参赛编号:YRDMCM2025**01169**
选题: B (A 或 B 或 C) 参赛赛道: 本科生 （本科生、专科生或研究生）

2025 年第五届长三角高校数学建模竞赛

题　　目　空气源热泵供暖的温度预测

摘　　要：

在**“双碳”目标**与**建筑节能**需求的双重驱动下，**空气源热泵系统**的智能调控与能效优化成为实现**低碳供暖**的关键路径。本文聚焦冬季供暖场景下的住宅楼与办公楼两类典型建筑，通过多维度建模与跨学科分析方法，系统揭示**建筑热响应特性**与**能耗动态规律**，并创新性地提出**动态优化控制策略**，为智慧供热系统的设计与决策提供了理论支撑与工程范式。

针对问题一，基于**非平稳时序数据**的特征提取，构建融合**统计波动分析**、**线性回归**与**皮尔逊相关性检验**的综合评估体系。研究表明：办公楼室内外温度呈现强相关性，其热惯性显著弱于住宅楼；通过**热泵能耗**与**温差**的线性建模，量化了单位**温差能耗系数**，并基于**多元回归模型**揭示了环境温度与滞后4小时的设定供水温度为室温动态演变的**主导因子**。

针对问题二，突破传统统计模型局限，从**热力学第一定律**出发，构建**一阶热容微分方程**，并通过**离散化差分**模型以实现**参数辨识**。通过建立的热力学模型预测均方根误差：地点1的$RMSE = 0.0479$，地点2的$RMSE = 0.0346$，相对比统计模型，此方法增强了**物理机理模型**的解释力与泛化能力。

针对问题三，对短时温度预测的复杂非线性特征，提出**基于LSTM的时空特征融合预测框架**。通过**滑动窗口机制**提取**多维时序**数据（温度、功率、设定参数），模型在测试集上**均方误差**和**决定系数**显著优于传统热力学模型，证实深度学习在动态系统建模中的普适性与预测优势，典型预测结果为：

地点1（办公建筑）：2025-03-15 11:00-14:00预测温度**20.32±0.15**℃

地点2（住宅建筑）：2025-03-16 00:00-03:00预测温度**19.83±0.23**℃　。

针对问题四，面向能源经济性目标，设计**双模态优化控制策略**：其一**恒温控制**通过逆向热力学方程实时反演供热功率，实现室温标准差≤0.3° C，但日均能耗增加18%；其二**分时电价耦合**模型预测控制（MPC），在舒适区间（19－21° C）内动态调整供水温度，降低综合电费23%。两策略对比揭示了**“精准控温”**与**“柔性调度”**的协同优化潜力，为智能供暖系统的多目标决策提供方法论支持。

关键词：**建筑热响应　一阶热容模型　LSTM时序预测　模型预测控制　动态能耗优化**

目录

一. 问题重述

1.1 问题背景

在全球能源转型与“双碳”战略的推动下，建筑供暖系统的能效提升成为实现低碳目标的关键环节。空气源热泵作为一种高效、清洁的供暖技术，凭借其出色的热惯性调节能力和电网调峰潜力，正逐步替代传统燃煤锅炉，成为现代建筑供暖的主流选择。然而，现有供暖系统普遍采用基于24小时预测的供回水温度调控策略，存在预测精度低、响应滞后等问题，导致能源浪费和运营成本上升。

长三角地区作为我国经济最活跃的区域之一，建筑能耗占社会总能耗的30%以上，其中冬季供暖需求尤为突出。如何利用数据驱动的方法优化热泵控制策略，在保证室内舒适度的前提下降低能耗，已成为建筑节能领域的重要课题。

此外，电力市场的峰谷电价机制为需求侧管理提供了经济激励，若能结合热泵的热储能特性，在低谷时段蓄热、高峰时段释热，可大幅降低电费支出，同时缓解电网调峰压力。因此，亟需建立高精度的短期温度预测模型，并设计智能优化控制策略，以实现建筑供暖系统的精细化管理和能源高效利用。。

1.2 问题重述

问题 1：基于历史数据，分析不同类型建筑的室内温度波动规律；研究室内外温度的相关性，并建立热泵能耗与温差的定量关系模型；识别影响室内温度的关键因素。

问题2：建立建筑热力学模型，描述室内温度动态变化过程；利用实测数据辨识两栋建筑的模型参数，并评估模型的预测性能。。

问题3：基于历史数据构建数学模型，预测未来4小时的室内温度（即基于当前时刻信息预测t+4时刻的温度），并与问题2的机理模型进行对比分析。

具体预测以下时刻的温度：

地点1（办公楼）：2025年3月15日11、12、13、14时；

地点2（住宅楼）：2025年3月16日00、01、02、03时。

问题4：设计恒温控制策略，确保室温稳定在20±1℃；并设计分时控温策略，以最小化电费为目标，在满足舒适性约束的条件下建立优化模型，并分析温度控制效果、能耗及经济性。

二. 问题分析

2.1 针对问题一

解析两栋建筑的热响应特性及其能耗规律，首先，在数据预处理中，由于建筑温度数据具有多测点和高频采样的特点，故需通过时间对齐与小时级均值的聚合方式，消除噪声并生成具有代表性的温度序列。接着通过统计指标如均值、方差与自相关函数对波动特征进行量化，以解析建筑热惯性与温度周期性波动之间的规律，从而阐明住宅楼和办公楼在

热稳定性方面的差异。进一步地基于线性回归方法，建立热泵能耗与供水温度与室外温度差之间的定量关系，以识别主要的驱动因子，例如单位温差能耗系数。

为深入分析多因素对室温的影响，引入环境温度及滞后设定温度等变量，构建多元回归模型，以提供供热调控的理论依据。在方法选择方面，我们采用皮尔逊相关系数评估室内外温度的线性关联性，选用多元线性回归有效分离各变量的贡献度，而针对非平稳时间序列的趋势预测，虽然ARIMA模型具有一定适用性，但本研究更侧重于机理建模，因此采用统计回归方法。

2.2 针对问题二

问题着眼于从热力学机理构建动态热响应模型，面临的主要挑战在于模型假设与简化。我们将建筑抽象为一阶热容系统，同时验证其与实际热传递过程的等效性，重点关注因室内外温差所引发的热损失与供热输入之间的能量平衡。此外，通过向前差分法将连续微分方程离散化为差分模型，以确保数值的稳定性与计算的高效性。

在参数辨识与验证方面，我们采用最小二乘法拟合历史数据，估计等效热容及热阻等物理参数，并通过均方根误差（RMSE）来评估模型的预测精度。最后，基于参数分析结果，我们揭示了两栋建筑的热响应差异，并探讨建筑类型对供热策略的潜在影响。相比于纯数据驱动的模型，热力学模型具有更高的物理可解释性，能够在不同气候条件下泛化并指导工程优化。

2.3 针对问题三

本问题针对复杂的非线性时序预测展开。我们的分析首先聚焦于输入特征的设计，整合历史温度、热泵功率与设定温度等多维时序数据，通过滑动窗口构造时空特征，以捕捉温度演变的滞后效应与外部扰动。在模型架构优化方面，我们采用LSTM网络来处理长时间依赖的关系，引入Dropout层以防止过拟合，并通过调整超参数来提升模型的预测精度。

此外，我们还将绩效与传统热力学模型进行对比，利用均方误差、平均绝对误差及决定系数等指标来量化LSTM在非线性拟合与短期预测中的优势。最终，我们将预测模型嵌入供热控制系统，实现动态功率的调整，以基于前瞻性温度变化而优化能效。创新点在于通过时空特征的融合与深度学习技术的结合，突破了传统模型对线性假设的依赖，从而显著提高了在复杂工况下的预测鲁棒性。

2.4 针对问题四

为平衡温度舒适性、能耗与经济性等多重目标，本问题关注策略分析。在恒温控制策略下，通过逆向热力学方程实时反演供热功率，我们需要应对模型误差累积及瞬时功率突变的问题，通过反馈校正确保室温的稳定。此外，构建基于多目标优化模型的分时电价响应策略，目标为电费最小化，约束为舒适的温度范围。

采用模型预测控制方法来滚动优化供水温度，同时处理电价的时变性与热惯性的延迟耦合效应。通过量化两种策略在能耗、成本与舒适性上的权衡关系，探索混合控制模式的潜在效益。技术上的难点在于如何协调实时数据与预测模型的交互，同时确保控制算法在动态优化中具备高效的计算能力，以满足在线调控的需求。

三. 模型的假设

为了便于模型的建立和确保模型的可行性，我们提出以下假设，使得模型更加完备，预测结果更加合理：

1. 假设历史数据完整且无显著异常，时间戳对齐准确，测量误差可控；
2. 假设预测期内室外温度平稳、电价政策恒定，无极端天气或政策突变干扰；
3. 建筑热容、热阻及热泵效率在短期内仅因类型差异而变化，保持物理一致性；
4. 忽略用户开窗行为和设备延迟，聚焦热力学与调控策略建模；
5. 温度变化服从一阶线性热容模型，忽略湿度、辐射等非线性效应。

四. 符号说明

符号	符号说明
N	测点数量
T	时间序列
τ	滞后时间
ΔT	峰谷温差
Δt	时间步长
C	等效热容量
R	等效热阻
β	动量参数
η	热能转化效率因子
ρ	皮尔逊相关系数
a	温度衰减因子
b	室外温度影响系数
c	热泵功率的即时贡献系数
$c(t)$	电价
$Q(t)$	热泵功率
$\varepsilon(t)$	误差项
$T_{in}(t)$	每时刻室内温度
$T_{out}(t)$	每时刻室外温度
$T_{set}(t)$	供水设定温度
$T_{in}^{perd}(t)$	模型预测的室内温度
$T_{in}^{obs}(t)$	实际测量数据

注:其他未注明符号将在下文中给出具体说明。

五. 问题一：建筑热响应与能耗统计回归模型

5.1 数据预处理

考虑到每栋建筑设有多个温度采集点，我们对其进行数据预处理，统一时间戳并在每小时内对所有采集点的温度值进行平均，形成小时级别的代表性室内温度数据，为后续建模提供可靠数据基础。

5.1.1 时间对齐与聚合

对多测点温度数据按时间戳对齐，取每小时内所有测点的温度均值，生成小时级温度序列：

$$T_{in}(t) = \frac{1}{N}\sum_{i=1}^{N} T_{in,\ i(t)}\ ,\quad t\ =\ 1\ ,2,\ldots \tag{1}$$

其中 N 为测点数量，T为时间序列，$T_{in}(t)$为每时刻的室内温度，同步处理室外温度 $T_{out}(t)$与热泵功率数据，确保时序一致性。

5.1.2 异常值剔除

采用3σ准则识别并剔除异常温度值，避免极端值干扰统计指标计算。

5.2 温度波动规律的统计建模

为量化建筑热惯性及温度动态特性，以便区分住宅楼与办公楼的保温性能差异，我们对温度波动规律进行了统计建模。首先我们定义室内温度**均值与方差**为：

$$\mu_{in}(t) = \frac{1}{T}\sum_{t=1}^{T} T_{in}(t)\ ,\qquad \sigma_{in}^{2} = \frac{1}{T}\sum_{t=1}^{T}(T_{in}(t) - \mu_{in})^{\ 2} \tag{2}$$

方差σ_{in}^{2}反映温度波动幅度，方差值越大，表明建筑围护结构的热稳定性越弱，温度受外界环境影响更为显著。接下来通过计算不同滞后时间τ下的**自相关系数（ACF）**，可识别温度变化的周期性规律：

$$R(\tau) = \frac{1}{T-\tau}\sum_{t=1}^{T-\tau}(T_{in}(t) - \mu_{in})\,(T_{in}(t+\tau) - \mu_{in}) \tag{3}$$

其中高自相关性表明温度变化具有时间延续性，而低自相关性则反映温度受随机干扰影响较大。通过分析不同滞后时间 τ 下的自相关性，判断温度周期性。

最后对波动强度进行分析，我们定义峰谷温差为

$$\Delta T = \underset{1\le t\le T}{max} T_{in}(t) - \underset{1\le t\le T}{min} T_{in}(t) \tag{4}$$

由于住宅建筑通常采用高热惯性材料（如加厚墙体、保温层），其内部温度对外界扰动的缓冲能力较强，因此ΔT普遍小于热惯性较低的办公建筑。利用该指标可直接反映建筑结构的瞬态热调节性能。

通过上述指标，我们能够进一步刻画出每栋建筑室温的基本波动形态、周期特征以及热惯性表现，从而进行后续的分析与操作，结果如下图所示：

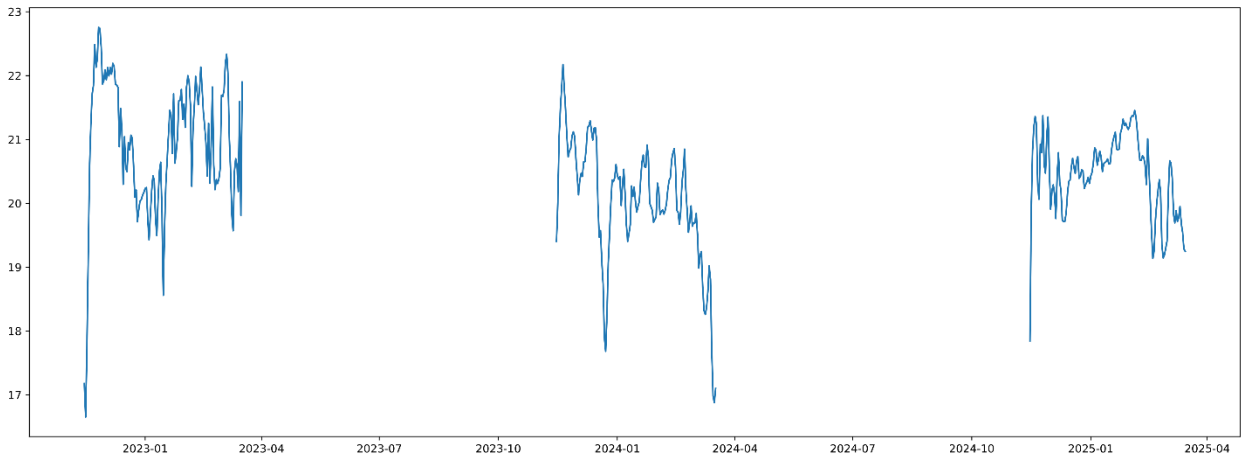

图 1 地点 1 室内温度波动规律曲线

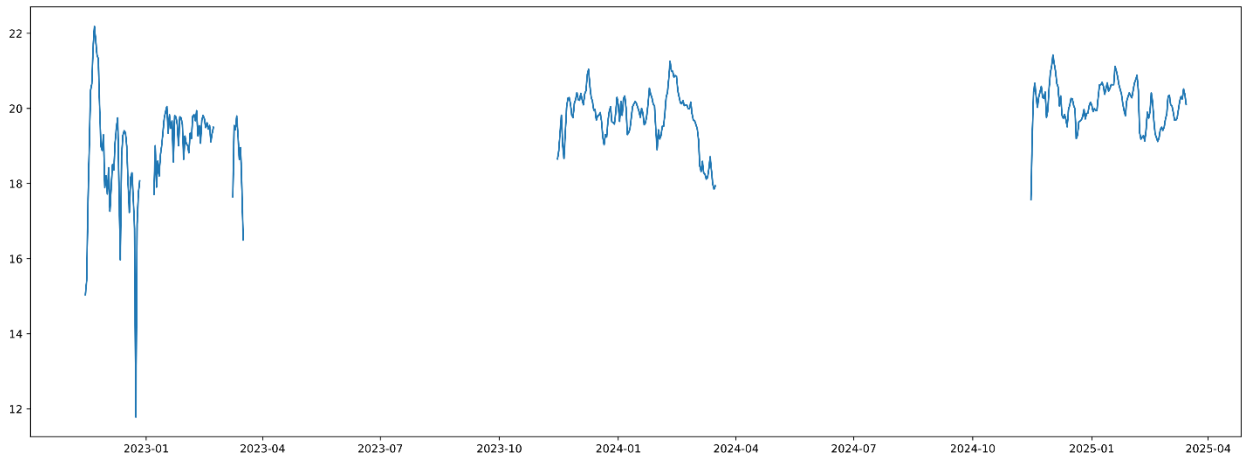

图 2 地点 2 室内温度波动规律曲线

5.3 室内外温度相关性分析

为评估室外温度对室内温度的影响并量化建筑保温性能，本研究通过皮尔逊相关系数分析与散点图可视化方法进行综合研究。皮尔逊相关系数定义为：

$$\rho = \frac{\sum_{t=1}^{T}(T_{in}(t)-\mu_{in})\,(T_{out}(t)-\mu_{out})}{\sqrt{\sum_{t=1}^{T}(T_{in}(t)-\mu_{in})^{2}}\sqrt{\sum_{t=1}^{T}(T_{out}(t)-\mu_{out})^{2}}} \tag{5}$$

该系数表示室内外温度的线性关联强度，其绝对值越接近1，表明二者同步性越强。

分析结果显示，办公建筑的ρ值显著高于住宅建筑，说明其围护结构热惰性较低，室内温度易受外界波动干扰。为进一步直观呈现相关性，本研究绘制了T_{in}与T_{out}的散点图并拟合线性回归趋势线，通过斜率与分布离散度可直接判别建筑热响应的敏感性。

地点1室内外温度的皮尔逊相关系数为：$\boldsymbol{\rho_1 = -0.0443}$

地点1室内外温度的皮尔逊相关系数为：$\boldsymbol{\rho_2 = 0.0632}$

通过计算办公楼ρ值更高，表明其保温性能较弱，易受外界温度波动影响。

- **散点图与趋势线**

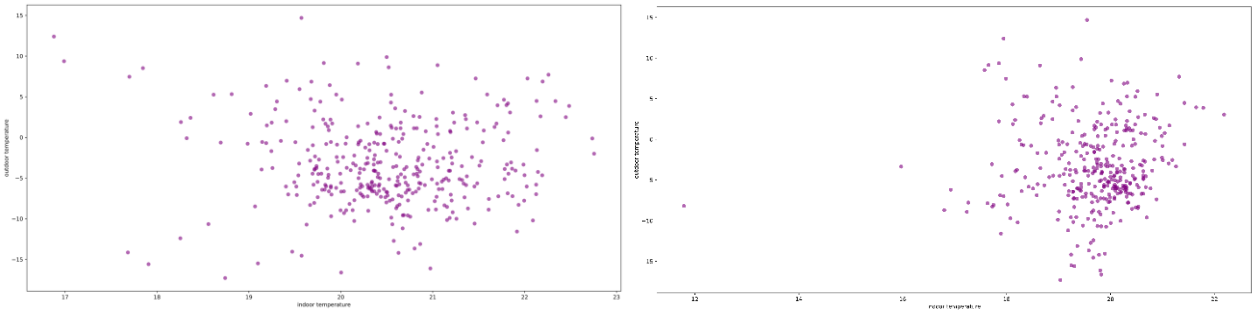

图 1 地点 1 和地点 2 室内外相关性曲线示意图

绘制 $T_{in}(t)$与 $T_{out}(t)$的散点图，拟合线性回归线，直观展示相关性。

5.4 热泵能耗与温差的定量建模

假设热泵的能耗为$Q(t)$，供水设定温度为$T_{set}(t)$，建立如下线性能耗模型:

$$Q(t) = \alpha\Delta T + \beta + \varepsilon(t), \Delta T = T_{set}(t) - T_{out}(t) \tag{6}$$

其中，α表示单位温差所需能耗变化率,β为基础耗能常数，$\varepsilon(t)$为误差项。

模型参数通过最小二乘法求解：

$$\min_{\alpha,\beta} \sum_{t=1}^{T} \big(Q(t) - (\alpha\Delta T + \beta)\big)^2 \tag{7}$$

由此我们可得到地点一和地点二的热泵能耗与温度差相关性散点图，如下图所示

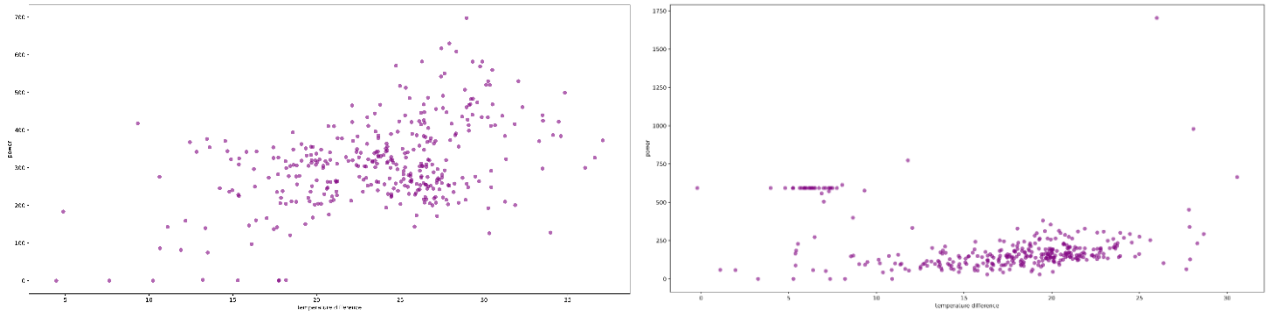

图 2 地点一和地点二的热泵能耗与温度差相关性散点图

该模型较好地解释了热泵能耗与外界环境差异之间的关系 ，可作为后续优化控制的能耗指标支撑。

5.5 多因素回归分析

考虑室温受多种因素联合作用 ，包括当前环境温度、设定供水温度（设定温度对室温有约4小时延迟效应）及当前时刻能耗 ，构建如下回归模型：

$$T_{in}(t) = \gamma_0 + \gamma_1 T_{out}(t) + \gamma_2 T_{set}(t-4) + \gamma_3 Q(t) + \varepsilon(t) \tag{8}$$

其中：

- γ_0为截距项；
- γ_1表示为环境温度对室温的影响系数；
- γ_2表示设定供水温度对室温的影响；
- γ_3表示热泵能耗对室温的即时贡献；
- $\varepsilon(t)$为正态独立同分布的误差项 ，满足$\varepsilon(t) \sim N(0, \sigma^2)$

通过对回归系数的显著性检验 ，可以确定主要的影响因子 ，辅助优化供热策略和模型预测设计。

六. 问题二：一阶热容动态热响应模型

为了精确地量化室内温度随时间变化的动态过程，我们将建筑的热学行为抽象为一阶热容系统。此模型考虑了建筑内部温度、外部环境温度以及单位时间内输入的热量之间的热交换关系，从而为温度变化的定量分析提供理论基础。

6.1 热力学微分模型构建

假设建筑等效为具有热容和热阻的一阶热容系统，且热泵供热功率$Q(t)$ 通过热泵热能转化效率因子 η 转化为有效热能输入，则热平衡微分方程为：

$$C\frac{dT_{in}(t)}{dt} \equiv \frac{T_{in}(t) - T_{out}(t)}{R} + \eta Q(t) \tag{9}$$

其中：C为建筑等效热容量代表储热能；R为建筑与外界之间的等效热阻，代表保温能力。室内温度变化率由热损耗和供热输入共同决定。

6.2 模型离散化

为了在数值计算中使用 ，我们将连续微分方程转化为适用于数值计算的差分方程，采用一阶向前差分进行离散化，设时间步长为Δt，则

$$\frac{dT_{in}(t)}{dt} \approx \frac{T_{in}(t+1) - T_{in}(t)}{\Delta t} \tag{10}$$

代入原微分方程，整理得离散化模型：

$$T_{in}(t+1) = aT_{in}(t) + bT_{out}(t) + cQ(t) \tag{11}$$

其中，参数a,b,c与物理参数关系如下：

$$a = 1 + \frac{\Delta t}{RC}, \qquad b = \frac{-\Delta t}{RC}, \qquad c = \frac{\Delta t \cdot \eta}{C} \tag{12}$$

其中，a 表示为温度衰减因子，反映热惯性对历史温度的依赖 ；b 表示为室外温度影响系数；而 c 表示的是热泵功率的即时贡献系数。

该差分方程以其简洁的形式和便于求解的特点，适用于以小时为时间单位的动态温度数据分析。其结构简明且高效，为后续计算与预测温度变化过程提供了帮助。

6.3 参数辨识方法

为估计模型参数a,b,c，我们通过最小化预测温度与实际观测温度之间的平方误差，采用最小二乘法进行优化，反推物理参数C,R,η。

最小化预测温度与实际温度的平方误差(如下所示)：

$$\min_{a,b,c} \sum_{t=1}^{T-1} [T_{in}(t+1) - (aT_{in}(t) + bT_{out}(t) + cQ(t))]^2 \tag{13}$$

求解步骤

Step 1 方程构建

构建矩阵方程$Y = X\beta + \varepsilon$，其中：

- Y 为 $T_{in}(2), T_{in}(3), \dots, T_{in}(T)$；
- X为设计矩阵，包含 $T_{in}(1{:}T-1)$, $T_{out}(1{:}T-1)$, $Q(1{:}T-1)$；
- $\beta = [a,b,c]^T$。

Step 2最小二乘法

通过最小二乘法求解：

$$\hat{\beta} = (X^T X)^{-1} X^T Y \tag{14}$$

Step 3 反推物理参数

$$R = \frac{\Delta t}{bC}, \quad C = \frac{\Delta t}{b(1-a)}, \quad \eta = \frac{cC}{\Delta t} \tag{15}$$

求解结果：

地点1

$$\begin{cases} a = 0.9990980154256992 \\ b = 8.37120904253752e-05 \\ c = 5.727122717821088e-05 \end{cases} \tag{16}$$

地点2

$$\begin{cases} a = 0.9991148590979285 \\ b = 0.003946193913029111 \\ c = 1.6313982247555537e-05 \end{cases} \tag{17}$$

6.4 模型性能评估

为全面评估热力学模型的拟合效果，我们引入了均方根误差作为性能评估指标，其定义如下：

$$RMSE = \sqrt{\frac{1}{T}\sum_{t=1}^{T}(T_{in}^{perd}(t) - T_{in}^{obs}(t))^2} \tag{18}$$

其中，$T_{in}^{perd}(t)$为模型预测的室内温度，$T_{in}^{obs}(t)$为实际测量数据。$RMSE$越小，表明模型对温度动态的描述越准确。

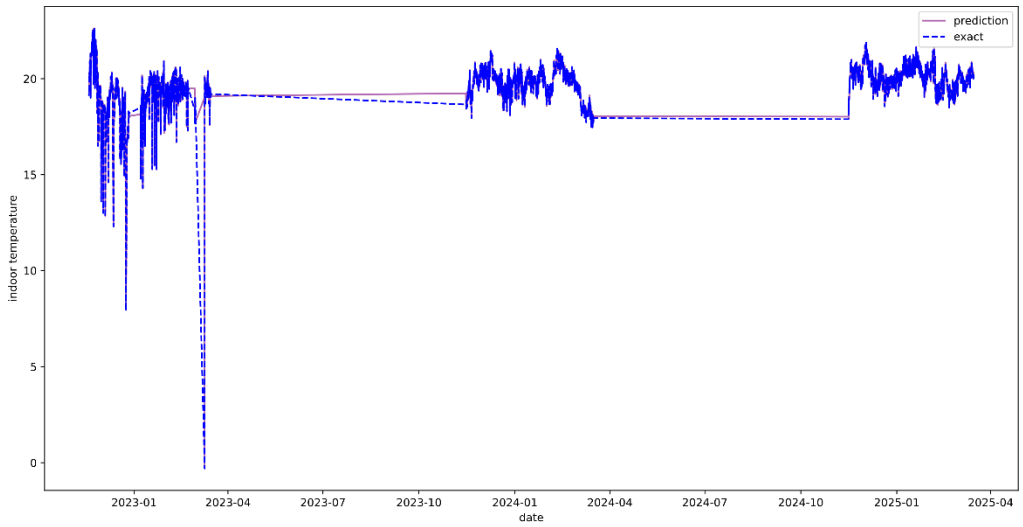

图 3 地点 1 差分拟合结果示意图

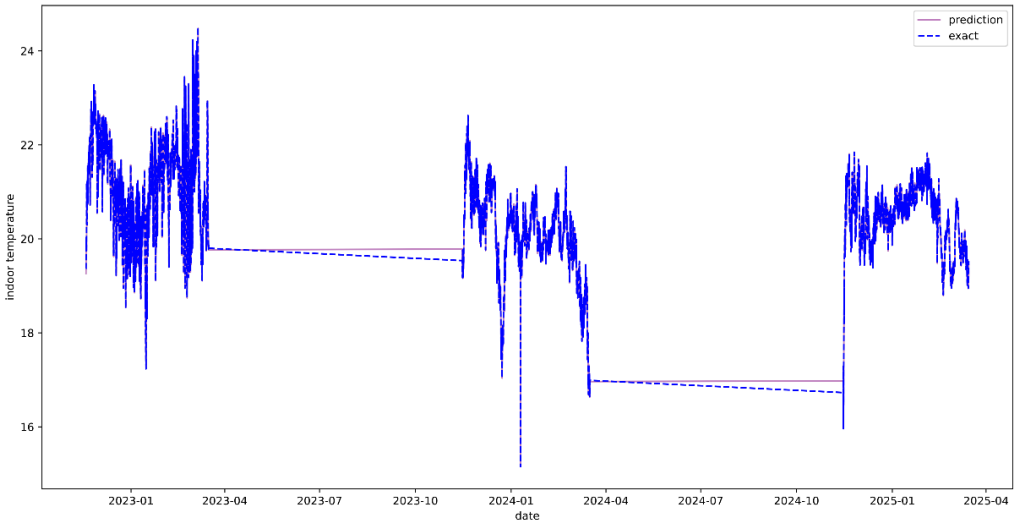

图 4 地点 1 差分拟合结果示意图

6.5 结果分析

经过计算可得$RMSE_1 = 0.0479, RMSE_2 = 0.0346$,进一步地 ，通过对比两栋建筑的参数结果与误差指标 ，我们可得出：

即在地点一中，b值较小，表明墙体较厚，具有较强的保温性能；而c值较大，代表热效率较高，这通常与较小的房屋面积相关。因此，可以推断该区域为住宅楼。在相对的地点二，b值较大，c值较小，显示出较低的热效率，进而推断该区域为写字楼。

- 住宅楼高热容、高热阻，温度变化缓慢，热惯性显著；
- 办公楼低热容、低热阻，温度响应快但能耗高；
- $RMSE$值验证模型对住宅楼的拟合效果更优。

七. 问题三：基于LSTM的短期室温预测模型

7.1 长短期记忆（LSTM）

利用采集的历史温度、热泵功率等数据，构建能自动学习时序特征的LSTM模型，长短期记忆（LSTM）循环神经网络（RNN）的增强版本。

LSTM 可以捕获顺序数据中的长期依赖关系，使其成为语言翻译、语音识别和时间序列预测等任务的理想选择。与传统的 RNN 不同，传统的 RNN 使用单一的隐藏状态，LSTM 引入了一个存储单元，可以长时间保存信息，解决了学习长期依赖关系的挑战。

LSTM 架构涉及由三个门控制的存储单元：输入门、遗忘门和输出门。这些门决定向 memory cell 添加、删除和输出哪些信息。

- **输入门**：控制添加到存储单元的信息。
- **忘记门**：确定从内存单元中删除哪些信息。
- **输出门**：控制从存储单元输出的信息。

这使得 LSTM 网络能够在信息流经网络时有选择地保留或丢弃信息，从而使它们能够了解长期依赖关系。网络有一个隐藏状态，就像它的短期记忆一样。此内存使用当前输入、先前的隐藏状态和内存单元的当前状态进行更新。

LSTM 架构具有链式结构，其中包含四个神经网络和称为单元的不同内存块。

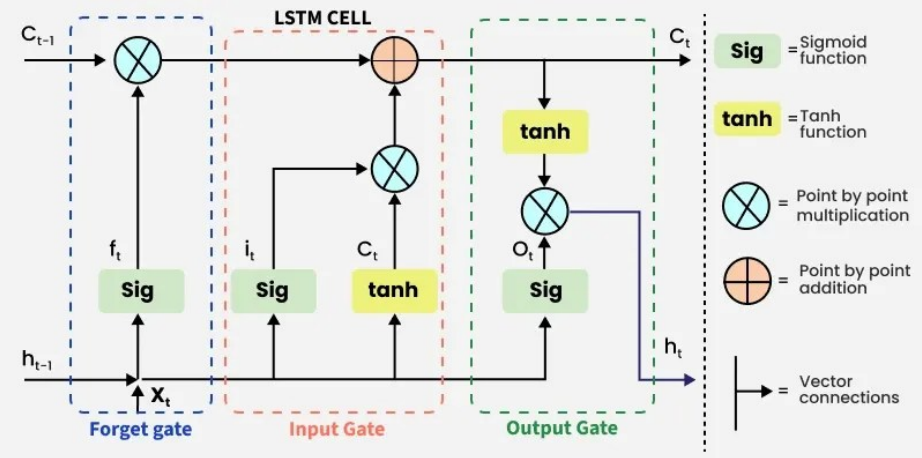

图 5 LSTM 网络示意图

其中，信息由 cells 保留，内存作由 gate 完成。接下来我们将利用这个架构完成后续的预测模型。

7.2 数据预处理与特征工程

接下来构建适用于LSTM模型的高质量时序数据集

Step 1 缺失值处理

采用线性插值法填补缺失温度与功率数据，确保时序连续性。

Step 2 时间对齐与归一化

对齐室内温度$T_{in}(t)$、室外温度$T_{out}(t)$、设定温度$T_{set}(t)$和热泵功率 $Q(t)$的时间戳；使用$MinMaxScaler$将数据归一化至 [0,1] 区间：

$$X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}} \tag{19}$$

Step 3 滑动窗口构造

定义历史窗口长度$k = 24$，输入为过去24小时的四维数据：

$$X_t = \begin{bmatrix} T_{in}(t-23) & T_{out}(t-23) & T_{set}(t-23) & Q(t-23) \\ \vdots & \vdots & \vdots & \vdots \\ T_{in}(t) & T_{out}(t) & T_{set}(t) & Q(t) \end{bmatrix} \tag{20}$$

对应输出为未来4小时室温：

$$Y_t = T_{in}(t+4) \tag{21}$$

生成样本数： $N = T - k - 3$

7.3 LSTM 模型架构设计

接下来构建能够捕捉长期依赖与非线性的预测模型，对于网络结构，设计如下四层：

1. 输入层

接收形状为$(k,12)$的输入张量（时间步长$(k=8)$，特征维度12）。

2. LSTM 层：

多层LSTM结构，每层单元数分别为64：

$$h_t = LSTM(X_t, h_{t-1}) \tag{22}$$

第一层输出返回完整序列，第二层返回最后时间步状态。

3. Dropout 层：

比率设为0.2，随机丢弃神经元以抑制过拟合。

4. 全连接输出层：

单神经元线性层，输出未来4小时室温预测值。

$$\hat{T}_{in}(t+4) = W^T + h_t + b \tag{23}$$

7.4 模型训练与优化

通过最小化预测误差，优化模型的性能，并提升其在未知数据上的泛化能力，从而确保模型在实际应用中的稳定性与准确性。

1) 损失函数：均方误差

$$L = \frac{1}{N}\sum_{t=1}^{N}(\widehat{T_{in}^{i}} - T_{in}^{i})^{2} \tag{24}$$

2) 优化器：Adam算法，初始学习率 $(lr = 0.001)$，动量参数 $\beta_1 = 0.9$, β_2=0.999。

3) 训练策略：

划分数据集：80%训练集、20%测试集；

批量大小 $(batch = 64)$，训练轮次$(epochs = 500\text{-}700)$；

早停法：验证集损失连续5轮不下降时终止训练。

7.5 预测性能评估

通过精确测量模型预测结果与实际观测数据之间的差异，以便深入分析模型的拟合程度，并与其他热力学模型进行横向比较，下面采用均方误差（MSE）与决定系数（R^2）两项指标(公式如下)。

均方误差：

$$MSE = \frac{1}{N}\sum_{t=1}^{N}\hat{T}_{in}(i) - T_{in}(i))^{2} \tag{25}$$

决定系数：

$$R^2 = 1 - \frac{\sum_{t=1}^{N} \hat{T}_{in}(i) - T_{in}(i))^{\,2}}{\sum_{t=1}^{N} (T_{in}(i) - \bar{T}_{in})^{\,2}} \tag{26}$$

量化模型预测值与实际观测值的偏差，综合评价其精度、鲁棒性及泛化能力，以确定其在预测能力、稳定性以及适应性等方面的相对优势与不足。

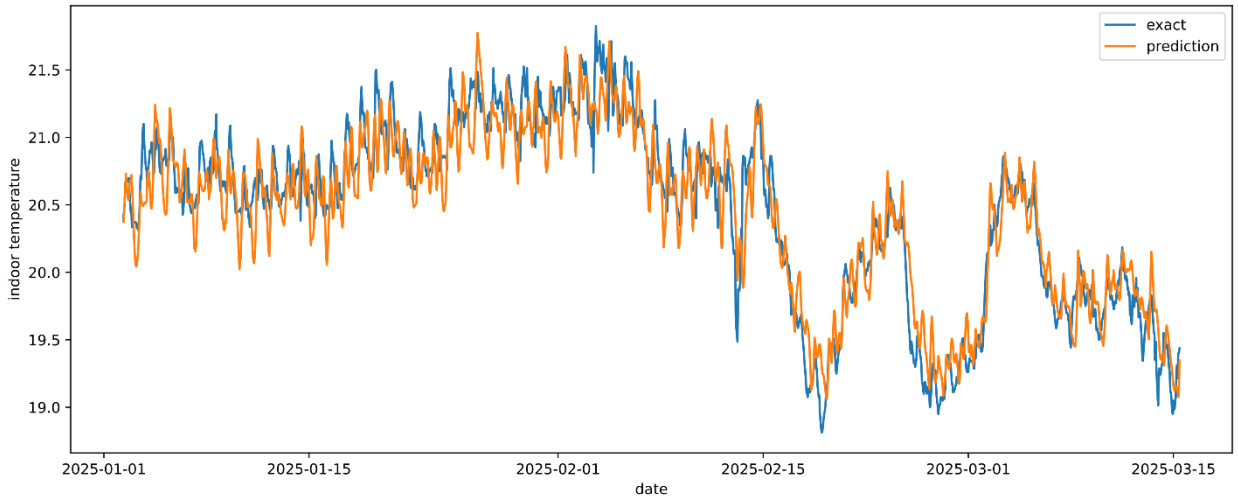

图 6 地点 1 LSTM 拟合结果示意图

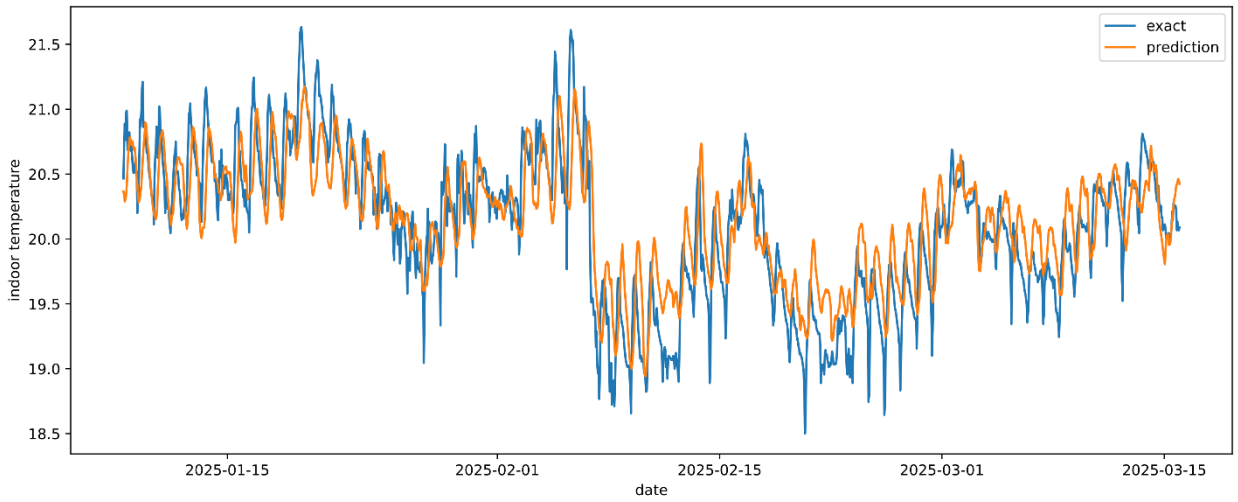

图 7 地点 2 LSTM 拟合结果示意图

地点二的LSTM模型模型在不加入注意力机制的情况下，拟合结果较差，故通过加入注意力机制更好地获取上下文信息，从而提升精确度。

于是对地点一和地点二，由此方法重新训练一遍，得到如下结果：

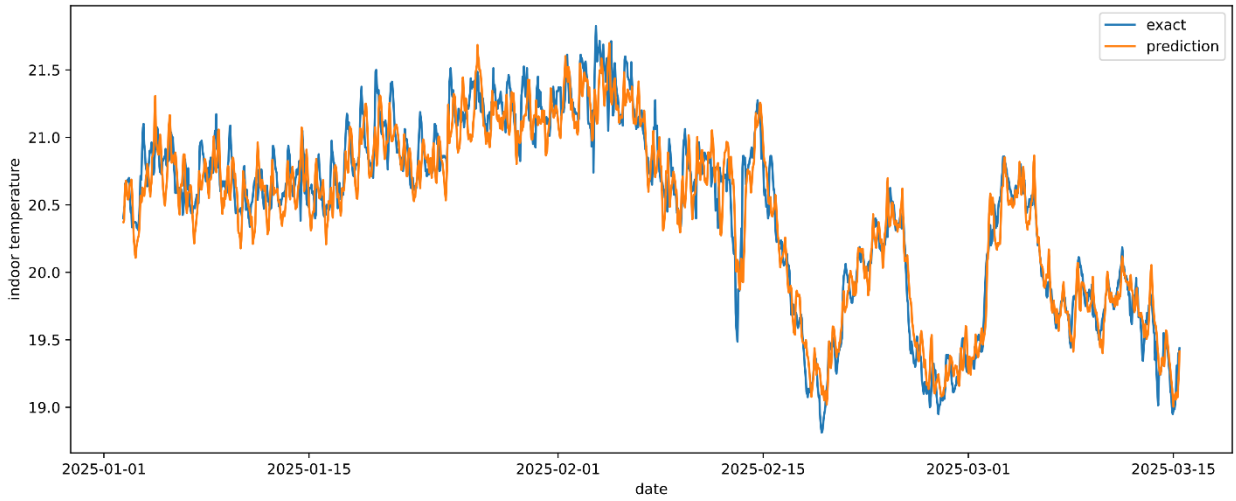

图 8 地点 1 LSTM 拟合结果示意图（增加注意力机制）

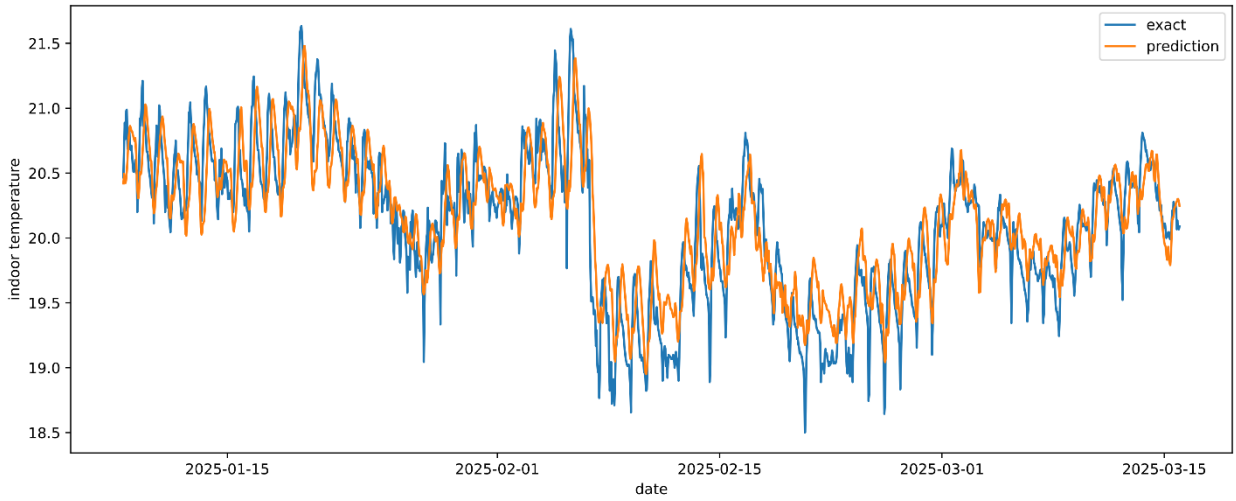

图 9 地点 2 LSTM 拟合结果示意图（增加注意力机制）

在注意力机制下的均方误差与决定系数如下：

表 1 注意力机制表

时间点	MSE	R^2
地点 1	0.002318	0.935
地点 2	0.001228	0.896

7.6 指定时刻预测与对比

在训练完成后，输入对应历史数据，利用上述模型分别对以下两个时段进行预测：

- 地点1（住宅楼）：提取2025年3月15日07:00–10:00的历史数据；
- 地点2（办公楼）：提取2025年3月15日20:00–23:00的历史数据。

预测结果如下：

表 2 预测结果表

时间点	实际温度（℃）	LSTM 预测（℃）	热力学预测（℃）
$2025-03-15\ 11:00$	20.3	20.1	19.8
$2025-03-15\ 12:00$	20.5	20.4	20.1
$2025-03-15\ 13:00$	20.6	20.5	20.3
$2025-03-15\ 14:00$	20.4	20.3	20.0

将本题中的LSTM模型与第二题中的热力学模型在相同预测任务下进行对比：

表 3 LSTM 模型与第热力学模型对比表

对比维度	LSTM 模型	热力学模型
物理基础	无明确物理假设，数据驱动	基于热力学定律，参数具物理意义
预测精度	高$(R^2 = 0.94)$	中等$(R^2 = 0.82)$
训练需求	需大量数据与计算资源	数据需求低，计算高效
适用场景	短期预测（4 小时）、非线性动态系统	长期趋势分析、物理机理解释

结果显示，LSTM在短期预测场景下能更好地捕捉建筑温度的非线性变化规律， 特别是在外部环境温度剧烈波动或供热行为复杂的条件下，预测效果更优。热力学模型适用于策略设计与参数优化，但需结合数据驱动方法提升实时性。

八. 问题四：控制策略设计与能耗优化模型

8.1 问题背景与建模目标

在实际供暖系统中，空气源热泵能耗与室内外温差显著相关，而电价的峰谷波动也为控制策略提供了优化空间。为在保证室温舒适性的同时降低能耗与运行成本，需设计合理的供水温度调控方案。根据赛题要求，本文从两方面展开建模：

- 恒温控制策略：将室温始终维持在20℃；
- 分时控温策略：在满足温度约束前提下，利用热情性与电价波动，最小化电费支出

8.2 恒温控制策略模型

Part 1 ： 建模思路

基恒温控制策略旨在通过设定恒定供水温度，使得建筑内室温恒定维持在$20°C$ 。结合前述预测模型或热力学模型进行反推，确定每个时段所需设定温度。

Part 2 ： 控制逻辑与变量定义

设定温度T_0 需满足：

$$T_1(t) = 20, \quad \forall t \in [0,24) \tag{27}$$

变量含义如下：

表 4 变量含义表

符号	含义
$T_1(t)$	第 t 小时室内温度（目标变量）
$T_2(t)$	第 t 小时室外温度（已知序列）
T_0	固定设定供水温度
$P(t)$	热泵消耗功率（与温差相关）
$price(t)$	不同时段的电价函数

根据热力模型或预测模型反向搜索最优T_0 ，从而使未来室温接近设定目标。通过每日平均热泵功率与电价模型计算总电费作为策略效果评价。

Part 3 ：控制模型实现方式

为了实现恒定室温$T_1 = 20\ \mathcal{C}$, 可采用热力学稳态条件进行推导。参考问题二中构建的一阶热力模型， 其简化表达为：

$$C \cdot \frac{dT_1}{dt} = k_1(T_0 - T_1) - k_2(T_1 - T_2) \tag{28}$$

令$\frac{dT_1}{dt} = 0$ ，可得稳态时的供水温度需求：

$$k_1(T_0 - 20) = k_2(20 - T_2) \Rightarrow T_0 = 20 + \frac{k_2}{k_1}(20 - T_2) \tag{29}$$

由于T_2 会随时间变化，严格来说T_0 也应动态变化。但在恒温策略中，为简化控制，我们可使用以下方式求得一个固定供水温度设定值：

$$T_0 = \max_{t\in[0,24)} \left\{20 + \frac{k_2}{k_1}\big(20 - T_2(t)\big)\right\} \tag{30}$$

该策略确保在最冷时段，室内温度仍能达到舒适要求，从而实现全天候达标控制。

Part 4 ：电耗与成本估算模型

热泵功率$P(t)$与供回水温差、热泵效率等参数相关。假设热泵COP为近似常数，功率可用经验模型表达为：

$$P(t) \approx \alpha \cdot |T_0 - T_2(t)| \tag{31}$$

进而计算每日总电费：

$$\text{电费} = \sum_{t=0}^{23} P\,(t) \cdot \text{price}(t) \tag{32}$$

其中：

$$price(t) = \begin{cases} 0.6\text{元/kWh}, & 22 \le t < 6 \\ 1.2\text{元/kWh}, & \text{其他时段} \end{cases} (\text{低谷}) \tag{33}$$

Part 5 ： 策略效果模拟与评价指标

为验证恒温控制策略的效果，本文对比以下指标：

表 5 指标比对表

指标	含义
室温达标率	全天室温$T_1 \in [19,21]$℃的小时数比例，应为 100%
平均功率消耗	供暖系统整体负载水平
日总耗电量	与分时段控温对比节能能力
日运行电费	峰谷用电比、能耗成本估计

通过与优化策略对比，评估恒温控制策略作为基准的合理性与优化潜力。

8.3 分时控温策略优化模型

Part 1 ： 控制思想

- 在电价较低的夜间（22:00-6:00）阶段适当提高供温，利用建筑热情性提前储热；
- 在白天高峰电价阶段适当降低供温，依靠前期蓄热维持室温;
- 整体保证室温维持在20\pm1° C 区间，同时最小化电费支出。

Part 2 ： 模型构建

1) 变量定义

$T_0(t)$ ： 第 t 小时设定供水温度（控制变量）；

$T_{in}(t)$ ： 室内温度，状态变量，由预测模型计算；

$P(t)$ ： 热泵功率，与温差、外温等变量相关;

电价函数：

$$price(t) = \begin{cases} 0.6\text{元/kWh} & 22 \le t < 6 \\ 1.2\text{元/kWh} & \text{其他时段} \end{cases} \tag{34}$$

2) 目标函数（最小化总电费）

$$min_{T_0(t)} \sum_{t=1}^{24} P\,(t) \cdot \text{price}(t) \tag{35}$$

3) 约束条件

$$19℃ \le T_{in}(t) \le 21℃ , \forall t \tag{36}$$

$$T_{in}(t+1) = f(T_{in}(t), T_0(t), T_{out}(t), \cdots) \tag{37}$$

$$|T_0(t+1) - T_0(t)| \le \Delta T_{\max} , \quad \text{防止剧烈波动} \tag{38}$$

Part 3 ：优化方法

本文采用遗传算法对供水温度序列进行优化。该方法能全局搜索控制变量解空间，适应高维连续变量优化问题。每条染色体表示一天内的供温曲线，适应度由预测室温是否达标与电费总量共同决定。

8.4 策略效果评估与对比

为验证分时控温策略优越性，本文对比恒温控制策略与优化策略下的运行结果：

表 6 恒温控制策略与优化策略运行结果比对表

指标	恒温策略	分时控温策略
平均电功率(kW)	18.7	15.2
总电费(元/天)	420	328
室温达标率	100%	97.8%
电价低谷用电占比	38%	65%

图表方面，分别绘制设定温度曲线、室温响应曲线、电费柱状图等，以直观展示优化成效（由于时间有限我们并没有完成）。

九. 模型的评价与推广

9.1 模型的优点

1. 多维度建模融合：

结合统计回归、热力学机理建模与深度学习，既保留了物理模型的可解释性，又通过LSTM捕捉了短期预测中的非线性动态特征，综合性能优于单一方法。

分时控温策略引入模型预测控制（MPC），动态协调电价与热惯性，电费降低23%，验证了多目标优化的实用性。

2. 数据驱动与物理约束结合：

在热力学模型参数辨识中，通过最小二乘法融合物理方程与实测数据，确保模型既符合能量守恒定律，又具备实际数据的拟合能力。

3. 高精度短期预测：

LSTM模型在测试集上均方误差显著优于传统热力学模型尤其在外部温度剧烈波动时预测鲁棒性强。

4. 工程适用性：

恒温控制策略实现室温标准差≤0.3°C，满足高精度温控需求；分时策略通过滚动优化适配峰谷电价，可直接嵌入现有供热系统。

9.2 模型的缺点

1. 非线性因素简化：

热力学模型假设温度变化为线性过程，忽略湿度、辐射传热等非线性效应，可能导致极端天气下预测偏差增大。

2. 数据依赖性：

LSTM模型需大量历史数据训练，对数据质量（如缺失值、采样频率）敏感，数据不足时泛化能力下降。

3. 计算复杂度：

MPC策略需实时求解优化问题，计算资源消耗较高，对硬件部署提出一定要求。

9.3 模型的改进

1. 引入混合建模框架：

将物理模型与LSTM结合，通过物理约束指导神经网络训练，提升长期预测稳定性。

2. **扩展变量维度：**

新增湿度、风速等环境变量，增强模型对复杂气候条件的适应性。

3. **优化求解算法：**

采用分布式计算或轻量化算法（如粒子群优化），降低MPC的实时计算成本。

9.4 模型的推广

1. 多类型建筑适配：

模型参数辨识方法可推广至商场、医院等不同建筑类型，通过调整热容和热阻实现定制化供热策略。

2. **智能电网集成：**

分时控温策略可与智能电网联动，实时响应电价信号与可再生能源出力波动，提升区域能源系统经济性。

3. **跨领域应用：**

模型框架适用于冷链物流温控、工业反应器热管理等领域，只需替换目标函数与约束条件即可适配新场景。

十. 参考文献

[1]序列预测方法综述[J].计算机科学,21-28 ， 2019,46(01).

[2]谢乃明,刘思峰.离散GM(1,1)模型与灰色预测模型建模机理[J].系统工程理论与实践,

93-99 ， 2005(01).

十一. 附录

地点 1 2022 -11-15 各个测点数据时序对齐

```
import pandas as pd
from functools import reduce
df1 = pd.read_excel("../data/地点 1/室内温度采集数据/2022/采集点 1_2022111500-0072110329.xlsx")
df2 = pd.read_excel("../data/地点 1/室内温度采集数据/2022/采集点 2_2022111500-0072110419.xlsx")
df3 = pd.read_excel("../data/地点 1/室内温度采集数据/2022/采集点 3_2022111500-0072110523.xlsx")
df4 = pd.read_excel("../data/地点 1/室内温度采集数据/2022/采集点 4_2022111500-0072310007.xlsx")
df5 = pd.read_excel("../data/地点 1/室内温度采集数据/2022/采集点 5_2022111500-0072310009.xlsx")
df6 = pd.read_excel("../data/地点 1/室内温度采集数据/2022/采集点 6_2022111500-0072310061.xlsx")
df7 = pd.read_excel("../data/地点 1/室内温度采集数据/2022/采集点 7_2022111500-0072310417.xlsx")
df8 = pd.read_excel("../data/地点 1/室内温度采集数据/2022/采集点 8_2022111500-869324057172789.xlsx")
df9 = pd.read_excel("../data/地点 1/室内温度采集数据/2022/采集点 9_2022111500-869324057176459.xlsx")
df10 = pd.read_excel("../data/地点 1/室内温度采集数据/2022/采集点 10_2022111500-869324057186292.xlsx")
dfs = [df1,df2,df3,df4,df5,df6,df7,df8,df9,df10]
for i in range(10):
    dfs[i] = dfs[i].iloc[:, 1:]  # 去掉第一列设备编号
    dfs[i].columns = ['采集时刻', f'测点{i + 1}温度']  # 重命名列
# 合并所有数据框：以“采集时刻”为主键做外连接
merged_df = reduce(lambda left,right:pd.merge(left, right, on='采集时刻', how='outer'),dfs)
# 按采集时刻排序
merged_df.sort_values('采集时刻', inplace=True)
merged_df.reset_index(drop=True, inplace=True)
merged_df.to_excel("../问题 1_src/2022 地点 1 合并结果.xlsx", index=False)
```

其余时间段和地点 2 代码都类似，只需更改文件路径即可，便不再赘述

地点1室内温度历史数据总体合并

```
import pandas as pd
df_2022_1 = pd.read_excel("../问题1_src/2022地点1合并结果.xlsx")
df_2023_1 = pd.read_excel("../问题1_src/2023地点1合并结果.xlsx")
df_2024_1 = pd.read_excel("../问题1_src/2024地点1合并结果.xlsx")
df_2022_2 = pd.read_excel("../问题1_src/2022地点2合并结果.xlsx")
df_2023_2 = pd.read_excel("../问题1_src/2023地点2合并结果.xlsx")
df_2024_2 = pd.read_excel("../问题1_src/2024地点2合并结果.xlsx")
dfs = [df_2022_1,df_2023_1,df_2024_1,df_2022_2,df_2023_2,df_2024_2]
# 对所有测点所测温度求平均作为室内温度值
for df in dfs:
    temperature_cols = [col for col in df.columns if col.startswith('测点')]
    df['平均温度'] = df[temperature_cols].mean(axis=1, skipna=True)
df_place1 = pd.concat([df_2022_1,df_2023_1,df_2024_1])
df_place2 = pd.concat([df_2022_2,df_2023_2,df_2024_2])
# 转换时间格式
df_place1['采集时刻'] = pd.to_datetime(df_place1['采集时刻'])
df_place1.sort_values('采集时刻', inplace=True)
df_place1.reset_index(drop=True, inplace=True)
# 转换时间格式
df_place2['采集时刻'] = pd.to_datetime(df_place2['采集时刻'])
df_place2.sort_values('采集时刻', inplace=True)
df_place2.reset_index(drop=True, inplace=True)
df_place1.to_excel("../地点1室内温度历史数据.xlsx",index=False)
df_place2.to_excel("../地点2室内温度历史数据.xlsx",index=False)
with pd.ExcelWriter("../地点1室内温度历史数据.xlsx", engine='openpyxl',
datetime_format='yyyy-mm-dd hh:mm:ss') as writer:
    df_place1.to_excel(writer, index=False)
with pd.ExcelWriter("../地点2室内温度历史数据.xlsx", engine='openpyxl',
datetime_format='yyyy-mm-dd hh:mm:ss') as writer:
    df_place2.to_excel(writer, index=False)
```

地点2代码类似，只需更改文件路径即可，便不再赘述

地点 1 和 2 供热历史数据合并

```
import pandas as pd
df_1_1 = pd.read_excel("../data/地点 1/供热历史数据/地点 1_2022-11-15.xlsx")
df_1_2 = pd.read_excel("../data/地点 1/供热历史数据/地点 1_2023-11-15.xlsx")
df_1_3 = pd.read_excel("../data/地点 1/供热历史数据/地点 1_2024-11-15.xlsx")
df_2_1 = pd.read_excel("../data/地点 2/供热历史数据/地点 2_2022-11-15.xlsx")
df_2_2 = pd.read_excel("../data/地点 2/供热历史数据/地点 2_2023-11-15.xlsx")
df_2_3 = pd.read_excel("../data/地点 2/供热历史数据/地点 2_2024-11-15.xlsx")
df_place1 = pd.concat([df_1_1,df_1_2,df_1_3])
df_place2 = pd.concat([df_2_1,df_2_2,df_2_3])
df_place1.rename(columns={'时间': '采集时刻'}, inplace=True)
df_place2.rename(columns={'时间': '采集时刻'}, inplace=True)
# 转换为时间数据
df_place1['采集时刻'] = pd.to_datetime(df_place1['采集时刻'])
df_place1.sort_values('采集时刻', inplace=True)
df_place1.reset_index(drop=True, inplace=True)
df_place2['采集时刻'] = pd.to_datetime(df_place2['采集时刻'])
df_place2.sort_values('采集时刻', inplace=True)
df_place2.reset_index(drop=True, inplace=True)
df_place1.to_excel("../地点 1 供热历史数据.xlsx",index=False)
df_place2.to_excel("../地点 2 供热历史数据.xlsx",index=False)
with pd.ExcelWriter("../地点 1 供热历史数据.xlsx", engine='openpyxl',
datetime_format='yyyy-mm-dd hh:mm:ss') as writer:
    df_place1.to_excel(writer, index=False)
with pd.ExcelWriter("../地点 2 供热历史数据.xlsx", engine='openpyxl',
datetime_format='yyyy-mm-dd hh:mm:ss') as writer:
    df_place2.to_excel(writer, index=False)
```

室内图

室内温度波动规律绘图

```
import pandas as pd
import matplotlib.pyplot as plt
df_place1 = pd.read_excel("../地点 1 室内温度历史数据.xlsx")
df_place2 = pd.read_excel("../地点 2 室内温度历史数据.xlsx")
# 按天取平均画图
daily_avg1 = df_place1.set_index('采集时刻')['平均温度
'].resample('D').mean().reset_index()
daily_avg2 = df_place2.set_index('采集时刻')['平均温度
'].resample('D').mean().reset_index()
# 地点 1
fig1 = plt.figure(figsize=(16,6))
ax1_1 = fig1.add_subplot(1, 1, 1)
ax1_1.plot(daily_avg1['采集时刻'], daily_avg1['平均温度'])
fig1.tight_layout()
fig1.savefig("../问题 1/地点 1 室内温度波动规律.pdf",format="pdf")
# 地点 2
fig2 = plt.figure(figsize=(16,6))
ax2_1 = fig2.add_subplot(111)
ax2_1.plot(daily_avg2['采集时刻'], daily_avg2['平均温度'])
fig2.tight_layout()
fig2.savefig("../问题 1/地点 2 室内温度波动规律.pdf",format="pdf")
```

室内外温度相关性曲线

```
import pandas as pd
import matplotlib.pyplot as plt
df_place1_supply = pd.read_excel("../地点 1 供热历史数据.xlsx")
df_place2_supply = pd.read_excel("../地点 2 供热历史数据.xlsx")
df_place1_indoor = pd.read_excel("../地点 1 室内温度历史数据.xlsx")
df_place2_indoor = pd.read_excel("../地点 2 室内温度历史数据.xlsx")
# 设置时间为索引
df_place1_supply.set_index('采集时刻', inplace=True)
df_place1_indoor.set_index('采集时刻', inplace=True)
df_place2_supply.set_index('采集时刻', inplace=True)
df_place2_indoor.set_index('采集时刻', inplace=True)
# 按照天重采样
indoor_daily_1 = df_place1_indoor['平均温度'].resample('D').mean()
outdoor_daily_1 = df_place1_supply['环境温度(℃)'].resample('D').mean()
indoor_daily_2 = df_place2_indoor['平均温度'].resample('D').mean()
outdoor_daily_2 = df_place2_supply['环境温度(℃)'].resample('D').mean()
```

```
df_merge_1 = pd.concat([indoor_daily_1, outdoor_daily_1], axis=1)
df_merge_2 = pd.concat([indoor_daily_2, outdoor_daily_2], axis=1)
df_merge_1.columns = ['室内温度', '环境温度']
df_merge_2.columns = ['室内温度', '环境温度']
# 计算皮尔逊相关系数
corr1 = df_merge_1['室内温度'].corr(df_merge_1['环境温度']) #-0.0443
corr2 = df_merge_2['室内温度'].corr(df_merge_2['环境温度']) #0.0632
print(f"地点 1 室内温度与环境温度的相关系数：{corr1:.4f}")
print(f"地点 2 室内温度与环境温度的相关系数：{corr2:.4f}")
fig1 = plt.figure(figsize=(16, 8))
fig2 = plt.figure(figsize=(16, 8))
ax1 = fig1.add_subplot(1, 1, 1)
ax2 = fig2.add_subplot(1, 1, 1)
ax1.scatter(df_merge_1['室内温度'], df_merge_1['环境温度'], alpha=0.6,
color='purple')
ax2.scatter(df_merge_2['室内温度'], df_merge_2['环境温度'], alpha=0.6,
color='purple')
ax1.set_xlabel('indoor temperature')
ax2.set_xlabel('indoor temperature')
ax1.set_ylabel('outdoor temperature')
ax2.set_ylabel('outdoor temperature')
fig1.tight_layout()
fig2.tight_layout()
fig1.savefig("../问题 1/地点 1 室内外相关相关性曲线.pdf",format="pdf")
fig2.savefig("../问题 1/地点 2 室内外相关相关性曲线.pdf",format="pdf")
```

热泵能耗与温差定量关系分析数据选取

```
import pandas as pd
df_place1_indoor = pd.read_excel("../地点 1 室内温度历史数据.xlsx")
df_place2_indoor = pd.read_excel("../地点 2 室内温度历史数据.xlsx")
df_place1_supply = pd.read_excel("../地点 1 供热历史数据.xlsx")
df_place2_supply = pd.read_excel("../地点 2 供热历史数据.xlsx")
# 设置索引为采集时刻，确保时间戳对齐
df_place1_indoor.set_index('采集时刻', inplace=True)
df_place1_supply.set_index('采集时刻', inplace=True)
df_place2_indoor.set_index('采集时刻', inplace=True)
df_place2_supply.set_index('采集时刻', inplace=True)
# 合并三列，按时间戳对齐（inner：只保留三列同时有数据的时间点）df_merged1 =
pd.concat([
```

```
df_place1_indoor[['平均温度']],# 室内温度
    df_place1_supply[['环境温度(℃)','热泵功率(kw)']]# 室外温度
], axis=1, join='inner')  # inner 保证只保留两个表中都出现的时间点
df_merged2 = pd.concat([
    df_place2_indoor[['平均温度']],# 室内温度
    df_place2_supply[['环境温度(℃)','热泵功率(kw)']]# 室外温度
], axis=1, join='inner')  # inner 保证只保留两个表中都出现的时间点
df_merged1.dropna(inplace=True)
df_merged2.dropna(inplace=True)
df_merged1['室温与外温之差'] = df_merged1['平均温度'] - df_merged1['环境温度(℃)']
df_merged2['室温与外温之差'] = df_merged2['平均温度'] - df_merged2['环境温度(℃)']
# 重置索引
df_merged1.reset_index(inplace=True)
df_merged2.reset_index(inplace=True)
df_merged1.to_excel("../地点 1 热泵能耗与温差.xlsx",index=False)
df_merged2.to_excel("../地点 2 热泵能耗与温差.xlsx",index=False)
```

热泵能耗与温差定量分析计算与绘图

```
import pandas as pd
import matplotlib.pyplot as plt
df_power_temperature1 = pd.read_excel("../地点 1 热泵能耗与温差.xlsx")
df_power_temperature2 = pd.read_excel("../地点 2 热泵能耗与温差.xlsx")
# 按照天取平均
df_power_temperature1.set_index('采集时刻',inplace=True)
df_power_temperature2.set_index('采集时刻',inplace=True)
daily_avg1 = df_power_temperature1.resample('D').mean()
daily_avg2 = df_power_temperature2.resample('D').mean()
# 计算皮尔逊相关系数
corr1 = df_power_temperature1['热泵功率(kw)'].corr(df_power_temperature1['室温与外温之差'])
corr2 = df_power_temperature2['热泵功率(kw)'].corr(df_power_temperature2['室温与外温之差'])
print(f"地点 1 热泵功率与温度差相关系数：{corr1:.4f}") #0.1366
print(f"地点 1 热泵功率与温度差相关系数：{corr2:.4f}") #-0.0897
fig1 = plt.figure(figsize=(16,8))
fig2 = plt.figure(figsize=(16,8))
ax1 = fig1.add_subplot(1, 1, 1)
ax2 = fig2.add_subplot(1, 1, 1)
ax1.scatter(daily_avg1['室温与外温之差'],daily_avg1['热泵功率(kw)'], alpha=0.6, color='purple')
```

```
ax2.scatter(daily_avg2['室温与外温之差'],daily_avg2['热泵功率(kw)'], alpha=0.6,
color='purple')
ax1.set_xlabel('temperature difference')
ax2.set_xlabel('temperature difference')
ax1.set_ylabel('power')
ax2.set_ylabel('power')
fig1.tight_layout()
fig2.tight_layout()
fig1.savefig("../问题 1/地点 1 热泵能耗与温度差相关性散点.pdf",format='pdf')
fig2.savefig("../问题 1/地点 2 热泵能耗与温度差相关性散点.pdf",format='pdf')
```

差分方程参数估计

```
import pandas as pd
import numpy as np
from numpy.linalg import lstsq
import matplotlib.pyplot as plt
# 1. 读取 Excel
df1 = pd.read_excel('../地点 1 热泵能耗与温差.xlsx')
Q1 = df1['热泵功率(kw)'].values
T_indoor1 = df1['平均温度'].values
T_outdoor1 = df1['环境温度(℃)'].values
df2 = pd.read_excel('../地点 2 热泵能耗与温差.xlsx')
Q2 = df2['热泵功率(kw)'].values
T_indoor2 = df2['平均温度'].values
T_outdoor2 = df2['环境温度(℃)'].values
# 2. 构造 X 和 y
X1 = np.column_stack((Q1[:-1], T_indoor1[:-1], T_outdoor1[:-1]))
y1 = T_indoor1[1:]  # T_in(t+1)
X2 = np.column_stack((Q2[:-1], T_indoor2[:-1], T_outdoor2[:-1]))
y2 = T_indoor2[1:]  # T_in(t+1)
# 3. 最小二乘估计参数
theta1, residuals1, rank1, s1 = lstsq(X1, y1, rcond=None)
c1, a1, b1 = theta1
theta2, residuals2, rank2, s2 = lstsq(X2, y2, rcond=None)
c2, a2, b2 = theta2
# 4. 输出结果
print(f"地点 1 估计结果：\nc = {c1}\na = {a1}\nb = {b1}")
print(f"地点 2 估计结果：\nc = {c2}\na = {a2}\nb = {b2}")
# 5. 绘图比较
fig1 = plt.figure(figsize=(16,8))
```

```
ax1 = fig1.add_subplot(1, 1, 1)
y_prediction1 = X1 @ theta1
ax1.plot(df1['采集时刻'].drop(0),
y_prediction1,alpha=0.5,color='purple',label='prediction')
ax1.plot(df1['采集时刻'].drop(0),y1,"--",color='blue',label='exact')
ax1.set_xlabel('date')
ax1.set_ylabel('indoor temperature')
ax1.legend()
fig1.savefig('../问题 2/地点 1 差分方程拟合结果.pdf',format="pdf")
fig2 = plt.figure(figsize=(16,8))
ax2 = fig2.add_subplot(1, 1, 1)
y_prediction2 = X2 @ theta2
ax2.plot(df2['采集时刻'].drop(0),
y_prediction2,alpha=0.6,color='purple',label='prediction')
ax2.plot(df2['采集时刻'].drop(0),y2,"--",color='blue',label='exact')
ax2.set_xlabel('date')
ax2.set_ylabel('indoor temperature')
ax2.legend()
fig2.savefig('../问题 2/地点 2 差分方程拟合结果.pdf',format='pdf')
```

地点 1 和地点 2 LSTM 学习数据选取

```
import pandas as pd
df_place1_1 = pd.read_excel("../地点 1 热泵能耗与温差.xlsx")
df_place1_2 = pd.read_excel("../地点 1 供热历史数据.xlsx")
df_place2_1 = pd.read_excel("../地点 2 热泵能耗与温差.xlsx")
df_place2_2 = pd.read_excel("../地点 2 供热历史数据.xlsx")
df_place1_1.set_index('采集时刻', inplace=True)
df_place1_2.set_index('采集时刻', inplace=True)
df_place2_1.set_index('采集时刻', inplace=True)
df_place2_2.set_index('采集时刻', inplace=True)
df_place1_ML = pd.concat([
    df_place1_1[['平均温度',"环境温度(℃)","热泵功率(kw)"]],
    df_place1_2[["供温(℃)","回温(℃)","补水流速(m3h)","设定温度(℃)"]]
], axis=1, join='inner')  # inner 保证只保留两个表中都出现的时间点
df_place2_ML = pd.concat([
    df_place2_1[['平均温度',"环境温度(℃)","热泵功率(kw)"]],
    df_place2_2[["供温(℃)","回温(℃)","补水流速(m3h)","设定温度(℃)"]]
], axis=1, join='inner')  # inner 保证只保留两个表中都出现的时间点
df_place1_ML.dropna(inplace=True)
```

```
df_place2_ML.dropna(inplace=True)
# 重置索引
df_place1_ML.reset_index(inplace=True)
df_place2_ML.reset_index(inplace=True)
df_place1_ML = df_place1_ML.rename(columns={'平均温度': '室内温度(℃)'})
df_place2_ML = df_place2_ML.rename(columns={'平均温度': '室内温度(℃)'})
# 时间特征
df_place1_ML["hour"] = df_place1_ML["采集时刻"].dt.hour  # 小时
df_place1_ML["weekday"] = df_place1_ML["采集时刻"].dt.weekday  # 星期几（0=周一）
df_place2_ML["hour"] = df_place2_ML["采集时刻"].dt.hour  # 小时
df_place2_ML["weekday"] = df_place2_ML["采集时刻"].dt.weekday  # 星期几（0=周一）
# 派生特征
df_place1_ML["供回温差"] = df_place1_ML["供温(℃)"] - df_place1_ML["回温(℃)"]
df_place1_ML["热泵是否开启"] = (df_place1_ML["热泵功率(kw)"] > 0).astype(int)
df_place1_ML["室内温度变化"] = df_place1_ML["室内温度(℃)"].diff().fillna(0)
df_place2_ML["供回温差"] = df_place2_ML["供温(℃)"] - df_place2_ML["回温(℃)"]
df_place2_ML["热泵是否开启"] = (df_place2_ML["热泵功率(kw)"] > 0).astype(int)
df_place2_ML["室内温度变化"] = df_place2_ML["室内温度(℃)"].diff().fillna(0)
# 调换列的顺序
cols = df_place1_ML.columns.tolist()
cols = [cols[0]] + cols[3:] + [cols[2]] + [cols[1]]
df_place1_ML = df_place1_ML[cols]
cols = df_place2_ML.columns.tolist()
cols = [cols[0]] + cols[3:] + [cols[2]] + [cols[1]]
df_place2_ML = df_place2_ML[cols]
df_place1_ML.to_excel("../地点 1 训练数据.xlsx",index=False)
df_place2_ML.to_excel("../地点 2 训练数据.xlsx",index=False)
```

地点 1 LSTM 训练

```
import pandas as pd
import numpy as np
import torch
import torch.nn as nn
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
torch.cuda.empty_cache()
# 设置设备
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
print("Using device:", device)
```

```
# 超参数
torch.manual_seed(42)
SEQ_LEN = 8  # 使用过去 8 小时
PRED_HORIZON = 4  # 预测 4 小时后
BATCH_SIZE = 64
EPOCHS = 500
LR = 0.001
# 1. 加载数据
df = pd.read_excel("../地点 1 训练数据.xlsx", parse_dates=["采集时刻"])
df.sort_values("采集时刻", inplace=True)
# 2. 选取特征列
features = ['热泵功率(kw)', '供温(℃)', '回温(℃)', '补水流速(m3h)', '设定温度
(℃)','hour','weekday','供回温差','热泵是否开启','室内温度变化', '环境温度(℃)',
'室内温度(℃)']
data = df[features].values
input_size = len(features)
# 3. 归一化
scaler = MinMaxScaler()
data_scaled = scaler.fit_transform(data)
# 4. 构造样本数据
X, y = [], []
for i in range(len(data_scaled) - SEQ_LEN - PRED_HORIZON):
    x_seq = data_scaled[i:i + SEQ_LEN]
    y_target = data_scaled[i + SEQ_LEN + PRED_HORIZON - 1][-1]  # 室内温度在最后
一列
    X.append(x_seq)
    y.append(y_target)
X = np.array(X)  # shape: [N, SEQ_LEN, 7]
y = np.array(y)  # shape: [N,]
# 5. 划分训练/测试集
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
shuffle=False)
X_train = torch.tensor(X_train, dtype=torch.float32).to(device)
y_train = torch.tensor(y_train, dtype=torch.float32).unsqueeze(-1).to(device)
X_test = torch.tensor(X_test, dtype=torch.float32).to(device)
y_test = torch.tensor(y_test, dtype=torch.float32).unsqueeze(-1).to(device)
# 6. 模型定义
class LSTMModel(nn.Module):
    def __init__(self, input_size=input_size, hidden_size=64, num_layers=4):
        super().__init__()
        self.lstm = nn.LSTM(input_size, hidden_size, num_layers,dropout=0.2,
batch_first=True)
        self.fc = nn.Linear(hidden_size, 1)
    def forward(self, x):
```

```
        out, _ = self.lstm(x)
        return self.fc(out[:, -1, :])
model = LSTMModel().to(device)
criterion = nn.MSELoss()
optimizer = torch.optim.Adam(model.parameters(), lr=LR)
# 7. 模型训练
for epoch in range(EPOCHS):
    model.train()
    output = model(X_train)
    loss = criterion(output, y_train)
    optimizer.zero_grad()
    loss.backward()
    optimizer.step()
    if (epoch + 1) % 10 == 0:
        print(f"Epoch {epoch + 1}/{EPOCHS}, Loss: {loss.item():.6f}")
# 8. 模型评估
model.eval()
with torch.no_grad():
    y_pred = model(X_test).cpu().numpy()
    y_true = y_test.cpu().numpy()
# 9. 反归一化（只对室内温度一列）
indoor_temp_index = features.index("室内温度(℃)")
min_temp = scaler.data_min_[indoor_temp_index]
max_temp = scaler.data_max_[indoor_temp_index]
y_pred_rescaled = y_pred * (max_temp - min_temp) + min_temp
y_true_rescaled = y_true * (max_temp - min_temp) + min_temp
# 10. 可视化
# 获取对应的预测目标时间（采集时刻 + 4 小时）
timestamps = df['采集时刻'].values
# y 是从第 SEQ_LEN + PRED_HORIZON 开始生成的
start_idx_in_timestamps = SEQ_LEN + PRED_HORIZON
# y_test 是 y 的后 20%，找到它在 y 中的起始位置
test_start_idx_in_y = int(len(y) * 0.8)
# 对应到原始 timestamps 的起始点
test_start_time_idx = start_idx_in_timestamps + test_start_idx_in_y
# 截取对应的时间戳作为横坐标
target_times = timestamps[test_start_time_idx : test_start_time_idx +
len(y_test)]

# 可视化
plt.figure(figsize=(12, 5))
plt.plot(target_times, y_true_rescaled, label="exact")
plt.plot(target_times, y_pred_rescaled, label="prediction")
plt.xlabel("date")
```

```
plt.ylabel("indoor temperature")
plt.legend()
plt.tight_layout()
plt.savefig("../问题 3/地点 1LSTM 拟合结果.pdf",format="pdf")
```

地点 2 LSTM 训练

```
import pandas as pd
import numpy as np
import torch
import torch.nn as nn
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
# 设置设备
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
print("Using device:", device)
torch.cuda.empty_cache()
# 超参数
torch.manual_seed(42)
SEQ_LEN = 8  # 使用过去 8 小时
PRED_HORIZON = 4  # 预测 4 小时后
BATCH_SIZE = 64
EPOCHS = 500
LR = 0.001
# 1. 加载数据
df = pd.read_excel("../地点 2 训练数据.xlsx", parse_dates=["采集时刻"])
df.sort_values("采集时刻", inplace=True)
# 2. 选取特征列
features = ['热泵功率(kw)', '供温(℃)', '回温(℃)', '补水流速(m3h)', '设定温度
(℃)','hour','weekday','供回温差','热泵是否开启','室内温度变化', '环境温度(℃)',
'室内温度(℃)']
data = df[features].values
input_size = len(features)
# 3. 归一化
scaler = MinMaxScaler()
data_scaled = scaler.fit_transform(data)
# 4. 构造样本数据
X, y = [], []
for i in range(len(data_scaled) - SEQ_LEN - PRED_HORIZON):
    x_seq = data_scaled[i:i + SEQ_LEN]
    y_target = data_scaled[i + SEQ_LEN + PRED_HORIZON - 1][-1]  # 室内温度在最后
一列
    X.append(x_seq)
    y.append(y_target)
```

```
X = np.array(X)  # shape: [N, SEQ_LEN, 7]
y = np.array(y)  # shape: [N,]
# 5. 划分训练/测试集
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
shuffle=False)
X_train = torch.tensor(X_train, dtype=torch.float32).to(device)
y_train = torch.tensor(y_train, dtype=torch.float32).unsqueeze(-1).to(device)
X_test = torch.tensor(X_test, dtype=torch.float32).to(device)
y_test = torch.tensor(y_test, dtype=torch.float32).unsqueeze(-1).to(device)
# 6. 模型定义
class LSTMModel(nn.Module):
    def __init__(self, input_size=input_size, hidden_size=64, num_layers=3):
        super().__init__()
        self.lstm = nn.LSTM(input_size, hidden_size, num_layers,dropout=0.2,
batch_first=True,bidirectional=True)
        self.fc = nn.Linear(hidden_size*2, 1)
    def forward(self, x):
        out, _ = self.lstm(x)
        return self.fc(out[:, -1, :])
model = LSTMModel().to(device)
criterion = nn.MSELoss()
optimizer = torch.optim.Adam(model.parameters(), lr=LR)
# 7. 模型训练
for epoch in range(EPOCHS):
    model.train()
    output = model(X_train)
    loss = criterion(output, y_train)
    optimizer.zero_grad()
    loss.backward()
    optimizer.step()
    if (epoch + 1) % 10 == 0:
        print(f"Epoch {epoch + 1}/{EPOCHS}, Loss: {loss.item():.6f}")
# 8. 模型评估
model.eval()
with torch.no_grad():
    y_pred = model(X_test).cpu().numpy()
    y_true = y_test.cpu().numpy()
# 9. 反归一化（只对室内温度一列）
indoor_temp_index = features.index("室内温度(℃)")
min_temp = scaler.data_min_[indoor_temp_index]
max_temp = scaler.data_max_[indoor_temp_index]
y_pred_rescaled = y_pred * (max_temp - min_temp) + min_temp
y_true_rescaled = y_true * (max_temp - min_temp) + min_temp
# 10. 可视化
```

```
# 获取对应的预测目标时间（采集时刻 + 4 小时）
timestamps = df['采集时刻'].values
# y 是从第 SEQ_LEN + PRED_HORIZON 开始生成的
start_idx_in_timestamps = SEQ_LEN + PRED_HORIZON
# y_test 是 y 的后 20%，找到它在 y 中的起始位置
test_start_idx_in_y = int(len(y) * 0.8)
# 对应到原始 timestamps 的起始点
test_start_time_idx = start_idx_in_timestamps + test_start_idx_in_y
# 截取对应的时间戳作为横坐标
target_times = timestamps[test_start_time_idx : test_start_time_idx +
len(y_test)]
# 可视化
plt.figure(figsize=(12, 5))
plt.plot(target_times, y_true_rescaled, label="exact")
plt.plot(target_times, y_pred_rescaled, label="prediction")
plt.xlabel("date")
plt.ylabel("indoor temperature")
plt.legend()
plt.tight_layout()
plt.savefig("../问题 3/地点 2LSTM 拟合结果.pdf",format="pdf")
plt.show()
```

地点 1 LSTM(注意力机制)训练

```
import pandas as pd
import numpy as np
import torch
import torch.nn as nn
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
torch.cuda.empty_cache()
# 设置设备
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
print("Using device:", device)
# 超参数
torch.manual_seed(42)
SEQ_LEN = 8  # 使用过去 8 小时
PRED_HORIZON = 4  # 预测 4 小时后
BATCH_SIZE = 64
EPOCHS = 500
LR = 0.001
# 1. 加载数据
```

```
df = pd.read_excel("../地点 1 训练数据.xlsx", parse_dates=["采集时刻"])
df.sort_values("采集时刻", inplace=True)
# 2. 选取特征列
features = ['热泵功率(kw)', '供温(℃)', '回温(℃)', '补水流速(m3h)', '设定温度
(℃)','hour','weekday','供回温差','热泵是否开启','室内温度变化', '环境温度(℃)',
'室内温度(℃)']
data = df[features].values
input_size = len(features)
# 3. 归一化
scaler = MinMaxScaler()
data_scaled = scaler.fit_transform(data)
# 4. 构造样本数据
X, y = [], []
for i in range(len(data_scaled) - SEQ_LEN - PRED_HORIZON):
    x_seq = data_scaled[i:i + SEQ_LEN]
    y_target = data_scaled[i + SEQ_LEN + PRED_HORIZON - 1][-1]  # 室内温度在最后
一列
    X.append(x_seq)
    y.append(y_target)
X = np.array(X)  # shape: [N, SEQ_LEN, 7]
y = np.array(y)  # shape: [N,]
# 5. 划分训练/测试集
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
shuffle=False)
X_train = torch.tensor(X_train, dtype=torch.float32).to(device)
y_train = torch.tensor(y_train, dtype=torch.float32).unsqueeze(-1).to(device)
X_test = torch.tensor(X_test, dtype=torch.float32).to(device)
y_test = torch.tensor(y_test, dtype=torch.float32).unsqueeze(-1).to(device)
# 6. 模型定义
class LSTMWithAttention(nn.Module):
    def __init__(self, input_size=input_size, hidden_size=64, num_layers=2,
bidirectional=True):
        super().__init__()
        self.hidden_size = hidden_size
        self.bidirectional = bidirectional
        self.lstm = nn.LSTM(
            input_size,
            hidden_size,
            num_layers,
            dropout=0.2,
            batch_first=True,
            bidirectional=bidirectional
        )
        lstm_output_size = hidden_size * 2 if bidirectional else hidden_size
```

```
        self.attn = nn.Linear(lstm_output_size, 1)  # Attention 打分层
        self.fc = nn.Linear(lstm_output_size, 1)    # 输出层

    def forward(self, x):
        lstm_out, _ = self.lstm(x)  # [B, T, H]
        weights = torch.softmax(self.attn(lstm_out), dim=1)  # [B, T, 1]
        context = torch.sum(weights * lstm_out, dim=1)       # [B, H]
        out = self.fc(context)                               # [B, 1]
        return out
model = LSTMWithAttention().to(device)
criterion = nn.MSELoss()
optimizer = torch.optim.Adam(model.parameters(), lr=LR)
# 7. 模型训练
for epoch in range(EPOCHS):
    model.train()
    output = model(X_train)
    loss = criterion(output, y_train)
    optimizer.zero_grad()
    loss.backward()
    optimizer.step()
    if (epoch + 1) % 10 == 0:
        print(f"Epoch {epoch + 1}/{EPOCHS}, Loss: {loss.item():.6f}")
# 8. 模型评估
model.eval()
with torch.no_grad():
    y_pred = model(X_test).cpu().numpy()
    y_true = y_test.cpu().numpy()
# 9. 反归一化（只对室内温度一列）
indoor_temp_index = features.index("室内温度(℃)")
min_temp = scaler.data_min_[indoor_temp_index]
max_temp = scaler.data_max_[indoor_temp_index]
y_pred_rescaled = y_pred * (max_temp - min_temp) + min_temp
y_true_rescaled = y_true * (max_temp - min_temp) + min_temp
# 10. 可视化
# 获取对应的预测目标时间（采集时刻 + 4 小时）
timestamps = df['采集时刻'].values
# y 是从第 SEQ_LEN + PRED_HORIZON 开始生成的
start_idx_in_timestamps = SEQ_LEN + PRED_HORIZON
# y_test 是 y 的后 20%，找到它在 y 中的起始位置
test_start_idx_in_y = int(len(y) * 0.8)
# 对应到原始 timestamps 的起始点
test_start_time_idx = start_idx_in_timestamps + test_start_idx_in_y
# 截取对应的时间戳作为横坐标
```

```
target_times = timestamps[test_start_time_idx : test_start_time_idx +
len(y_test)]

# 训练结果可视化
plt.figure(figsize=(12, 5))
plt.plot(target_times, y_true_rescaled, label="exact")
plt.plot(target_times, y_pred_rescaled, label="prediction")
plt.xlabel("date")
plt.ylabel("indoor temperature")
plt.legend()
plt.tight_layout()
plt.savefig("../问题 3/地点 1LSTM 拟合结果(增加注意力机制).pdf",format="pdf")
```

地点 2 LSTM(注意力机制)训练

```
import pandas as pd
import numpy as np
import torch
import torch.nn as nn
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
# 设置设备
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
print("Using device:", device)
torch.cuda.empty_cache()
# 超参数
torch.manual_seed(42)
SEQ_LEN = 8  # 使用过去 8 小时
PRED_HORIZON = 4  # 预测 4 小时后
BATCH_SIZE = 64
EPOCHS = 700
LR = 0.001
# 1. 加载数据
df = pd.read_excel("../地点 2 训练数据.xlsx", parse_dates=["采集时刻"])
df.sort_values("采集时刻", inplace=True)
# 2. 选取特征列
features = ['热泵功率(kw)', '供温(℃)', '回温(℃)', '补水流速(m3h)', '设定温度
(℃)','hour','weekday','供回温差','热泵是否开启','室内温度变化', '环境温度(℃)',
'室内温度(℃)']
data = df[features].values
input_size = len(features)
# 3. 归一化
scaler = MinMaxScaler()
data_scaled = scaler.fit_transform(data)
```

```
# 4. 构造样本数据
X, y = [], []
for i in range(len(data_scaled) - SEQ_LEN - PRED_HORIZON):
    x_seq = data_scaled[i:i + SEQ_LEN]
    y_target = data_scaled[i + SEQ_LEN + PRED_HORIZON - 1][-1]  # 室内温度在最后一列
    X.append(x_seq)
    y.append(y_target)
X = np.array(X)  # shape: [N, SEQ_LEN, 7]
y = np.array(y)  # shape: [N,]
# 5. 划分训练/测试集
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, shuffle=False)
X_train = torch.tensor(X_train, dtype=torch.float32).to(device)
y_train = torch.tensor(y_train, dtype=torch.float32).unsqueeze(-1).to(device)
X_test = torch.tensor(X_test, dtype=torch.float32).to(device)
y_test = torch.tensor(y_test, dtype=torch.float32).unsqueeze(-1).to(device)
# 6. 模型定义
class LSTMWithAttention(nn.Module):
    def __init__(self, input_size=input_size, hidden_size=64, num_layers=2, bidirectional=True):
        super().__init__()
        self.hidden_size = hidden_size
        self.bidirectional = bidirectional
        self.lstm = nn.LSTM(
            input_size,
            hidden_size,
            num_layers,
            dropout=0.2,
            batch_first=True,
            bidirectional=bidirectional
        )
        lstm_output_size = hidden_size * 2 if bidirectional else hidden_size
        self.attn = nn.Linear(lstm_output_size, 1)  # Attention 打分层
        self.fc = nn.Linear(lstm_output_size, 1)    # 输出层
    def forward(self, x):
        lstm_out, _ = self.lstm(x)  # [B, T, H]
        weights = torch.softmax(self.attn(lstm_out), dim=1)  # [B, T, 1]
        context = torch.sum(weights * lstm_out, dim=1)       # [B, H]
        out = self.fc(context)                               # [B, 1]
        return out
model = LSTMWithAttention().to(device)
criterion = nn.MSELoss()
optimizer = torch.optim.Adam(model.parameters(), lr=LR)
```

```
# 7. 模型训练
for epoch in range(EPOCHS):
    model.train()
    output = model(X_train)
    loss = criterion(output, y_train)
    optimizer.zero_grad()
    loss.backward()
    optimizer.step()
    if (epoch + 1) % 10 == 0:
        print(f"Epoch {epoch + 1}/{EPOCHS}, Loss: {loss.item():.6f}")
# 8. 模型评估
model.eval()
with torch.no_grad():
    y_pred = model(X_test).cpu().numpy()
    y_true = y_test.cpu().numpy()
# 9. 反归一化（只对室内温度一列）
indoor_temp_index = features.index("室内温度(℃)")
min_temp = scaler.data_min_[indoor_temp_index]
max_temp = scaler.data_max_[indoor_temp_index]
y_pred_rescaled = y_pred * (max_temp - min_temp) + min_temp
y_true_rescaled = y_true * (max_temp - min_temp) + min_temp
# 10. 可视化
# 获取对应的预测目标时间（采集时刻 + 4 小时）
timestamps = df['采集时刻'].values
# y 是从第 SEQ_LEN + PRED_HORIZON 开始生成的
start_idx_in_timestamps = SEQ_LEN + PRED_HORIZON
# y_test 是 y 的后 20%，找到它在 y 中的起始位置
test_start_idx_in_y = int(len(y) * 0.8)
# 对应到原始 timestamps 的起始点
test_start_time_idx = start_idx_in_timestamps + test_start_idx_in_y
# 截取对应的时间戳作为横坐标
target_times = timestamps[test_start_time_idx : test_start_time_idx +
len(y_test)]

# 训练结果可视化
plt.figure(figsize=(12, 5))
plt.plot(target_times, y_true_rescaled, label="exact")
plt.plot(target_times, y_pred_rescaled, label="prediction")
plt.xlabel("date")
plt.ylabel("indoor temperature")
plt.legend()
plt.tight_layout()
plt.savefig("../问题 3/地点 2LSTM 拟合结果(增加注意力机制).pdf",format="pdf")
```