策略如下，电站 1 由于有 15 度的倾角，会导致不易积灰，故而清洗策略可能为 0，为正常现象。

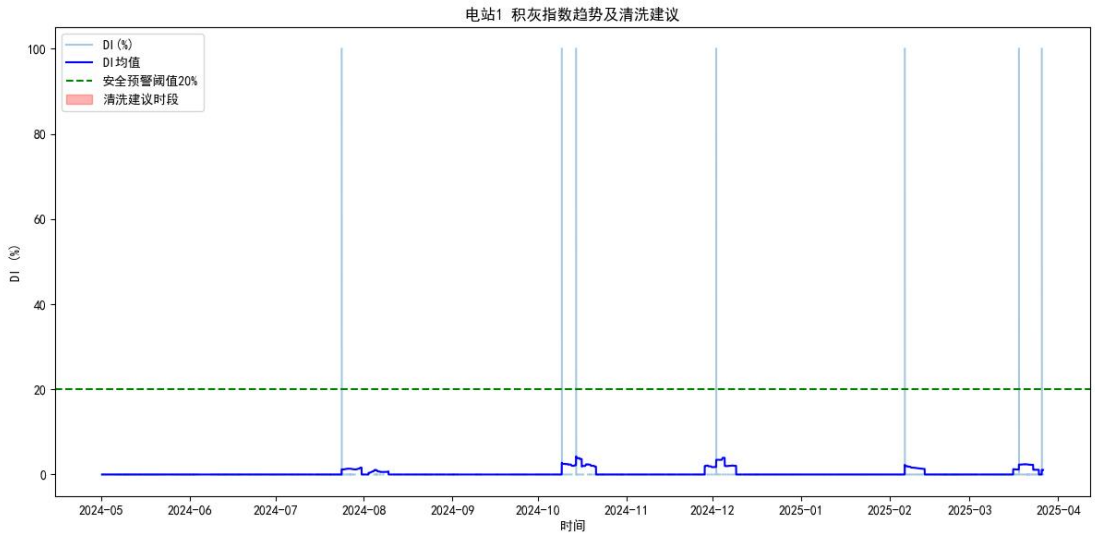

图 5.15 电站 1 积灰指数趋势及清洗建议图

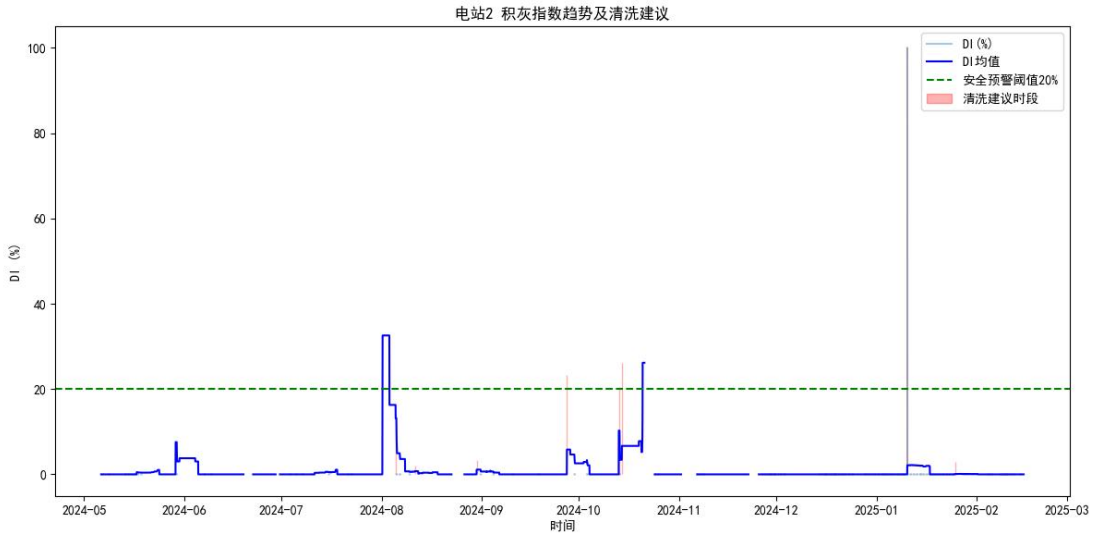

图 5.16 电站 2 积灰指数趋势及清洗建议图

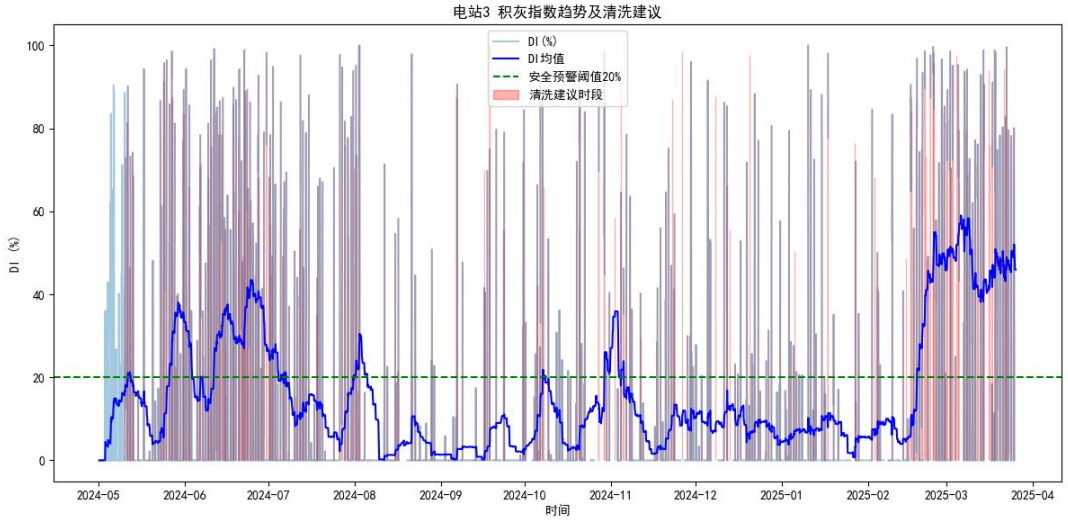

图 5.17 电站 3 积灰指数趋势及清洗建议图

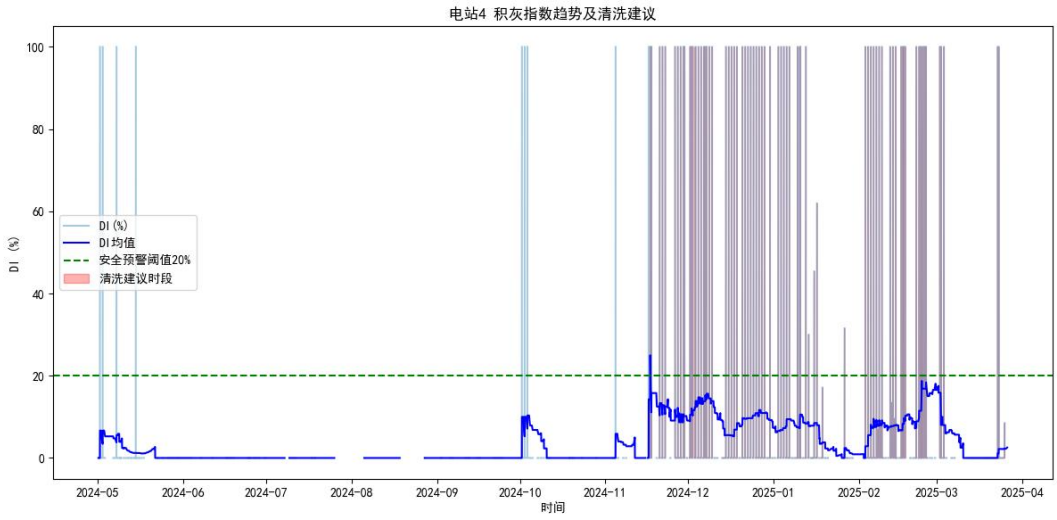

图 5.18 电站 4 积灰指数趋势及清洗建议图

5.3.2 清洗价格变动对于清洗决策影响

（1）模型建立

基于确定清洗单价之下清洗节点的动态决策的模型，本文重新构建清洗单价，分析清洗价格在区间[0.1,5.1] 元/kW 内变化时（步长 0.1 元/kW），各光伏电站的清洗次数变化规律，揭示价格敏感性与经济性阈值。

由于此时清洗策略会对后续产生影响，我们添加新的指标，在一次清洗之后的三天之内，电站不接受清洗，构建如下模型：当开始清洗时，触发时间变化。

$$\tau(t)=72 \tag{1.45}$$

触发之后会按照时间递减：

$$\tau(t+1)=\max(\tau(t)-1,0) \tag{1.46}$$

由于添加了冷却时间，此时对于约束条件应重新改变，其余部分仍基于上述模型。

$$\begin{cases} \text{经济性触发：} S(\mathrm{t}) \geq \mathrm{C}_{\mathrm{clean}} \\ \text{安全性触发：} \overline{DI}_W(\mathrm{t}) > \theta_{safe} \\ \text{冷却期结束：} \tau(t)=0 \end{cases} \tag{1.47}$$

（2）模型求解与分析

最终结果以不同价格下清洗次数不同折线图形式呈现，其中，由于电站 1 倾角 β=15°，显著降低积灰速率，经济性触发条件 $S(\mathrm{t}) \geq \mathrm{C}_{\mathrm{clean}}$ 始终未被满足，全年清洗次数 $\mathrm{N}_1(\mathrm{c})=0$。但为防止极端积灰，建议每年至少执行 1 次安全性兜底清洗。

而电站 3 虽倾角 0°，但清洗次数随价格上升呈现线性下降，如图 3。当清洗价格从 0.1 元/kW 增至 0.3 元/kW 时，全年清洗次数 $N_3(c)$ 从 36 次/年降至 32 次/年，降幅约 11%。

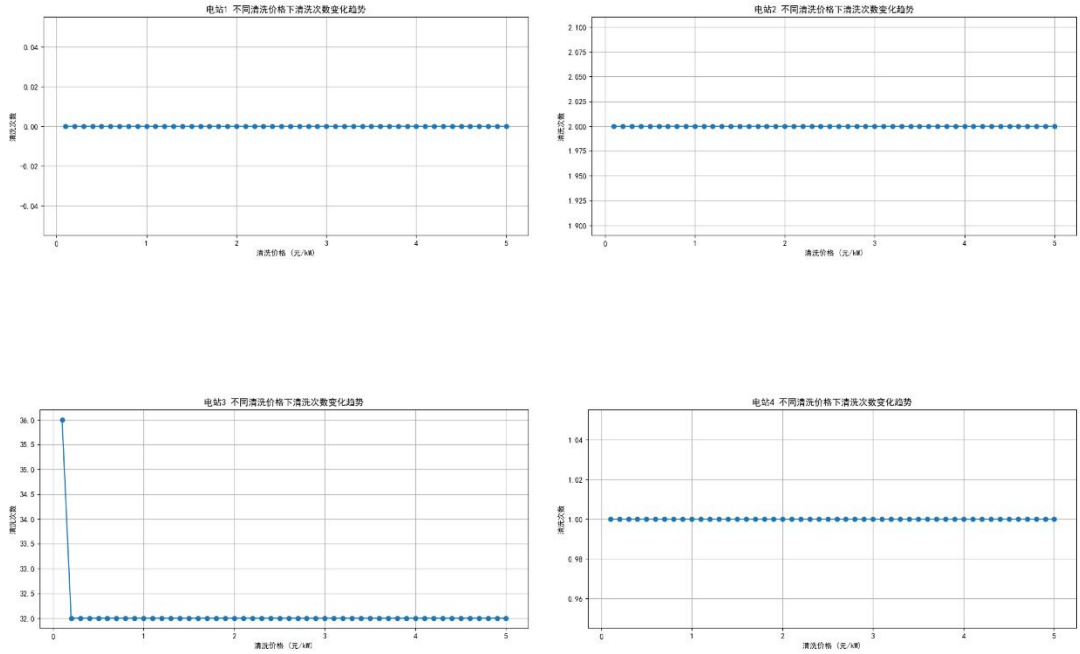

图 5.19 电站 1-4 积灰指数趋势及清洗价格图

六、模型的评价与改进

为了评估关键参数或输入变量的变化对模型输出的影响程度，从而验证模型的鲁棒性、可靠性及参数选择的合理性。针对光伏积灰检测与清洗预警模型，我们选取以下核心参数进行灵敏度分析：温度损失系数 α_1 和湿度损失系数 β 。

对于温度损失系数 α_1 的灵敏度分析：保持其它参数不变，令 α_1 在 $[0.3\%,0.5\%]/°C$ 内变化，由结果可知，理论功率 $\hat{P}(t)$ 随 α_1 的增大而线性减小。则 α_1 灵敏度分析通过；对于湿度损失系数 β 的灵敏度分析：保持其它参数不变，令 β 在一个小区间内变化，我们得到：湿度每增加 $x\%$，理论功率就会相应的损失 $x\% \times \beta$ 。则 β 灵敏度分析通过。

七、参考文献

[1] 李实;田春宁;鞠振河;殷志刚;.光伏系统的优化设计[J].沈阳工程学院学报(自然科学版),2011,v.7;No.26(02).

[2] 张锋凌;邹慧芳;顾冬;王汉蒙;薛原;饶琪;王钦永;张凤举;邹丽文;.塔式光热电站大规模定日镜上部结构快速安装技术[J].施工技术(中英文),2021,v.50;No.591(20).

[3] 李实;鞠振河;杜士鹏;殷志刚;康微微;.太阳能照明系统发电特性和优化选择[J].可再生能源,2009,v.27;No.145(03).

[4] 中国电建集团西北勘测设计研究院有限公司.一种塔式光热电站定日镜桩基础接地装置及其接地方法:CN110165434A[P].2019-08-23.

[5]于静,车俊铁,张吉月.太阳能发电技术综述[J].世界科技研究与发展, 2008, 30(1):4.DOI:10.3969/j.issn.1006-6055.2008.01.014.

[6]廖驰,钟杰,戴靠山,等.塔式光热电站定日镜结构风致响应与振动控制研究综述[J].土木与环境工程学报(中英文), 2023, 45(2):13.

附　录

附录1 天气和风向编码表

无

原始天气	编码
多云	1
浮尘	2
雷阵雨	3
霾	4
晴	5
雾	6
小雨	7
扬沙	8
阴	9
阵雨	10
中雨	11
大雨	12
暴雨	13
雨	14

原始风向	编码
北风	1
东北风	2
东风	3
东南风	4
南风	5
西北风	6
西风	7
西南风	8

附录2 发电数据清洗（python代码）

A1_1-4 发电数据缺失补充.py

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

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


# ======== 高斯函数 ========
def gaussian(x, A, mu, sigma):
    return A * np.exp(-((x - mu) ** 2) / (2 * sigma ** 2))

```

```python
# ======== 单文件修复 ========
def process_power_data(input_path: str, output_path: str):
    """
    input_path : 原始 Excel（含“时间、当日累计发电量 kwh”）
    output_path: 修复后 Excel 保存路径
    """
    # 1) 若目标文件夹不存在，自动创建
    os.makedirs(os.path.dirname(output_path), exist_ok=True)

    # 2) 读取 & 生成完整 5 min 索引
    raw_df = pd.read_excel(input_path, parse_dates=['时间'])
    df = raw_df.drop_duplicates('时间').set_index('时间').sort_index()
    full_index = pd.date_range(df.index.min().floor('5min'),
                               df.index.max().ceil('5min'),
                               freq='5min', name='时间')
    df = df.reindex(full_index)
    df['原始标记'] = df['当日累计发电量 kwh'].notna()

    # 3) 功率、累计差
    df['累计发电量 kwh'] = df['当日累计发电量 kwh'].ffill()
    df['功率'] = df['累计发电量 kwh'].diff().clip(lower=0).fillna(0)

    # 4) 按天处理
    all_dfs = []
    for day, day_df in df.groupby(df.index.date):
        day_df = day_df.copy()
        day_df['当日分钟
'] = (day_df.index - day_df.index.normalize()).total_seconds() / 60

        # —— 高斯拟合
        valid = (day_df['功率'] > 0) & day_df['原始标记']
        if valid.sum() > 10:
            x, y = day_df.loc[valid, ['当日分钟', '功率']].values.T
            try:
                dur = x.max() - x.min()
                popt, _ = curve_fit(gaussian, x, y, p0=[y.max(), x.mea
n(), dur / 2], maxfev=5000)
                day_df['拟合功率'] = gaussian(day_df['当日分钟
'], *popt).clip(lower=0)
            except Exception:
                day_df['拟合功率'] = day_df['功率
'].rolling(6, min_periods=1).mean()
        else:
            day_df['拟合功率'] = day_df['功率']

        # —— 修复规则
        day_df['累计差'] = day_df['累计发电量 kwh'].diff().fillna(0)
        day_df['修复功率'] = day_df['功率']

```

```
        # 1) 功率=0 & 累计差=0
        flat = (day_df['功率'] == 0) & (day_df['累计差'] == 0)
        missing = flat & (~day_df['原始标记'])          # 真缺失
        keep0   = flat & day_df['原始标记']             # 夜间/停机
        day_df.loc[missing, '修复功率'] = day_df.loc[missing, '拟合功率']
        day_df.loc[keep0,   '修复功率'] = 0

        # 2) 22 点-次日 2 点 强制 0
        night = (day_df.index.hour >= 22) | (day_df.index.hour < 2)
        day_df.loc[night, '修复功率'] = 0

        # 3) 0 → 大跳增
        jump_thr = 400
        prev = day_df['功率'].shift(1).fillna(0)
        jump_mask = (prev == 0) & (day_df['功率'] > jump_thr)
        day_df.loc[jump_mask, '修复功率'] = day_df.loc[jump_mask, '拟合功率']

        # 4) 累计突增前段回填
        big = day_df['累计差'] > 300
        for ts in day_df.index[big]:
            for i in range(1, 13):
                idx = ts - pd.Timedelta(minutes=5 * i)
                if idx not in day_df.index:
                    break
                if day_df.at[idx, '功率'] > 0 or day_df.at[idx, '累计差'] != 0:
                    break
                if day_df.at[idx, '修复功率'] == 0:
                    day_df.at[idx, '修复功率'] = day_df.at[idx, '拟合功率']

        all_dfs.append(day_df)

    merged = pd.concat(all_dfs)
    merged['修复累计发电量'] = merged['修复功率'].fillna(0).cumsum() + merged['累计发电量 kwh'].iloc[0]

    merged[['累计发电量 kwh', '功率', '拟合功率', '修复功率', '修复累计发电量']].to_excel(output_path)
    print(f"✅ {os.path.basename(input_path)} 处理完成 → {output_path}")


# ======== 批量调用 ========
if __name__ == '__main__':
    # 待处理的电站 Excel 文件名
    stations = ["电站 1 发电数据.xlsx",
```

```
                "电站 2 发电数据.xlsx",
                "电站 3 发电数据.xlsx",
                "电站 4 发电数据.xlsx"]

    # 输出目录
    out_dir = r"A1_发电数据_缺失值补充"
    os.makedirs(out_dir, exist_ok=True)

    for fname in stations:
        in_path  = fname
        base     = os.path.splitext(fname)[0] + "_修复结果.xlsx"
        out_path = os.path.join(out_dir, base)
        process_power_data(in_path, out_path)
```

附录3 发电数据小时化小处理（python代码）

A1_1-4 小时处理.py

```
import pandas as pd

# 要处理的四个修复结果文件名
input_files = [
    "电站 1 发电数据_修复结果.xlsx",
    "电站 2 发电数据_修复结果.xlsx",
    "电站 3 发电数据_修复结果.xlsx",
    "电站 4 发电数据_修复结果.xlsx"
]

for file in input_files:
    try:
        # 读取数据
        df = pd.read_excel(file, parse_dates=['时间'])
        df = df.sort_values('时间').reset_index(drop=True)

        # 计算每日归零的修复累计发电量
        df['修复累计发电量（每日归零）'] = 0.0
        df['日期'] = df['时间'].dt.date
        for date, group in df.groupby('日期'):
            cumulative = group['修复功率'].cumsum()
            df.loc[group.index, '修复累计发电量（每日归零）'] = cumulative
        df.drop(columns=['日期'], inplace=True)

        # 按小时汇总
        hourly_df = df.resample('1h', on='时间').agg({
```

```
            '功率': 'sum',
            '拟合功率': 'sum',
            '修复功率': 'sum',
            '累计发电量 kwh': 'last',
            '修复累计发电量': 'last',
            '修复累计发电量（每日归零）': 'last'
        }).reset_index()

        # 输出文件名（不覆盖）
        new_filename = file.replace("修复结果.xlsx", "每小时汇总.xlsx")
        hourly_df.to_excel(new_filename, index=False)
        print(f"✅ 已保存：{new_filename}")

    except Exception as e:
        print(f"❌ 处理失败：{file}，错误：{e}")
```

附录4 环境检测仪数据缺失补充（python代码）

A1_1-4 天气小时处理.py

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

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

# 高斯模型
def gaussian(x, A, mu, sigma):
    return A * np.exp(- ((x - mu) ** 2) / (2 * sigma ** 2))

def repair_radiation_data(input_file, output_file):
    df = pd.read_excel(input_file)
    df.columns = ['时间', '辐照强度']
    df['时间'] = pd.to_datetime(df['时间']).dt.floor('min')
    df = df.drop_duplicates('时间').set_index('时间').sort_index()

    # 重建完整时间索引（按分钟）
    full_index = pd.date_range(df.index.min(), df.index.max(), freq='1min')
    df = df.reindex(full_index)
    df.index.name = '时间'
    df['原始'] = df['辐照强度'].notna()
```

```
24.
25.     days = pd.unique(df.index.date)
26.     all_dfs = []
27.
28.     for day in days:
29.         day_df = df.loc[str(day)].copy()
30.         day_df['分钟
    '] = (day_df.index - day_df.index.normalize()).total_seconds() / 60
31.
32.         valid_data = day_df.loc[day_df['辐照强度'].notna(), ['分钟', '
    辐照强度']]
33.         if len(valid_data) >= 10:
34.             try:
35.                 x = valid_data['分钟'].values
36.                 y = valid_data['辐照强度'].values
37.                 p0 = [y.max(), x[np.argmax(y)], (x.max() - x.min()) /
    4]
38.                 popt, _ = curve_fit(gaussian, x, y, p0=p0, maxfev=8000
    )
39.                 day_df['拟合值'] = gaussian(day_df['分钟
    '], *popt).clip(lower=0)
40.             except Exception as e:
41.                 print(f"拟合失败：{day}，使用滑动平均。错误：{e}")
42.                 day_df['拟合值'] = valid_data['辐照强度
    '].rolling(6, min_periods=1).mean().reindex(day_df.index, method='near
    est').fillna(0)
43.         else:
44.             day_df['拟合值'] = valid_data['辐照强度
    '].rolling(6, min_periods=1).mean().reindex(day_df.index, method='near
    est').fillna(0)
45.
46.         day_df['修复强度'] = day_df['辐照强度']
47.         day_df.loc[day_df['辐照强度'].isna(), '修复强度
    '] = day_df.loc[day_df['辐照强度'].isna(), '拟合值']
48.
49.         all_dfs.append(day_df)
50.
51.
52.     final_df = pd.concat(all_dfs)
53.     final_df[['辐照强度', '拟合值', '修复强度', '原始
    ']].to_excel(output_file)
54.
55.     # 可视化最近一天
56.     last_day = final_df.index.max().normalize()
57.     plot_df = final_df.loc[last_day:last_day + pd.Timedelta(days=1)]
58.     plt.figure(figsize=(15, 5))
59.     plt.plot(plot_df.index, plot_df['辐照强度'], label='原始值
    ', alpha=0.6)
60.     plt.plot(plot_df.index, plot_df['修复强度'], '--', label='修复后')
61.     plt.title("辐照强度拟合修复示例（最近一天）")
```

```
    plt.xlabel("时间")
    plt.ylabel("W/m²")
    plt.legend()
    plt.grid()
    plt.tight_layout()
    plt.savefig("辐照强度_修复示意图.png")
    plt.close()

    print("✅ 修复完成，结果已保存：", output_file)

# 执行示例
repair_radiation_data("电站 1 环境检测仪数据.xlsx", "电站 1_辐照强度_修复结果.xlsx")
```

附录5 环境检测仪数据小时化处理（python代码）

A1_1.py

```
import pandas as pd
import numpy as np
from scipy import stats
from openpyxl.styles import PatternFill
import matplotlib.pyplot as plt

# 设置中文字体支持
plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False


# 夜间时段定义
NIGHT_HOURS = (0, 5)  # 0:00-5:00 视为夜间
NIGHT_FILL = PatternFill(start_color='FFF2CC', end_color='FFF2CC', fill_type='solid')

def load_and_clean_data(power_file, irrad_file):
    """
    加载并清洗发电和辐照数据
    :return: 清洗后的发电数据、辐照数据
    """
    # ========== 1. 发电数据处理 ==========
    df_power = pd.read_excel(
        power_file,
        parse_dates=['时间'],
```

```
        usecols=['时间', '当日累计发电量 kwh']
    ).rename(columns={'当日累计发电量 kwh': '累计发电量'})

    df_power = df_power.sort_values('时间').set_index('时间')
    hourly_power = df_power.resample('1h').last().ffill()
    hourly_power['小时发电量
'] = hourly_power.groupby(hourly_power.index.date)['累计发电量
'].diff().fillna(0)
    hourly_power['小时发电量'] = hourly_power['小时发电量
'].clip(lower=0)

    # ========== 2. 辐照数据处理 ==========
    df_irrad = pd.read_excel(
        irrad_file,
        parse_dates=['时间'],
        usecols=['时间', '辐照强度 w/m2']
    ).rename(columns={'辐照强度 w/m2': '辐照强度'})

    df_irrad = df_irrad.set_index('时间').resample('1h').mean()

    # ========== 3. 高级清洗（夜间修正数据）==========
    # 处理发电数据夜间值
    night_mask_power = (hourly_power.index.hour >= NIGHT_HOURS[0]) & (
hourly_power.index.hour < NIGHT_HOURS[1])
    hourly_power.loc[night_mask_power, '小时发电量'] = 0

    # 处理辐照数据夜间值
    night_mask_irrad = (df_irrad.index.hour >= NIGHT_HOURS[0]) & (df_i
rrad.index.hour < NIGHT_HOURS[1])
    df_irrad.loc[night_mask_irrad, '辐照强度'] = 0

    return hourly_power, df_irrad

def export_to_excel(df_power, df_irrad, output_power_path, output_irra
d_path):
    """数据导出到 Excel"""
    # 发电数据导出
    if not df_power.empty:
        df_power.to_excel(output_power_path, index=True)
        print(f"发电数据已保存到 {output_power_path}")
    else:
        print("警告：发电数据为空，未生成文件")

    # 辐照数据导出
    if not df_irrad.empty:
        df_irrad.to_excel(output_irrad_path, index=True)
        print(f"辐照数据已保存到 {output_irrad_path}")
    else:
        print("警告：辐照数据为空，未生成文件")
```

```
68.
69.if __name__ == "__main__":
70.    input_files = {
71.        'power_file': '电站 4 发电数据.xlsx',
72.        'irrad_file': '电站 4 环境监测仪数据.xlsx'
73.    }
74.
75.    try:
76.        # 清洗数据
77.        df_power, df_irrad = load_and_clean_data(
78.            input_files['power_file'],
79.            input_files['irrad_file']
80.        )
81.
82.        # 导出数据
83.        export_to_excel(
84.            df_power,
85.            df_irrad,
86.            '清洗后的发电数据.xlsx',
87.            r'A1_辐射数据_缺失补充\电站 4 环境检测仪数据_每小时汇总.xlsx'
88.        )
89.
90.        # 生成趋势图
91.        plt.figure(figsize=(15, 5))
92.        plt.plot(df_power.index, df_power['小时发电量'], label='发电量')
93.        plt.plot(df_irrad.index, df_irrad['辐照强度'], label='辐照强度
   ', alpha=0.7)
94.        plt.title('发电量与辐照强度趋势')
95.        plt.legend()
96.        plt.savefig('电站 4 趋势对比图.png')
97.        plt.show()
98.
99.    except Exception as e:
100.        print(f"处理失败: {str(e)}")
```

附录6 天气重复值处理（python代码）

A1_天气_L.py

```
1. import pandas as pd
2. import numpy as np
3. import os
4.
5.
```

```
def process_group(group):
    """处理时间重复的分组数据（增强版）"""
    if len(group) == 1:
        return group.iloc[0]

    # 列分类定义
    mean_cols = ["最高温度", "最低温度", "风速", "湿度"]
    mode_cols = ["当前温度", "天气", "风向"]
    sun_cols = ["日出时间", "日落时间"]

    # ====== 均值列处理 ======
    try:
        # 计算初始均值（至少需要1个有效值）
        initial_mean = group[mean_cols].mean(skipna=True)

        # 填充缺失值并计算距离
        filled_data = group[mean_cols].fillna(initial_mean)
        distances = (filled_data - initial_mean).abs().sum(axis=1)

        # 删除距离最大的行（至少保留1条）
        filtered_group = group.drop(distances.idxmax())

        # 计算最终均值（允许全空值）
        final_mean = filtered_group[mean_cols].mean(skipna=True)
    except Exception as e:
        print(f"处理分组时发生错误（时间：{group.name}）：{str(e)}")
        final_mean = pd.Series([np.nan] * len(mean_cols), index=mean_cols)

    # ====== 众数列处理 ======
    mode_values = {}
    for col in mode_cols:
        try:
            # 使用原始数据计算众数（非处理后的数据）
            non_null = group[col].dropna()
            if len(non_null) > 0:
                mode_val = non_null.mode()
                mode_values[col] = mode_val[0] if not mode_val.empty else np.nan
            else:
                mode_values[col] = np.nan
        except:
            mode_values[col] = np.nan

    # ====== 日出日落时间处理 ======
    sun_values = {}
    for col in sun_cols:
        try:
            # 优先使用处理后的数据，其次用原始数据
```

```
            non_null = filtered_group[col].dropna()
            if non_null.empty:
                non_null = group[col].dropna()
            sun_values[col] = non_null.iloc[0] if not non_null.empty else np.nan
        except:
            sun_values[col] = np.nan

    return pd.Series({
        **final_mean.to_dict(),
        **mode_values,
        **sun_values,
        "时间": group.name
    })


# ====== 主处理流程 ======
if __name__ == "__main__":
    # 读取原始数据
    input_path = "电站4天气数据.xlsx"
    origin_df = pd.read_excel(input_path)

    # 时间格式处理
    origin_df["时间"] = pd.to_datetime(origin_df["时间"], errors="coerce")

    # 步骤1：数据拆分
    duplicate_mask = origin_df.duplicated("时间", keep=False)
    duplicate_df = origin_df[duplicate_mask].sort_values("时间")
    unique_df = origin_df[~duplicate_mask]

    # 步骤2：处理重复数据
    processed_data = []
    if not duplicate_df.empty:
        for time, group in duplicate_df.groupby("时间"):
            try:
                processed_row = process_group(group)
                processed_data.append(processed_row)
            except Exception as e:
                print(f"处理时间 {time} 时发生严重错误: {str(e)}")
                continue

    # 创建结果DataFrame
    processed_df = pd.DataFrame(processed_data, columns=origin_df.columns)

    # 步骤3：合并数据
    final_df = pd.concat([unique_df, processed_df], axis=0)

```

```
    # 最终排序去重（处理后的数据优先）
    final_df = final_df.sort_values("时间", ascending=True)
    final_df = final_df.drop_duplicates("时间", keep="first")

    # 创建输出目录
    output_dir = "A1_天气数据_重复值整合"
    os.makedirs(output_dir, exist_ok=True)

    # 保存结果（保留原始格式）
    output_path = os.path.join(output_dir, "电站 4 天气数据整合.xlsx")
    final_df.to_excel(output_path, index=False, engine="openpyxl")

    # 打印验证信息
    print("\n" + "=" * 40)
    print(f"原始数据记录数: {len(origin_df)}")
    print(f"发现重复时间数: {len(duplicate_df.groupby('时间'))}")
    print(f"最终有效记录数: {len(final_df)}")
    print(f"已清除重复记录: {len(origin_df) - len(final_df)}")
    print(f"结果文件已保存至: {os.path.abspath(output_path)}")
    print("=" * 40)
```

附录7 天气小时化处理（python代码）

A1_1-4 天气小时处理.py

```
import pandas as pd
import numpy as np

def process_weather_hourly(file_path):
    # 读取数据
    df = pd.read_excel(file_path, parse_dates=['时间'])

    # 保留所需列
    df = df[['时间', '当前温度', '最高温度', '最低温度', '天气', '风向', '风速', '湿度', '日出时间', '日落时间']]

    # 提取日期
    df['日期'] = df['时间'].dt.date

    # 提取每个日期中第一次出现的日出/日落时间
    sunrise_sunset = df.groupby('日期')[['日出时间', '日落时间']].first().reset_index()

```

```
    # 时间四舍五入到整点（分钟 >=30 进 1 小时）
    df['时间'] = df['时间'].dt.round('h')

    # 去重：防止同一小时有多条记录
    df = df.drop_duplicates('时间').set_index('时间')

    # 构造完整小时时间索引
    time_range = pd.date_range(start=df.index.min(), end=df.index.max(), freq='1h')
    df_full = df.reindex(time_range)

    # 数值列插值
    numeric_cols = ['当前温度', '最高温度', '最低温度', '风速', '湿度']
    df_full[numeric_cols] = df_full[numeric_cols].interpolate(method='linear')

    # 类别列用前向填充
    df_full['天气'] = df_full['天气'].ffill()
    df_full['风向'] = df_full['风向'].ffill()

    # 重置索引
    df_full.reset_index(inplace=True)
    df_full.rename(columns={'index': '时间'}, inplace=True)

    # 添加日期列用于合并
    df_full['日期'] = df_full['时间'].dt.date

    # 合并日出/日落时间
    df_full = df_full.merge(sunrise_sunset, on='日期', how='left')

    # 删除辅助列
    df_full.drop(columns=['日期'], inplace=True)

    return df_full

# 使用示例
result = process_weather_hourly('电站 4 天气数据整合.xlsx')
result.to_excel('电站 4 天气数据整合_每小时汇总.xlsx', index=False)

print("整理完成，已保存为 '电站 4 天气数据整合_每小时汇总.xlsx'")
```

附录8 问题二求解（python代码）

A2.py

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

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

# 全局配置
stations = {
    '电站1': {'capacity': 4998.30, 'tilt': 15, 'clean_date': '2024-12-05'},
    '电站2': {'capacity': 5581.00, 'tilt': 0, 'clean_date': '2025-01-02'},
    '电站3': {'capacity': 4456.00, 'tilt': 0, 'clean_date': '2025-01-02'},
    '电站4': {'capacity': 1794.61, 'tilt': 0, 'clean_date': '2025-01-02'}
}


def compute_theoretical_power(row, eta, alpha1, beta, tref, h0, h1, ifactor):
    G = row['辐照强度']
    T_env = row['环境温度']
    H = row['湿度']
    W = row['风速']
    if G <= 0:
        return 0
    T_back = T_env + G / (h0 + h1 * W)
    P_pred = eta * G * (1 - alpha1 * (T_back - tref)) * (1 - beta * H) * ifactor
    return max(P_pred, 0)


def sliding_statistics(series, window=168):
    return series.rolling(window=window, min_periods=1).mean(), series.rolling(window=window, min_periods=1).std()


def process_station(station_name, config):
    print(f"\n>>> 正在处理：{station_name}")

    # 文件路径
    power_file = f"A2/A2_发电数据/{station_name}发电数据_每小时汇总.xlsx"
    env_file = f"A2/A2_辐射数据/{station_name}环境检测仪数据_每小时汇总.xlsx"
    weather_file = f"A2/A2_天气数据/{station_name}天气数据整合_每小时汇
```

```
    总.xlsx"
43.
44.     # 读取数据
45.     power_df = pd.read_excel(power_file, parse_dates=['时间'])[['时间
    ', '修复累计发电量（每日归零）']].rename(
46.         columns={'修复累计发电量（每日归零）': '累计发电量'})
47.     env_df = pd.read_excel(env_file, parse_dates=['时间'])[['时间', '辐
    照强度']]
48.     weather_df = pd.read_excel(weather_file, parse_dates=['时间'])[['时
    间', '当前温度', '风速', '湿度']].rename(
49.         columns={'当前温度': '环境温度'})
50.
51.     # 合并数据
52.     df = power_df.merge(env_df, on='时间').merge(weather_df, on='时间')
53.     df = df.set_index('时间').sort_index()
54.     df['实际功率'] = df['累计发电量'].diff().fillna(0).clip(lower=0)
55.
56.     # 参数
57.     cap = config['capacity']
58.     beta = config['tilt']
59.     area = cap / 0.18 / 1000
60.     eta = cap / area / 1000
61.     tref = 25
62.     alpha1 = 0.004
63.     beta_h = 0.02
64.     k = 0.2 if beta > 0 else 0
65.     ifactor = 1 - k * np.sin(np.radians(beta))
66.     clean_date = pd.to_datetime(config['clean_date'])
67.
68.     # 拟合模型
69.     train_df = df[(df.index.date >= clean_date.date()) & (df.index.dat
    e <= (clean_date + timedelta(days=2)).date())]
70.     valid = train_df[(train_df['辐照强度'] > 200) & (train_df['实际功率
    '] > 0)]
71.
72.     def model_func(xdata, h0, h1):
73.         G, T_env, H, W = xdata
74.         T_back = T_env + G / (h0 + h1 * W)
75.         return eta * G * (1 - alpha1 * (T_back - tref)) * (1 - beta_h
    * H) * ifactor
76.
77.     xdata = np.array([valid['辐照强度'], valid['环境温度'], valid['湿度
    '], valid['风速']])
78.     ydata = valid['实际功率'].values
79.     popt, _ = curve_fit(model_func, xdata, ydata, p0=[20, 5], maxfev=1
    0000)
80.     h0_fit, h1_fit = popt
81.
82.     # 理论功率与DI
```

```
    df['理论功率'] = df.apply(
        lambda row: compute_theoretical_power(row, eta, alpha1, beta_h, tref, h0_fit, h1_fit, ifactor), axis=1)
    df['DI'] = (df['理论功率'] - df['实际功率']) / df['理论功率'].replace(0, np.nan) * 100
    df['DI'] = df['DI'].clip(lower=0)
    df['DI 均值'], df['DI 波动'] = sliding_statistics(df['DI'])

    # 预警判断
    theta1, theta2 = (3, 4) if station_name == '电站 1' else (10, 5)
    df['预警'] = (df['DI 均值'] > theta1) & (df['DI 波动'] < theta2)

    # 保存结果
    df.reset_index().to_excel(f"A2/ANS/{station_name}_DI 分析结果.xlsx", index=False)

    # 绘图
    plt.figure(figsize=(12, 6))
    plt.plot(df.index, df['DI'], label='DI(%)', alpha=0.4)
    plt.plot(df.index, df['DI 均值'], label='DI 均值', color='blue')
     plt.axhline(theta1, color='red', linestyle='--', label='预警阈值')
     plt.title(f"{station_name} 积灰指数趋势")
     plt.xlabel("时间")
     plt.ylabel("DI (%)")
     plt.legend()
     #plt.tight_layout()
     if station_name == '电站 4':
         plt.legend(loc='upper right')
     else:
         plt.tight_layout()

     plt.savefig(f"A2/ANS/{station_name}_DI 趋势图.png")
     plt.show()
     plt.close()
     print(f">>> 处理完成: {station_name}")


# 主程序
for name, config in stations.items():
    process_station(name, config)
```

附录9 问题三第一小问求解（python代码）

A3_1 清洗节点决策.py

```python

import os
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit
from datetime import timedelta

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

# 全局配置
stations = {
    '电站1': {'capacity': 4998.30, 'tilt': 15, 'clean_date': '2024-12-05'},
    '电站2': {'capacity': 5581.00, 'tilt': 0, 'clean_date': '2025-01-02'},
    '电站3': {'capacity': 4456.00, 'tilt': 0, 'clean_date': '2025-01-02'},
    '电站4': {'capacity': 1794.61, 'tilt': 0, 'clean_date': '2025-01-02'}
}

def compute_theoretical_power(row, eta, alpha1, beta, tref, h0, h1, ifactor):
    G = row['辐照强度']
    T_env = row['环境温度']
    H = row['湿度']
    W = row['风速']
    if G <= 0:
        return 0
    T_back = T_env + G / (h0 + h1 * W)
    P_pred = eta * G * (1 - alpha1 * (T_back - tref)) * (1 - beta * H) * ifactor
    return max(P_pred, 0)

def sliding_statistics(series, window=168):
    return series.rolling(window=window, min_periods=1).mean(), series.rolling(window=window, min_periods=1).std()

def dynamic_cleaning_decision(df, cap, p_electric=0.582, theta_safe=20):
    C_clean = 2 * cap
    df = df.copy()
    df['发电损失'] = df['DI'] / 100 * df['实际功率'] * p_electric
    df['累积损失'] = df['发电损失'].cumsum()

```

```
    df['清洗建议'] = False
    if (df['累积损失'] >= C_clean).any():
        t_clean = df[df['累积损失'] >= C_clean].index[0]
        df.loc[t_clean:, '清洗建议'] = True
    elif (df['DI 均值'] > theta_safe).any():
        t_clean = df[df['DI 均值'] > theta_safe].index[0]
        df.loc[t_clean:, '清洗建议'] = True
    return df


def process_station(station_name, config):
    print(f"\n>>> 正在处理: {station_name}")

    # 文件路径
    power_file = f"A2/A2_发电数据/{station_name}发电数据_每小时汇总.xlsx"
    env_file = f"A2/A2_辐射数据/{station_name}环境检测仪数据_每小时汇总.xlsx"
    weather_file = f"A2/A2_天气数据/{station_name}天气数据整合_每小时汇总.xlsx"

    # 读取数据
    power_df = pd.read_excel(power_file, parse_dates=['时间'])[['时间', '修复累计发电量（每日归零）']].rename(
        columns={'修复累计发电量（每日归零）': '累计发电量'})
    env_df = pd.read_excel(env_file, parse_dates=['时间'])[['时间', '辐照强度']]
    weather_df = pd.read_excel(weather_file, parse_dates=['时间'])[['时间', '当前温度', '风速', '湿度']].rename(
        columns={'当前温度': '环境温度'})

    # 合并数据
    df = power_df.merge(env_df, on='时间').merge(weather_df, on='时间')
    df = df.set_index('时间').sort_index()
    df['实际功率'] = df['累计发电量'].diff().fillna(0).clip(lower=0)

    # 参数
    cap = config['capacity']
    beta = config['tilt']
    area = cap / 0.18 / 1000
    eta = cap / area / 1000
    tref = 25
    alpha1 = 0.004
    beta_h = 0.02
    k = 0.2 if beta > 0 else 0
    ifactor = 1 - k * np.sin(np.radians(beta))
    clean_date = pd.to_datetime(config['clean_date'])

    # 拟合模型
    train_df = df[(df.index.date >= clean_date.date()) & (df.index.dat
```

```
    e <= (clean_date + timedelta(days=2)).date())]
    valid = train_df[(train_df['辐照强度'] > 200) & (train_df['实际功率'] > 0)]

    def model_func(xdata, h0, h1):
        G, T_env, H, W = xdata
        T_back = T_env + G / (h0 + h1 * W)
        return eta * G * (1 - alpha1 * (T_back - tref)) * (1 - beta_h * H) * ifactor

    xdata = np.array([valid['辐照强度'], valid['环境温度'], valid['湿度'], valid['风速']])
    ydata = valid['实际功率'].values
    popt, _ = curve_fit(model_func, xdata, ydata, p0=[20, 5], maxfev=10000)
    h0_fit, h1_fit = popt

    # 理论功率与DI
    df['理论功率'] = df.apply(lambda row: compute_theoretical_power(row, eta, alpha1, beta_h, tref, h0_fit, h1_fit, ifactor), axis=1)
    df['DI'] = (df['理论功率'] - df['实际功率']) / df['理论功率'].replace(0, np.nan) * 100
    df['DI'] = df['DI'].clip(lower=0)
    df['DI 均值'], df['DI 波动'] = sliding_statistics(df['DI'])

    # 添加清洗时间决策建议
    df = dynamic_cleaning_decision(df, cap)

    # 输出结果
    os.makedirs("A3/ANS_1", exist_ok=True)
    df.reset_index().to_excel(f"A3/ANS_1/{station_name}_DI 分析结果_含清洗建议.xlsx", index=False)

    # 绘图
    plt.figure(figsize=(12, 6))
    plt.plot(df.index, df['DI'], label='DI(%)', alpha=0.4)
    plt.plot(df.index, df['DI 均值'], label='DI 均值', color='blue')
    plt.axhline(20, color='green', linestyle='--', label='安全预警阈值 20%')
    plt.fill_between(df.index, 0, df['DI'], where=df['清洗建议'], color='red', alpha=0.3, label='清洗建议时段')
    plt.title(f"{station_name} 积灰指数趋势及清洗建议")
    plt.xlabel("时间")
    plt.ylabel("DI (%)")
    plt.legend()
    plt.tight_layout()
    plt.savefig(f"A3/ANS_1/{station_name}_清洗决策趋势图.png")
    plt.close()
    print(f">>> 处理完成：{station_name}")
```

```
122.
123. # 主程序
124. for name, config in stations.items():
125.     process_station(name, config)
```

附录10 问题三第二小问求解（python代码）

A3_清洗价格对清洗决策影响.py

```
1. import os
2. import pandas as pd
3. import numpy as np
4. import matplotlib.pyplot as plt
5. from scipy.optimize import curve_fit
6. from datetime import timedelta
7.
8. plt.rcParams['font.sans-serif'] = ['SimHei']
9. plt.rcParams['axes.unicode_minus'] = False
10.
11. # 全局配置
12. stations = {
13.     '电站1': {'capacity': 4998.30, 'tilt': 15, 'clean_date': '2024-12-05'},
14.     '电站2': {'capacity': 5581.00, 'tilt': 0, 'clean_date': '2025-01-02'},
15.     '电站3': {'capacity': 4456.00, 'tilt': 0, 'clean_date': '2025-01-02'},
16.     '电站4': {'capacity': 1794.61, 'tilt': 0, 'clean_date': '2025-01-02'}
17. }
18.
19. def compute_theoretical_power(row, eta, alpha1, beta, tref, h0, h1, ifactor):
20.     G = row['辐照强度']
21.     T_env = row['环境温度']
22.     H = row['湿度']
23.     W = row['风速']
24.     if G <= 0:
25.         return 0
26.     T_back = T_env + G / (h0 + h1 * W)
27.     P_pred = eta * G * (1 - alpha1 * (T_back - tref)) * (1 - beta * H) * ifactor
28.     return max(P_pred, 0)
29.
```

```
def sliding_statistics(series, window=168):
    return series.rolling(window=window, min_periods=1).mean(), series.rolling(window=window, min_periods=1).std()

def dynamic_cleaning_count(df, cap, price_list, p_electric=0.582, theta_safe=20, cooldown_hours=72):
    result = {}
    for price in price_list:
        C_clean = price * cap
        #print(C_clean)
        cleaned_times = 0
        acc_loss = 0
        last_clean_time = None
        df_copy = df.copy()
        df_copy['发电损失'] = df_copy['DI'] / 100 * df_copy['实际功率'] * p_electric
        df_copy['清洗建议'] = False

        for t in df_copy.index:
            # 冷却期判断 3 天内不再清洗
            if last_clean_time is not None and (t - last_clean_time).total_seconds() < cooldown_hours * 3600:
                continue

            acc_loss += df_copy.at[t, '发电损失']
            di_mean = df_copy.at[t, 'DI 均值']
            triggered = False

            if acc_loss >= C_clean:
                triggered = True
            elif di_mean > theta_safe:
                triggered = True

            if triggered:
                df_copy.at[t, '清洗建议'] = True
                cleaned_times += 1
                acc_loss = 0
                last_clean_time = t  # 设置当前清洗时间，启动冷却期

        result[round(price, 1)] = cleaned_times
    return result


def process_station(station_name, config):
    print(f"\n>>> 正在处理: {station_name}")
    # 文件路径
    power_file = f"A2/A2_发电数据/{station_name}发电数据_每小时汇总.xlsx"
    env_file = f"A2/A2_辐射数据/{station_name}环境检测仪数据_每小时汇总.xlsx"
```

```
    weather_file = f"A2/A2_天气数据/{station_name}天气数据整合_每小时汇总.xlsx"

    # 读取数据
    power_df = pd.read_excel(power_file, parse_dates=['时间'])[['时间', '修复累计发电量（每日归零）']].rename(
        columns={'修复累计发电量（每日归零）': '累计发电量'})
    env_df = pd.read_excel(env_file, parse_dates=['时间'])[['时间', '辐照强度']]
    weather_df = pd.read_excel(weather_file, parse_dates=['时间'])[['时间', '当前温度', '风速', '湿度']].rename(
        columns={'当前温度': '环境温度'})

    # 合并数据
    df = power_df.merge(env_df, on='时间').merge(weather_df, on='时间')
    df = df.set_index('时间').sort_index()
    df['实际功率'] = df['累计发电量'].diff().fillna(0).clip(lower=0)

    # 参数
    cap = config['capacity']
    beta = config['tilt']
    area = cap / 0.18 / 1000
    eta = cap / area / 1000
    tref = 25
    alpha1 = 0.004
    beta_h = 0.02
    k = 0.2 if beta > 0 else 0
    ifactor = 1 - k * np.sin(np.radians(beta))
    clean_date = pd.to_datetime(config['clean_date'])

    # 拟合模型
    train_df = df[(df.index.date >= clean_date.date()) & (df.index.date <= (clean_date + timedelta(days=2)).date())]
    valid = train_df[(train_df['辐照强度'] > 200) & (train_df['实际功率'] > 0)]
    def model_func(xdata, h0, h1):
        G, T_env, H, W = xdata
        T_back = T_env + G / (h0 + h1 * W)
        return eta * G * (1 - alpha1 * (T_back - tref)) * (1 - beta_h * H) * ifactor
    xdata = np.array([valid['辐照强度'], valid['环境温度'], valid['湿度'], valid['风速']])
    ydata = valid['实际功率'].values
    popt, _ = curve_fit(model_func, xdata, ydata, p0=[20, 5], maxfev=10000)
    h0_fit, h1_fit = popt

    # DI 计算
    df['理论功率
```

```
    '] = df.apply(lambda row: compute_theoretical_power(row, eta, alpha1,
    beta_h, tref, h0_fit, h1_fit, ifactor), axis=1)
115.     df['DI'] = (df['理论功率'] - df['实际功率']) / df['理论功率
    '].replace(0, np.nan) * 100
116.     df['DI'] = df['DI'].clip(lower=0)
117.     df['DI 均值'], df['DI 波动'] = sliding_statistics(df['DI'])
118.
119.     # 模拟不同价格下的清洗次数
120.     price_range = np.arange(0.1, 5.1, 0.1)  #
121.     result = dynamic_cleaning_count(df, cap, price_range)
122.
123.     # 保存 CSV
124.     os.makedirs("A3/ANS_2", exist_ok=True)
125.     pd.DataFrame(list(result.items()), columns=['清洗价格(元/kW)', '清
    洗次数']).to_excel(f"A3/ANS_2/{station_name}_价格清洗次数统
    计.xlsx", index=False)
126.
127.     # 绘图
128.     plt.figure(figsize=(10, 5))
129.     plt.plot(list(result.keys()), list(result.values()), marker='o',
    label=station_name)
130.     plt.title(f"{station_name} 不同清洗价格下清洗次数变化趋势")
131.     plt.xlabel("清洗价格 (元/kW)")
132.     plt.ylabel("清洗次数")
133.     plt.grid(True)
134.     plt.tight_layout()
135.     plt.savefig(f"A3/ANS_2/{station_name}_清洗价格影响趋势图.png")
136.     plt.close()
137.
138. # 主程序
139. for name, config in stations.items():
140.     process_station(name, config)
```