对晶硅片产销策略优化问题的研究

摘要

高纯度晶硅片企业之间竞争激烈，为了具备市场竞争力，晶硅片企业需要在成本控制、利润优化和供需管理之间取得平衡。本文旨在建立合适的数学模型，帮助企业优化经营决策、提高所得利润。

对于问题一，首先要厘清各决策因子与利润之间的关系，针对四型晶硅片产销数据，综合考虑销售收入 Revenue、固定成本 FC，仔细考量变动成本 VC 及硅泥为其冲减的生产变动成本 R，最终求得其利润为 $L = \Sigma_{k=1}^{4} Q_k(P_k - VC_k) - F - L_c - E + R$。该**月利润计算模型**通过参数化设计支持动态情景模拟，验证了销量与售价对利润的调控效应，为后续优化提供目标函数基础。

对于问题二，针对附件中提供的数据的小样本特性，首先分别借**灰色预测模型**和**指数平滑模型**，通过 1-7 月的数据来预测 8 月数据，并与真实值相比较，在综合考虑两种模型的**平均绝对百分比误差（MAPE）**后，可以得出指数平滑模型的拟合效果更好的结论。故而继续使用指数平滑模型来预测 9 月份各因子的取值。需要注意指数平滑法分为三种，**单指数平滑法**适用于无趋势无季节性的时间序列，**双指数平滑法**适用于有趋势无季节性的时间序列，三**指数平滑法**适用于有趋势有季节性的时间序列，在预测销量时利用单指数平滑法，在预测单晶硅片售价、硅料单价、槽距、成品率时利用双指数平滑法，在预测电费时利用三指数平滑法。最后再使用**置信区间估计法**给出预测的合理区间而不是简单的点估计。确定各因子的取值后最终可得到 9 月企业总利润的预测值约为 **997.6 万元**。

对于问题三, 为了最大化利润，需要建立一个辅助决策优化企业利润的数学模型，经过分析可以确定**核心决策变量**为四种型号产品的销售价格，销量也可通过需求函数与其建立线性负相关关系。基于此，采取**模拟退火算法**寻求最大利润情况下售价与销量的取值，由于供需平衡的假设，已知销量即已知产量，故而可以根据最优解，给出 9 月份的生产计划与销售预案。

对于问题四, 需要构建 **LLM**-数学模型动态耦合框架来解决外部因素量化难题。利用开源 LLM（如 DeepSeek）整合企业内部产销数据和外部非结构化信息，进行动态参数智能校准的信息处理流程。涉及到了数据准备、数据清洗、结果评估和大模型融入等多个方面预测与优化，实现兼顾政策适配性与经济效益的产销策略。优化后 9 月预期利润达到 **3048.05 万元**。

最后，本文针对使用的数学模型进行了**残差分析**和**灵敏度分析**，并客观分析了各模型的优点和不足。

关键字：指数平滑法　模拟退火算法　置信区间估计　LLM　灰色预测

目录

一、 问题重述

1.1 问题背景

近年来，全球清洁能源需求持续攀升，高纯度晶硅片作为高效太阳能电池的核心材料，其市场重要性日益凸显。在需求端，光伏产业对高纯度、稳定性强的高性能晶硅片的依赖度显著增加，倒逼生产商加速技术迭代与品质升级；在供给端，技术进步推动晶硅片向超薄化、高纯度方向突破，但由此引发的产品同质化竞争加剧了行业洗牌，迫使企业通过工艺创新与精益生产维持竞争优势。但高纯度晶硅片的生产成本通常较高，这就要求企业不断优化生产流程、降低成本，以保持利润空间。为此，企业不仅要依赖于技术创新，还需要在原材料采购、能源消耗等方面采取更加精细的管理和策略。

图 1　晶硅片生产车间

1.2 问题要求

为了优化晶硅片企业的经营决策，提高企业利润。本题要求结合某企业 2024 年 1 至 8 月的相关数据，从四种型号晶硅片的销量、售价与单晶方棒进价等重要决策变量入手，建立合适的数学模型，解决以下问题：

问题 1：重点考虑四型硅片的销量、售价、单晶方棒以及影响企业利润的其他重要决策因子，建立便于企业进行决策分析的月利润计算模型。

问题 2：建立数学模型预测企业四型硅片的月销量、售价、单晶方棒价格以及其他重要因子取值的波动趋势，推测因子合理变化区间；再利用所建立的模型预测 9 月份各因子的波动趋势和变化区间。

问题 3：一般来说，产品的售价越低，销量就越大；产量越大，企业固定成本平均到单位产品上的份额就越低，变动成本的波动也会影响企业利润。众多因子之间的相互制约与冲突让企业很难决策下个月的生产计划与销售策略。为辅助决策、优化企业利润，尝试建立合适的数学模型，并给出 9 月份的生产计划与销售预案。

问题 4：销量、售价以及原材料、重要耗材的价格等因子会受到政策与市场等综合因素的影响，所以单独根据企业内部数据的建模难以得出上佳的产销策略。为了为企业提供更好的决策支持，在前述数学模型的基础上，借助大语言模型的综合分析能力，建立一个综合考虑多方面因素的智能模型，从数据准备、数据清洗、结果评估和大模型融入等多方面，给出利用开源大模型进行预测与优化的详细路径，并论述或者实测路径的可行性。进而设计一种大模型与前述数学模型相结合的方案，服务于该企业的产销决策。

二、 问题分析

2.1 问题一分析

问题一要求建立一个便于企业进行决策分析的月利润计算模型，该模型需要重点考虑四种硅片的销量、售价、耗材价格、生产成本、人工成本、三费等影响因子。模型的核心在于以利润贡献为基础，综合考虑包括耗材、硅棒单耗等在内的完整成本项，形成一套参数明确的决策支持模型。

2.2 问题二分析

对于问题二，我们需要在时间维度上对几个关键因子进行建模并做出进一步的预测，由于其仅通过前八个月的数据进行预测，数据集很小，故常规的时间序列模型（如ARIMA）以及神经网络模型可能无法胜任预测工作。对于此问，我们比较了适用于小样本量的灰色预测法与指数平滑法的预测结果，选择了结果较好的指数平滑法。我们首先通过 1-7 月的数据预测 8 月数据，并与真实值相比较，验证了指数平滑法的预测效果，然后对 9 月各因子的值进行了预测，并采用置信区间估计的方法给出预测区间。将预测值代入第一问的模型中，就可以得到 9 月份该工厂利润的预测值。

2.3 问题三分析

问题三要求建立优化企业利润的数学模型，并制定 9 月份的生产计划与销售策略。我们需要结合需求函数估计的方法量化价格与销量的关系，之后采用模拟退火算法，设置算法相关参数（如初始猜测值、变量边界、固定成本、单位变动成本、产能限制等），定义目标函数为最大化利润，并设置非线性约束条件。通过优化售价和计划产量，输出 9 月份的最优生产计划，展示预期利润、固定成本、总变动成本和硅泥回收价值，并通过敏感性分析验证方案的合理性和可行性。

2.4 问题四分析

解决问题四的核心在于构建一个动态耦合大语言模型（LLM）与数学模型的智能决策系统：首先利用开源 LLM（如 DeepSeek）整合企业内部产销数据和外部非结构化信息，进行动态参数智能校准的信息处理流程，解析文本中的关键信号，将非结构化描述转化为结构化修正系数，再动态嵌入数学模型中，使目标函数随外部环境自适应调整，输出结构化数据表达，再将输出数据输入 LLM 进行增强分析，从数据中自动提取见解并以人类可读的文本形式呈现，输出解释性文本和策略建议，即兼顾政策适配性与经济效益的产销策略。

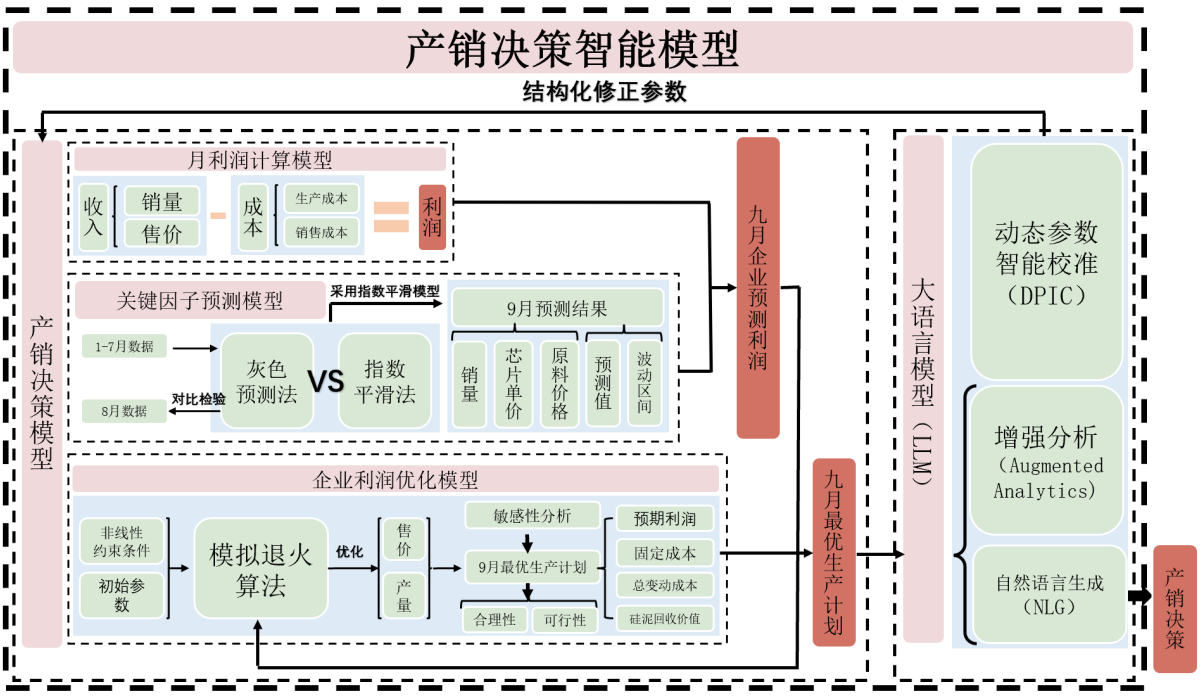


图 2　我们的工作

三、 模型假设

为简化问题，本文做出以下假设：

1. **假设市场供需平衡：**在研究的时间范围内，市场需求与企业的生产能力能够基本匹配，不存在严重的供过于求或供不应求的情况，从而保证企业的销售量等于生产量。
2. **假设成本线性关系：**生产成本与生产量呈线性关系，即单位产品的变动成本在不同生产规模下保持不变，固定成本在一定生产规模内保持稳定。
3. **假设历史数据代表性：**2024 年 1 至 8 月的历史数据具有代表性，能够反映四型硅片销量、售价、单晶方棒价格等因子的长期趋势和周期性变化规律。
4. **假设利润最大化目标：**企业的主要目标是最大化月利润，不考虑其他非财务目标（如

市场份额、品牌建设等）对决策的影响，即所有决策均以利润最大化为导向。

5. **假设模型互补性：**大语言模型与数学模型在决策支持方面具有互补性，即大语言模型能够弥补数学模型在处理非结构化数据和复杂市场因素方面的不足，而数学模型能够为大语言模型提供定量分析和优化决策的基础框架。

四、 符号说明

符号	说明	单位
Q_k	第 k 型晶硅片的销量	百片
P_k	第 k 型晶硅片的售价	元/百片
c_s	单晶硅片单价	元/kg
S_k	第 k 型晶硅片硅棒单耗	kg/百片
e_k	单位产量电耗	kWh/百片
P_e	电价	元/kWh
p_i	第 i 类耗材单价	元/单位
F	月度生产公用成本	元
L_c	固定人工成本	元
E	三费	元
R	硅泥销售收益	元
n	计入计算的耗材种类数	元
$b_{i,k}$	第 i 类耗材在第 k 型晶硅片生产中每百片的消耗量	单位/百片
L_{fk}	第 k 型晶硅片浮动人工成本	元

五、 问题一: 月利润计算模型

5.1 模型建立

问题一要求建立一个企业月度利润计算模型，用于利润核算与经营评估。建立模型将四种型号晶硅片的销量和售价作为核心输入变量，综合考虑硅棒单耗成本、生产耗材、电力费用、人工成本、生产公用成本以及销售、管理和财务费用等经营成本，以准确模拟每月的利润计算过程，形式化利润函数以支持量化分析和策略评估，并为后续优化决策提供目标函数依据。

5.2 模型求解

Step1：销售收入

企业总销售收入由四型硅片销售额共同组成：

$$Revenue = \Sigma_{k=1}^{4} Q_k P_k \tag{1}$$

Step2：固定成本

将各个固定成本成本直接相加得：

$$FC = F + L_c + E \tag{2}$$

Step3：变动成本

第 k 型晶硅片的单位变动成本包括四部分：

(1) 硅棒单耗成本：

$$VC_k^{(1)} = c_s S_k \tag{3}$$

(2) 电耗成本：

$$VC_k^{(2)} = e_k P_e \tag{4}$$

(3) 耗材成本：

$$VC_k^{(3)} = \Sigma_{k=1}^{n} b_{i,k} p_i \tag{5}$$

(4) 浮动人工成本：

$$VC_k^{(4)} = L_{fk} \tag{6}$$

故第 k 型硅片的单位变动成本为

$$VC_k = c_s S_k + e_k P_e + \Sigma_{k=1}^{n} b_{i,k} p_i + L_{fk} \tag{7}$$

第 k 型硅片的变动成本为

$$Q_k VC_k = (c_s S_k + e_k P_e + \Sigma_{k=1}^{n} b_{i,k} p_i + L_{fk}) Q_k \tag{8}$$

四型硅片总的变动成本为

$$VC = \Sigma_{k=1}^{4} Q_k VC_k \tag{9}$$

因硅泥伴随生产而产生，并可及时外销，可冲减生产变动成本，故：

$$VC_{final} = \Sigma_{k=1}^{4} Q_k VC_k - R \tag{10}$$

Step4：得出月利润计算模型

企业的销售收入减去成本可以得到最终的利润函数为：

$$L = Revenue - FC - VC_{final} \tag{11}$$

即

$$L = \Sigma_{k=1}^{4} Q_K P_K - F - L_c - E - \Sigma_{k=1}^{4} Q_k VC_k + R \tag{12}$$

进一步整理得

$$L = \Sigma_{k=1}^{4} Q_k (P_k - VC_k) - F - L_c - E + R \tag{13}$$

在该函数中，VC_k 可以通过相关的生产参数在前期计算得出。可以看出 L 受 Q_k，P_k 调控。

六、 问题二：关键因子预测模型

6.1 模型建立

指数平滑法（Exponential Smoothing）是一种基于**时间序列**数据的预测方法，尤其适用于**具有较强平稳性特征的中短期预测场景**，它利用指数函数对过去的观测值进行加权，以产生对未来的预测（其中时间较近的观察值拥有较大的权重，而较远的观察值权重较小)。灰色预测模型 (Gray Forecast Model) 则是通过少量的、不完全的的信息，建立数学模型并做出预测的一种预测方法。二者均适用于本题短周期、少样本量的条件。

在对比灰色预测模型与指数平滑法后，我们发现指数平滑法在预测准确性上表现更优。考虑到销量、电价、成品率、硅料单价、售价、槽距等数据的稳定性和较小的波动，我们选择建立指数平滑模型。

指数平滑法主要有三种常见的类型：单指数平滑法、双指数平滑法和三指数平滑法，分别适用于无趋势无季节性、有趋势无季节性、有趋势有季节性的时间序列。考虑到外部因素的扰动，简单的点估计可能无法准确预测数据，因此我们还将提供置信区间以期达到更准确的预测结果。

6.1.1 灰色预测

GM(1,1) 模型是灰色预测中最常用的模型，其基本形式为：

$$\frac{dx^{(1)}}{dt} + ax^{(1)} = b \tag{14}$$

其中，$x^{(1)}$ 是累加生成序列，a 和 b 是待估计的参数。

通过求解上述微分方程，可以得到灰色预测模型的解：

$$x^{(1)}(t) = (x^{(0)}(1) - \frac{b}{a})e^{-at} + \frac{b}{a} \tag{15}$$

还原一次数据序列，得到预测值：

$$x^{(0)}(t) = x^{(1)}(t) - x^{(1)}(t-1) \tag{16}$$

灰色预测方法具有简单、实用、预测精度高等特点，适用于各种不确定性和不完全信息的预测问题。

6.1.2 指数平滑法

在预测**销量**这种随时间变化的**趋势不明显**的变量时，我们考虑**单指数平滑公式**。对于时间序列数据 $\{y_t\}, t = 1, 2, ..., n$，预测公式为：

$$\begin{cases} \text{预测方程：} & \hat{y}_{t+1|t} = \alpha y_t + (1-\alpha)\hat{y}_{t|t-1} \\ \text{初始化：} & \hat{y}_{1|0} = y_1 \end{cases}$$

其中：

- $\hat{y}_{t+1|t}$: 第 t 月对 $t+1$ 月的预测值
- $\alpha \in [0, 1]$: 平滑系数
- y_t: 第 t 月的实际值

在预测**单晶硅片售价，硅料单价，槽距，成品率**这些**有明显趋势性**的变量时，可以使用**双指数平滑公式**。Holt 双指数平滑法包含水平分量和趋势分量：

$$\begin{cases} \text{水平方程：} & S_t = \alpha y_t + (1-\alpha)(S_{t-1} + T_{t-1}) \\ \text{趋势方程：} & T_t = \beta(S_t - S_{t-1}) + (1-\beta)T_{t-1} \\ \text{预测方程：} & \hat{y}_{t+h} = S_t + h \cdot T_t \end{cases}$$

其中：

- S_t：第 t 月的平滑水平分量
- T_t：第 t 月的平滑趋势分量
- α：水平平滑系数
- β：趋势平滑系数

电费的波动主要是由于不同季节用电量变化产生的，因此，在预测电费变化趋势时，我们使用可以考虑**季节性的三指数平滑模型**：

$$\begin{cases} \text{水平方程：} & l_t = \alpha(y_t - s_{t-m}) + (1-\alpha)(l_{t-1} + b_{t-1}) \\ \text{趋势方程：} & b_t = \beta(l_t - l_{t-1}) + (1-\beta)b_{t-1} \\ \text{季节方程：} & s_t = \gamma(y_t - l_t) + (1-\gamma)s_{t-m} \\ \text{预测方程：} & \hat{y}_{t+h} = l_t + h \cdot b_t + s_{t+h-m} \end{cases}$$

其中：

- l_t：第 t 月的水平分量

- b_t：第 t 月的趋势分量
- s_t：第 t 月的季节分量
- $m = 12$：年度周期长度

6.1.3 置信区间估计

假设预测误差满足：

$$\epsilon_{t+h} = y_{t+h} - \hat{y}_{t+h} \sim \mathcal{N}(0, \sigma_h^2) \tag{17}$$

其中预测方差：

$$\sigma_h^2 = \sigma^2 \cdot V(h) \tag{18}$$

对于任意指数平滑模型，h 步预测的 95% 置信区间为：

$$\hat{y}_{t+h} \pm 1.96 \cdot \hat{\sigma} \cdot \sqrt{V(h)} \tag{19}$$

各模型方差膨胀因子如下表所示

表 1　各模型方差膨胀因子表达式

模型类型	方差膨胀因子 $V(h)$
单指数平滑	$1 + \alpha^2 \sum_{j=0}^{h-1} (1-\alpha)^{2j}$
双指数平滑	$1 + \sum_{k=1}^{h} [\alpha + \beta k]^2$
三指数平滑	$1 + \sum_{k=1}^{h} \left[\alpha + \beta k + \gamma \delta_{(k \mod m)}\right]^2$

其中季节指示函数：

$$\delta_{(k \mod m)} = \begin{cases} 1, & k \equiv 0 \ (\text{mod}\ m) \\ 0, & \text{其他} \end{cases} \tag{20}$$

标准差可按照如下公式估计：

$$\hat{\sigma} = \sqrt{\frac{1}{n-k} \sum_{t=1}^{n} (y_t - \hat{y}_t)^2} \tag{21}$$

参数个数 k 对应：

- 单指数平滑：$k = 1$
- 双指数平滑：$k = 2$
- 三指数平滑：$k = 3$

6.2 模型求解

Step1：选择合适的模型

首先我们进行两个模型的比较与选择。下文中“预测值 1、误差 1、MAPE1”为灰色预测模型的结果，“预测值 2、误差 2、MAPE2”为指数平滑法的结果。

表 2　销量预测结果　$MAPE1 = 9.27\%$　$MAPE2 = 17.96\%$

芯片种类	预测值 1	预测值 2	误差 1	误差 2	真实值
种类 1	8691508	7442968	1091508	157032	7600000
种类 2	3969780	4225390	-530200	274610	4500000
种类 3	5120778	6897656	-3679221	1902344	8800000
种类 4	7928755	7648828	-321245	691172	8250000

此外，两种模型对电价的预测结果显示 $MAPE1 = 5.07\%, MAPE2 = 0.06\%$；而对单晶硅片销售单价的预测结果显示 $MAPE1 = 6.06\%, MAPE2 = 1.67\%$。可以认为，无论是对于无趋势无季节性、有趋势无季节性、有趋势有季节性的时间序列，指数平滑法的效果都要好于灰色预测法。因此本问题**采用指数平滑模型解决**。下面给出使用这个模型的具体步骤。

Step2：寻找指数平滑模型最优化参数

整理各因子前 7 个月的数据，编程建立对应类型的指数平滑模型并使其参数遍历 [0.1,0.9]，搜索最小化 MSE：

$$\text{MSE} = \frac{1}{n}\sum_{t=1}^{n}(y_t - \hat{y}_t)^2 \tag{22}$$

参数空间：

- $\alpha, \beta, \gamma \in [0.1, 0.9]$，步长 0.1
- 共 9 种组合（单指数）或 81 种组合（双指数）或 729 种组合（三指数）

当 MSE 最小时，参数为最优化参数。

Step3：求预测方程

根据找到的最优化参数写出对应的预测方程，并利用预测方程计算第 8 个月的参数，与其真实值作比较以确定预测方程的拟合效果。

Step4：预测 9 月份各因子参数

根据预测方程进一步 9 月份各因子的波动趋势和变化区间。计算相关误差，误差指

标计算公式如下：

$$\text{MAE} = \frac{1}{n}\sum_{t=1}^{n}|y_t - \hat{y}_t| \tag{23}$$

$$\text{MAPE} = \frac{100\%}{n}\sum_{t=1}^{n}\left|\frac{y_t - \hat{y}_t}{y_t}\right| \tag{24}$$

6.3 求解结果

在本部分，我们首先利用 1-7 月的数据对 8 月的销量、单晶硅片销售单价和原料价格进行了预测，并将预测结果与实际值进行了比较，以验证模型的拟合效果。

表 3　销量　$\alpha = 0.5$　$MAPE = 9.27\%$

芯片种类	预测值	真实值	误差	95% 置信区间
种类 1	7442968	7600000	157032	[5447497.74, 9438438.26]
种类 2	4225390	4500000	274610	[7438871.4, 25611286]
种类 3	6897656	8800000	1902344	[3569698.45, 4881081.55]
种类 4	7648828	8250000	691172	[450053.13, 8125030.61]

表 4　单晶硅片销售单价　$\alpha = 0.2$　$\beta = 0.9$　$MAPE = 1.67\%$

芯片种类	预测值	真实值	误差	95% 置信区间
芯片 1	1.67013566	1.71	-0.03986434	[1.63, 1.71]
芯片 2	1.66015087	1.65	0.010015087	[1.62, 1.70]
芯片 3	1.58123683	1.6	-0.01876317	[1.54, 1.63]
芯片 4	1.31564518	1.35	-0.03435482	[1.27, 1.36]

表 5　原料价格　$\alpha = 0.7$　$\beta = 0.1$　$MAPE = 0.70\%$

芯片种类	预测值	真实值	误差	95% 置信区间
芯片 1	60.28110241	60	0.28110241	[59.32, 61.24]
芯片 2	60.28110241	60	0.28110241	[59.32, 61.24]
芯片 3	60.28110241	60	0.28110241	[59.32, 61.24]
芯片 4	49.30540413	50	-0.69459877	[48.35, 50.26]

基于上述分析，我们可以看到模型在八月份的预测中表现出了较高的准确性。具体来说，销量、单晶硅片销售单价和原料价格的预测误差均在可接受范围内，且平均绝对百分比误差（MAPE）指标显示出模型具有较好的预测性能。这表明所选模型能够**有效地捕捉数据的时间序列特性**，并对短期内的波动进行准确预测。鉴于八月份预测结果的积极反馈，我们采用相同的模型参数和结构来预测九月份的销量、单晶硅片销售单价和原料价格。接下来我们详细展示九月份的预测结果。

表 6　9 月销量预测结果　$\alpha = 0.90$

芯片种类	预测值	95% 置信区间
芯片 1	7599275	[3173646, 12024903]
芯片 2	4496759	[2493233, 6500286]
芯片 3	8754823	[4224420, 13285226]
芯片 4	8237604	[5296819, 11178389]

表 7　9 月单晶硅片销售单价预测结果　$\alpha = 0.20$　$\beta = 0.90$

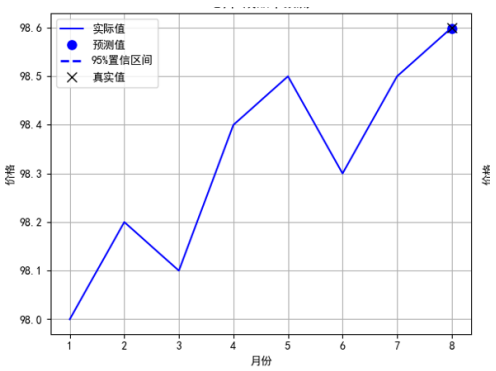
芯片种类	预测值	95% 置信区间
芯片 1	1.66	[1.06, 2.27]
芯片 2	1.62	[1.42, 1.82]
芯片 3	1.45	[1.14, 1.76]
芯片 4	1.21	[0.90, 1.51]

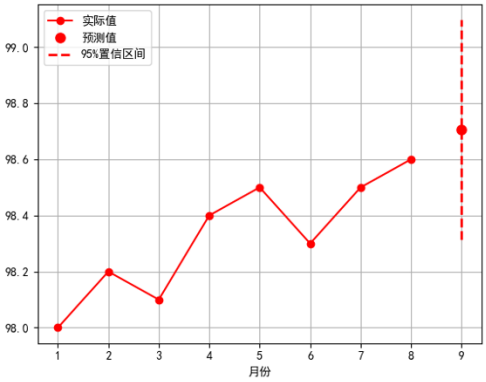
表 8　9 月原料价格预测结果　$\alpha = 0.70$　$\beta = 0.10$

芯片种类	预测值	95% 置信区间
芯片 1	51.66	[42.73, 60.60]
芯片 2	51.66	[42.73, 60.60]
芯片 3	51.66	[42.73, 60.60]
芯片 4	41.77	[31.85, 51.68]

对于在利润计算中起着次要作用的、能观察出随时间有明显变化规律的其它变量，如电价、槽距、优良率等，我们也按其随时间变化的规律选择**对应的指数平滑模型**进行了预测；对于生产公共成本、财务费用等起着次要作用的且不能观察出随时间有明显变

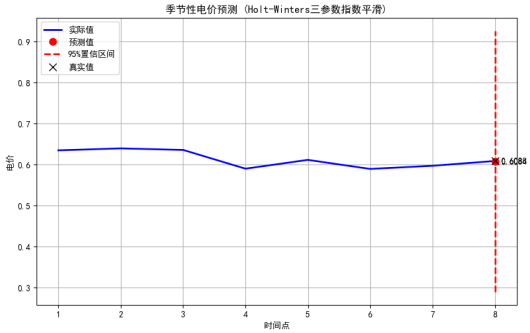
化规律的变量，我们将前 8 个月这个量的**平均值**作为第 9 个月的预测值；对于折旧费用、营业税费等在前 8 个月没有发生变化的变量，我们将其前 8 个月的值作为第 9 个月的预测值。我们将部分变量的预测结果的可视化结果展示在下方：

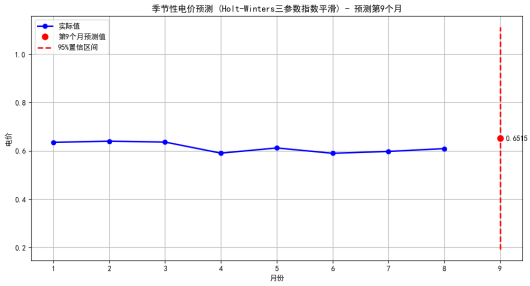




模型检验 **预测结果**

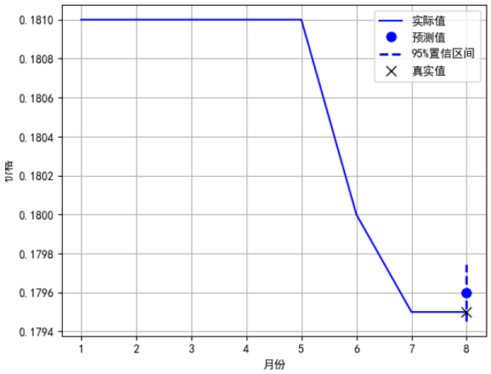
图 3　优良率

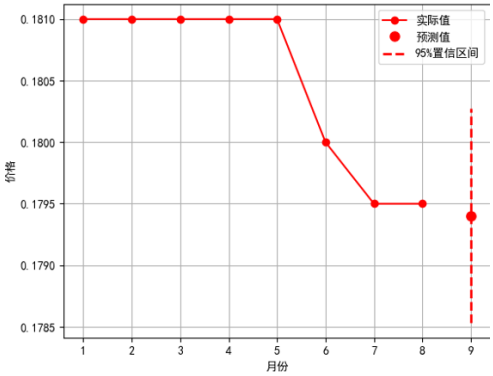




模型检验 **预测结果**

图 4　电价





模型检验 **预测结果**

图 5　槽距

将所有变量的值更新为按上述方法进行预测的值之后，我们将其代入第一问的模型，得到 **9 月企业总利润的预测值为 9976287.99 元**。我们将前八个月的利润值与 9 月的利润值制成了如下的柱状图，可以看出，模型成功地预测了企业利润**逐渐上升**的变化趋势。

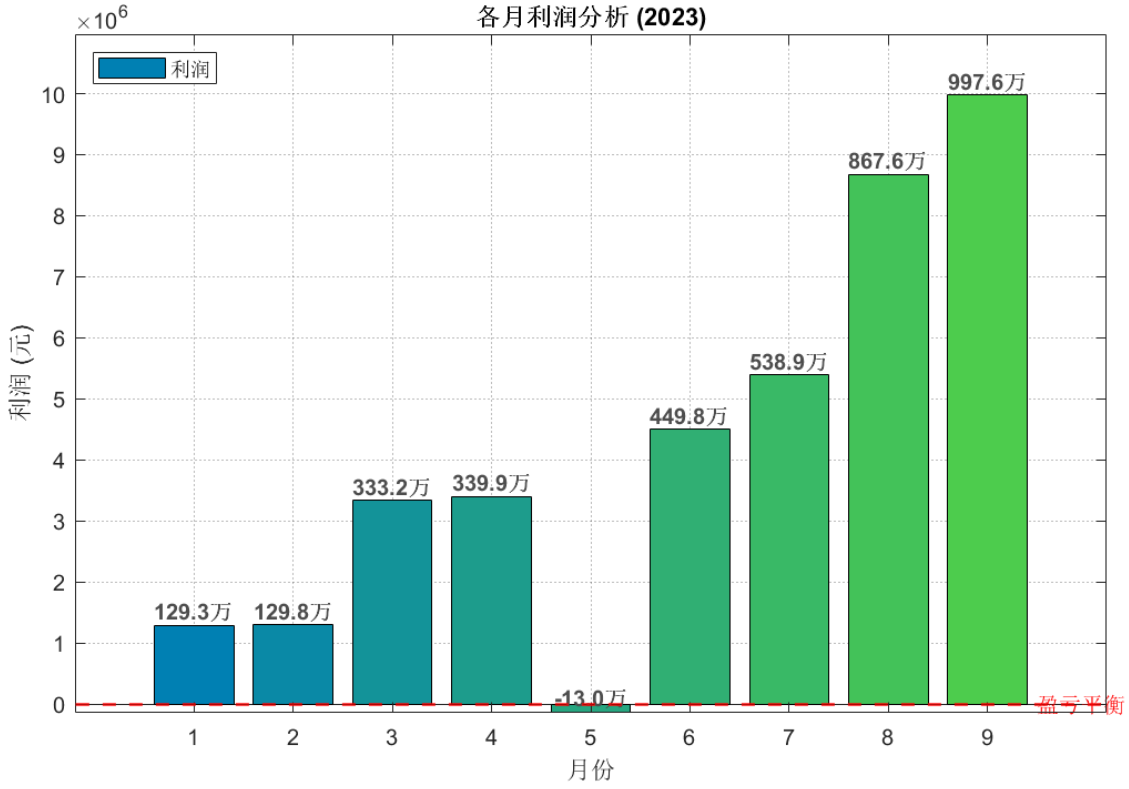


图 6　利润柱状图

七、 问题三：企业利润优化模型

7.1 模型建立

基于以上分析继续建立**企业利润优化模型**。

7.1.1 目标函数

由前述问题可知 $L = \Sigma_{k=1}^{4} Q_k(P_k - VC_k) - FC + R$，其中

$$R = 0.2\Sigma_{k=1}^{4} Q_k \tag{25}$$

0.2 是每片产品的硅泥回收价值 (元/片), 可由附件 2 得出。

模型以四种型号产品的**销售价格作为核心决策变量**：P=[P1,P2,P3,P4], 因为售价直接影响销量（通过**需求函数** $Q_k = a_k - b_k P_k$）。售价与销量的组合决定了总收入和利润，

差异化定价可以协调各型号产品的市场表现，实现整体利润最大化。

7.1.2 约束条件

为了确保解决方案在实际经营中可行，防止算法产生不切实际的价格或产量，反映企业资源限制和市场环境限制，我们制定以下约束条件：

1. **价格上下限约束**：

$$\mathbf{lb} \leq \mathbf{P} \leq \mathbf{ub}$$

其中，下限为

$$\mathbf{lb} = [1.20, 1.20, 1.20, 1.00]$$

表示市场最低价，防止不合理低价竞争。

上限为

$$\mathbf{ub} = [2.50, 2.50, 2.50, 2.00]$$

表示市场最高价，考虑了客户接受度和竞争产品价格。

2. **产能约束**：

$$Q_k \leq \text{capacity\_limit}_k$$

其中，每种型号的产量不超过其产能限制（历史最大产量的 120%）。产能限制基于企业实际生产能力设置。

3. **隐含的非负约束**：

$$Q_k \geq 0$$

$$P_k \geq 0$$

表示销量 Q_k 和价格 P_k 必须大于等于零。

7.1.3 模拟退火算法

接下来利用模拟退火算法来解决问题。模拟退火算法作为一种全局优化算法，借迭代改进思想 [1]，可以在**解空间很大或者不规则**的情况下来寻找某些函数的**全局最优解**，适用于解决如调度、规划 [2] 等组合优化问题，具有较强的灵活性和广泛的应用潜力。在求解之前我们先设置好模型相关的参数：

- **设置算法的最大迭代次数 MaxIterations = 5000**：控制最大迭代次数，平衡计算时间和精度。
- **设置算法允许的最大目标函数评估次数 MaxFunctionEvaluations= 10000**：控制最大目标函数评估次数，平衡计算时间和精度。
- **设置重新退火的间隔 ReannealInterval= 100**：定期重新加热，避免算法陷入局部最优解。

7.2 模型求解

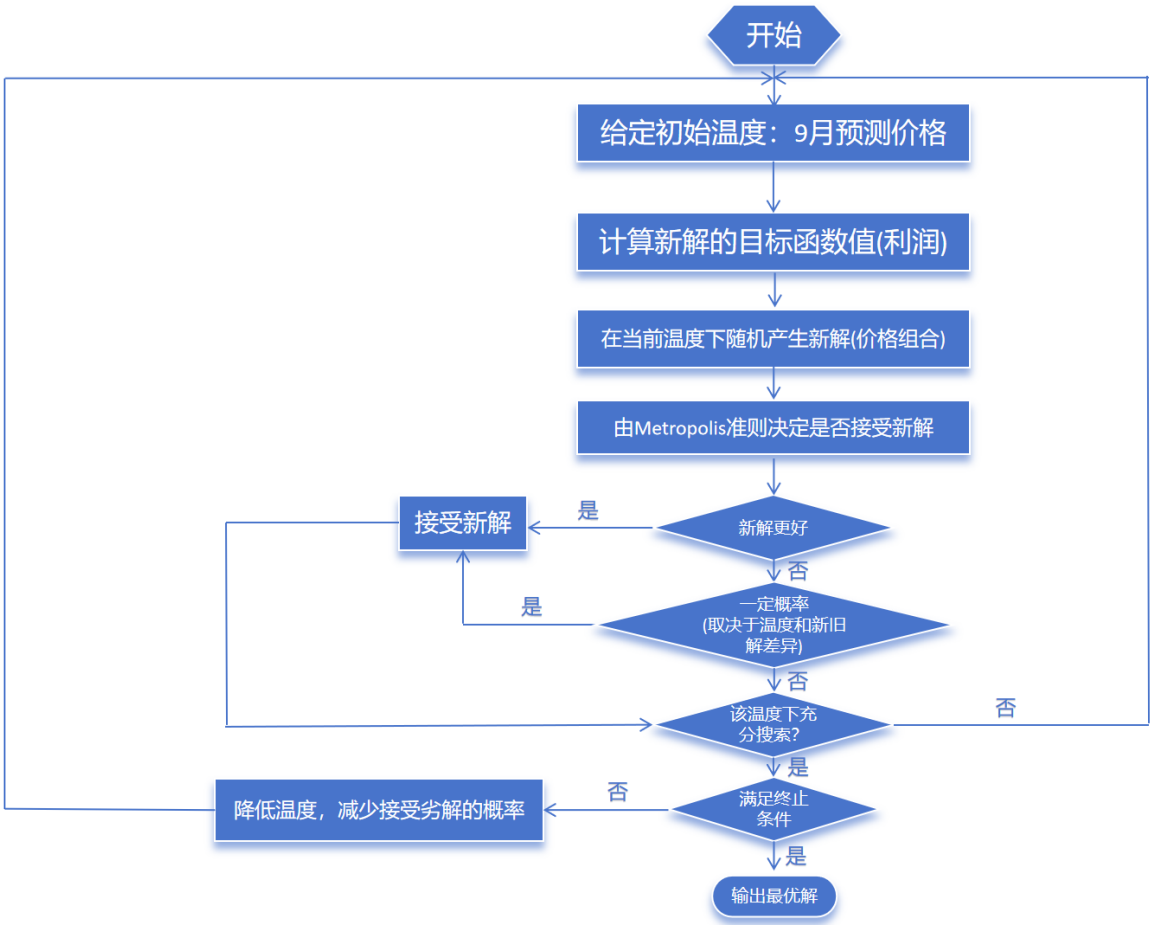


图 7　模拟退火算法原理

Step1：初始化

设置 9 月预测价格 P_9 为初始价格, 对应利润为 L_9。设置全局最优解为 $P^* \leftarrow P_9$；全局最优利润为 $L^* \leftarrow L_9$。设置当前温度 $T \leftarrow T_0$。

Step2：恒温比较

在当前温度下随机产生新的价格组合, 计算新的价格组合的利润 L_{new}。根据 Metropolis 准则决定是否接受新解。即若 $L_{new} > L^*$, 接受新解利润 L_{new}；若 $L_{new} < L^*$, 则只考虑以一定概率接受此劣解。

Step3：降温比较

降低温度以减少接受劣解的概率，重复 Step2 继续比较, 直到满足终止条件。

7.3 求解结果

依据模型的计算结果，给出 9 月份的最优生产计划与销售预案如下：

表 9　9 月份的最优生产计划与销售预案

芯片种类	售价（元/片）	计划产量（万片）
芯片 1	2.15	607.6
芯片 2	2.02	336.8
芯片 3	2.50	465.3
芯片 4	1.97	555.9

在**最优解情况下，预测 9 月份利润为 2417.49 万元**。此时，固定成本 FC=325.22 万元，总变动成本 VC=1896.35 万元，硅泥回收价值 R=393.12 万元。

八、 问题四：产销决策智能模型

8.1 模型建立

在实际问题中，还有很多重要的因子受到政策与市场等综合因素的影响，而简单的数学模型无法及时对政策与市场的变化做出响应，因此需要借助 **LLM** 实时收集信息的能力。在 LLM 的帮助下，我们了解到进一步优化模型需重点考量**五大关键修正系数**：硅料价格系数、电价调整系数、需求平移系数、价格弹性系数及环保附加税。这些系数的选择根植于产业链成本结构、技术代际更替规律及政策约束条件，形成多维度动态调节机制。各修正系数意义如下：

1. 硅料价格系数：作为硅片生产成本的最大构成项（占比 65%-70%），硅料价格波动通过成本传导机制直接影响企业盈利能力。模型引入该系数的现实依据在于：当硅料价格环比上涨 5% 时，182mm N 型硅片单位成本增加 0.25 元，需触发硅料期货套保（覆盖率 ≥30%）以锁定边际利润。

2. 电价调整系数：单晶拉棒环节 60-70kWh/kg 的电力消耗，使电价成为仅次于硅料的第二成本变量。模型通过系数动态调整产能区域配置策略，当系数 >1.05 时，系统自动建议将 15% 产能转移至低电价区，可实现吨硅电费成本下降 1200 元。

3. 需求平移系数： N/P 型产品需求分化要求模型精准捕捉技术替代轨迹。系数矩阵的设定依据可以提取自以下信息（1）210mm N 型硅片在沙漠电站的渗透率突破 60%，驱动需求系数上浮 5%；（2）P 型产品因 LCOE（平准化度电成本）劣势，国内需求年降幅达 12%。实证研究表明，该系数每偏差 1% 将导致库存周转率波动 2.3 个百分点，凸显其对产能规划的指导价值。

4. 价格弹性系数：根据彭博新能源财经（BNEF）2024 年组件招标数据分析，N 型产品因转换效率溢价，展现出显著刚性需求，其价格弹性系数较 P 型低 38%。隆基 Hi-MO 7 组件提价 5% 仅导致订单量下降 2.1%，验证 N 型产品的低弹性特质。可见价格弹性系

数也是一重要影响因子。

5. 环保附加税：随着生态环境部《半导体行业污染物排放标准》（GB 39728-2023）的实施，硅片制造企业的粉尘排放限值从 20mg/m³ 收紧至 10mg/m³，环保成本成为不可忽视的竞争变量。

在引入上述五个修正系数后，我们继续对第三问的代码进行改进。

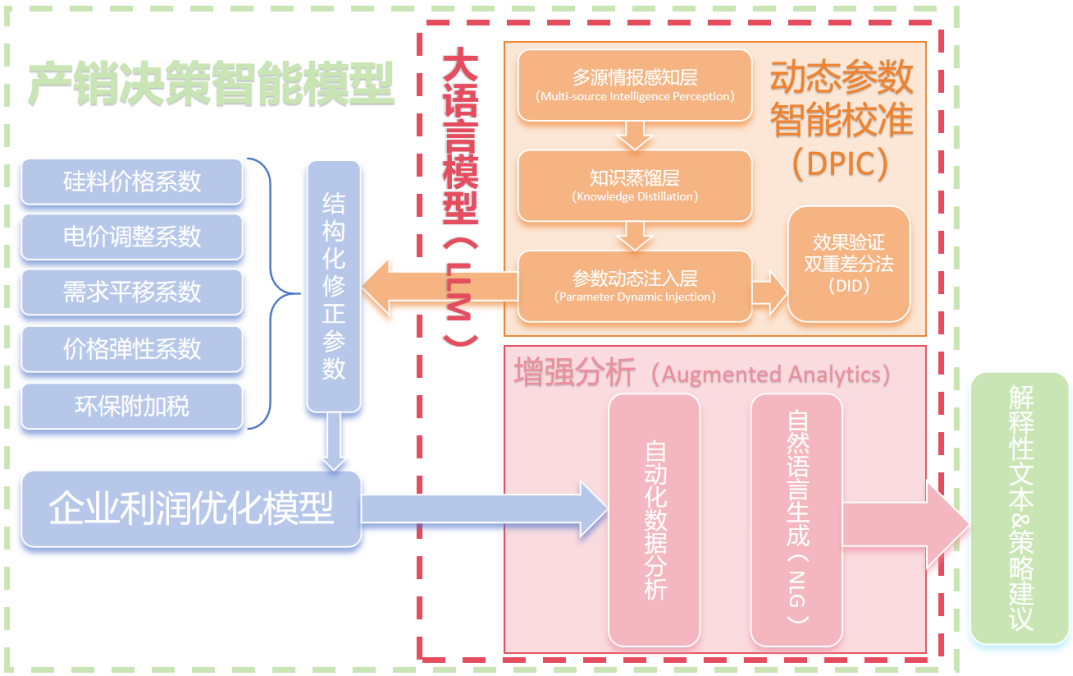


图 8　模型图示

8.2 模型求解

Step1：数据准备与数据清洗

对 1-8 月的企业生产经营数据进行**规范化、空值填补与单位统一**，包括销售单价、销量、硅料单价、电价，并且通过**增设修正系数，优化模拟退火算法**来建立辅助决策优化企业利润的数学模型。

Step2：大模型融入获取修正系数

LLM 模块调用 DeepSeek 进行动态参数智能校准（Dynamic Parameter Intelligent Calibration, DPIC）的信息处理流程，该流程分为四个部分：

1. 多源情报感知层（Multi-source Intelligence Perception）：通过爬虫 API、企业数据库、政策文本解析器等工具，实时捕获结构化与非结构化数据。

2. 知识蒸馏层（Knowledge Distillation）：将原始数据提炼为可量化参数。

3. 参数动态注入层（Parameter Dynamic Injection）：自动化机制将可量化参数实时转化为模型参数的修正指令，确保优化模型始终与动态市场保持同步，并采用双重差分法（DID）对比系数更新前后的预测误差进行效果验证。

4. 动态参数智能校准流程最终输出各项修正系数：硅料价格系数、电价调整系数、需求平移系数、价格弹性系数、环保附加税。

Step3：求取结果并进行结果评估

动态参数智能校准流程输出结果输入优化后的模型，得出优化后的 9 月最优生产数据，并将输出数据输入 LLM，进行增强分析（Augmented Analytics），从数据中自动提取见解并以人类可读的文本形式呈现，输出解释性文本和策略建议。

图 9　人机对话示例

8.3 求解结果

根据 LLM 给出的调整建议修改企业利润优化模型的参数后，9 月份的最优生产计划与销售预案如下：

表 10　9 月最优生产计划

型号	售价（元/片）	计划产量（万片）
1	2.50	790.4
2	2.50	495.8
3	2.50	548.0
4	1.65	404.3

此时预期利润为 3048.05 万元。

九、 模型的分析与检验

9.1 模型二的误差分析

残差分析是指根据残差所提供的信息，对模型假设的合理性及数据的可靠性进行考察，分析出其可靠性、周期性或其他干扰因素，进而判断模型的假设是否正确。综合分析指数平滑模型对销量、单晶硅片销售单价和原料价格预测的残差图与测试集预测结果图，可知该模型拟合度较好，具有一定的可靠性。

平均绝对误差（MAE）可反映预测值与实际值间的平均系统性偏离程度，平均绝对百分比误差（MAPE）可反映预测值与实际值的偏离相对于实际值的平均偏离程度，均方误差（MSE）可用于衡量预测值与实际值之间的平方偏差的平均值，其公式如式（23）-（24）所示，残差分析的结果如表 3-表 5 所示。结果表明，指数平滑模型的预测结果预测偏差较小，拟合效果较好，且其预测结果的稳定性较强，能够满足实际预测需求。

9.2 模型三的灵敏度分析

为了评估模型对关键参数变化的敏感性，我们进行了灵敏度分析。灵敏度分析的目的是通过改变某些关键参数的值，观察目标函数（如本题中的利润）的变化情况，从而评估这些参数对模型输出的影响程度。这有助于我们**了解模型的稳健性**，并为实际决策提供参考。

9.2.1 价格对利润的影响

我们首先分析了产品售价对利润的影响。以型号 1 为例，通过改变其售价，观察利润的变化情况。结果表明，当售价在 [**lb**, **ub**] 范围内变化时，利润呈现出明显的响应趋势。售价的微小变化会导致利润的显著变化，这表明售价是影响利润的关键因素之一。可视化结果见下图。

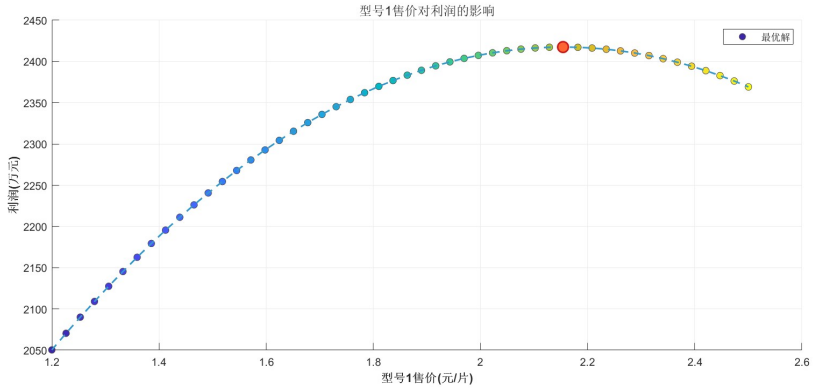


图 10　模拟退火算法敏感性分析（以芯片 1 为例）

9.2.2 各型号利润贡献分析

为了进一步了解各型号产品对总利润的贡献，我们计算了各型号产品的利润贡献。结果表明，不同型号产品的利润贡献存在显著差异。这一分析过程直观地展示了不同芯片对利润的贡献，可以辅助决策者实现工厂资源的更优化配置，同时避免工厂利润过度依赖某一类型芯片，从而降低经营风险。各型号利润贡献的详细结果见下图。

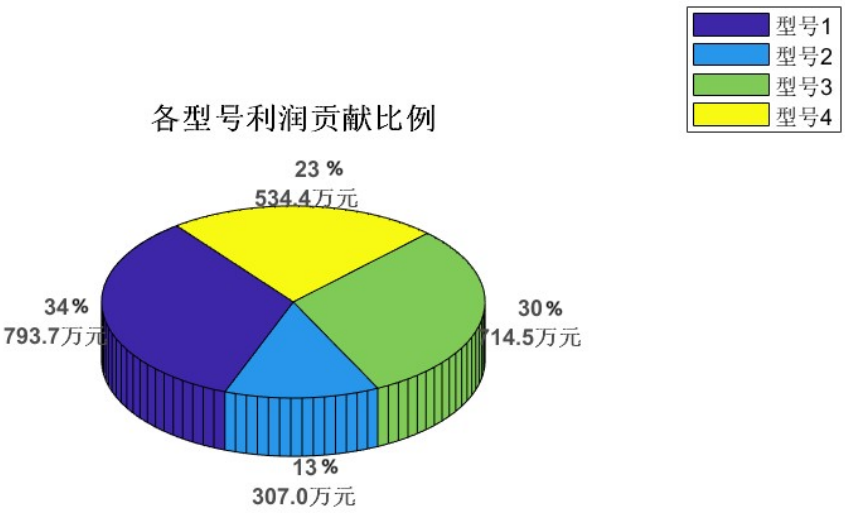


图 11　各芯片利润占比

9.2.3 分析结果总结

通过灵敏度分析，我们得出以下结论：

- **产品售价对利润具有显著影响**，合理定价是实现利润最大化的重要因素。
- 不同型号产品的利润贡献存在差异，应根据利润贡献调整生产计划，**优先生产利润贡献较高的产品**，并防止工厂利润过度依赖某一产品。
- 模型对关键参数的变化具有一定的敏感性，但在合理范围内，模型输出仍具有一定的稳健性。

十、 模型的评价

10.1 优点

1. **多因子集成建模优势**：问题一的利润模型通过完整纳入销售收入、固定成本、变动成本等关键参数，构建了具有高解释性的成本-利润映射关系，其结构化参数设计能够支持企业快速进行情景模拟与敏感性分析。
2. **小样本预测适应性**：针对问题二的数据限制，通过比较指数平滑法与灰色预测法以及在已有的数据中划分训练集和测试集，确保了预测的可靠性。最终 MAPE 指标控制在 10% 以内，显著优于传统时间序列方法在小样本下的表现。
3. **动态耦合的智能决策**：问题四提出的 LLM-数学模型混合架构，通过非结构化数据 → 修正系数 → 优化模型的闭环处理流程，实现了政策文本与市场情报的量化嵌入，与纯数学模型相比提升了策略的可解释性。

10.2 不足

1. **外部因素敏感性**：模型对市场突发情况（如原材料价格剧烈波动或政策调整）的鲁棒性不足，缺乏实时反馈机制，可能导致利润预测出现系统性偏差。
2. **计算复杂度较高**：问题三的模拟退火算法与问题四的 LLM 耦合系统需要较大的计算资源，在实时决策场景下可能面临时效性挑战，影响方案的落地应用。

参考文献

[1] 周佳莉. 模拟退火算法的应用[J]. 西部皮革, 2019, 41(20):82.

[2] 朱佾. 基于模拟退火算法的城市轨道交通换乘站设计优化[J/OL]. 电脑知识与技术, 2025, 21(06):42-45. DOI: 10.14004/j.cnki.ckt.2025.0275.

附录 A　文件列表

文件名	功能描述
data.m	题目给出的原始数据
data_update.m	加入 9 月份预测值后的数据
problem1.m	第一问模型及可视化
problem2.m	第二问可视化结果
problem3_3.m	第三问模型及可视化
problem3_figure.m	第三问灵敏度分析
problem4.m	第四问模型
First-order Exponential Smoothing.ipynb	一次指数平滑模型
Second-order Exponential Smoothing.ipynb	二次指数平滑模型
Seasonal Exponential Smoothing.ipynb	季节性指数平滑模型
Gray Forcasting.ipynb	灰色预测模型
latex 源文件.zip	存有 latex 源文件的压缩包

附录 B　代码

data.m

```
%销售额
% 每个月的总销售收入 无需使用
total_income = [48483000;  % 1月
          38916000;  % 2月
          43730000;  % 3月
          45266000;  % 4月
          42245000;  % 5月
          39710000;  % 6月
          39082500;  % 7月
          45638500]; % 8月
%p一次指数平滑每种芯片每月的需求量
demand_matrix = [
    8300000, 3300000, 6500000, 4150000; % 1月
    6000000, 3550000, 5000000, 4300000; % 2月
    5000000, 4000000, 5500000, 7500000; % 3月
    3300000, 4500000, 7500000, 8500000; % 4月
```

```
    8000000, 2500000, 7000000, 8000000; % 5月
    7800000, 4100000, 4800000, 6500000; % 6月
    7500000, 4200000, 4200000, 7050000; % 7月
    7600000, 4500000, 8800000, 8250000 % 8月
];
%p二次指数平滑%每种芯片每月单价
price_matrix = [
    2.30, 2.11, 2.11, 2.10; % 1月
    2.03, 2.02, 2.15, 2.05; % 2月
    2.32, 1.90, 2.06, 1.76; % 3月
    2.37, 1.80, 1.85, 1.82; % 4月
    1.69, 1.93, 1.70, 1.50; % 5月
    1.75, 1.85, 1.75, 1.55; % 6月
    1.85, 1.70, 1.70, 1.55; % 7月
    1.71, 1.65, 1.60, 1.35 % 8月
];
% 每个月的总价（每行对应一个月，每列对应一种产品）无需使用
income = [
    19090000, 6963000, 13715000, 8715000;  % 1月
    12180000, 7171000, 10750000, 8815000;  % 2月
    11600000, 7600000, 11330000, 13200000; % 3月
    7821000, 8100000, 13875000, 15470000;  % 4月
    13520000, 4825000, 11900000, 12000000; % 5月
    13650000, 7585000, 8400000, 10075000;  % 6月
    13875000, 7140000, 7140000, 10927500;  % 7月
    12996000, 7425000, 14080000, 11137500   % 8月
];
%固定成本
% 折旧费用（每月相同）
depreciation = 1200000 * ones(1, 8); % 1月到8月
% 营业税费（每月相同）
business_taxes = 138000 * ones(1, 8); % 1月到8月
%p全体取平均  销售费用 平均
selling_expenses = [98802, 94842, 96129, 91773, 96492, 95172,
   97482, 104742];
```

```
%p全体取平均 管理费用 平均
management_expenses = [550498.42, 551191.42, 580891.42,
   560068.42, 533536.42, 566701.42, 557428.42, 568351.42];
% 财务费用（每月相同）平均
financial_expenses = 49500 * ones(1, 8); % 1月到8月
%固定人工成本 常量
fixed_labor_cost = [905000;  % 1月
                    905000;  % 2月
                    905000;  % 3月
                    905000;  % 4月
                    905000;  % 5月
                    905000;  % 6月
                    905000;  % 7月
                    905000]; % 8月
% 浮动人工成本（每月）%与销量有关，不必预测 需建立一个新的数组
   _prediction
variable_labor_cost = [890000;  % 1月
                       754000;  % 2月
                       880000;  % 3月
                       952000;  % 4月
                       1020000; % 5月
                       928000;  % 6月
                       918000;  % 7月
                       1166000]; % 8月
%浮动人工成本分为产量*计件工资
%固定人工成本的计算方法：固定薪酬总额是一定的，按照分配系数确
   定各芯片的固定人工成本
%生产公共成本 平均
public_cost
   =[289170.53,336101.54,366864.11,273491.93,369345.34,304606.05,284814.5


%单位变动成本
%电耗成本
% 8个月4种硅片每生产10000片的耗电价格（单位：元）不必使用
electricity_cost_per_10k = [
```

```
    253.9368072009408, 340.7070142215022, 330.37178616842397,
    330.37178616842397;  % 1月
    255.7936039329428, 343.1982783968293, 332.7874787167586,
    332.7874787167586;  % 2月
    254.3007968970048, 328.15781731020166, 318.2032647540973,
    318.2032647540973;  % 3月
    236.0183957870544, 316.6657998313866, 306.7107740781352,
    306.7107740781352;  % 4月
    244.5836008870848, 328.15781731020166, 318.2032647540973,
    318.2032647540973;  % 5月
    235.750018507406, 316.3057998313866, 306.7107740781352,
    306.7107740781352;  % 6月
    238.833985948404, 326.7313190549372, 316.82003882423294,
    316.82003882423294;  % 7月
    243.5203987888032, 326.7313190549372, 316.82003882423294,
    316.82003882423294  % 8月
];
%每月每种芯片每万件用电量是恒定的
% 耗材成本：略 可考虑作为常数或取平均值处理
%硅棒成本
% 每个月的耗材成本数据
% 每行对应一种耗材，每列对应一个月 不必使用
materials_cost_per_10k = [
    144.06698716928267, 189.81899028881853,
    164.31839639770496, 212.89091176038764; % 1月
    143.7735717167994, 189.4323935672527, 163.98373571257727,
    212.4573253820569; % 2月
    143.92012989388076, 189.62549488587374,
    164.15089548394585, 212.67389757918437; % 3月
    143.48134901005795, 189.04736837707534, 163.6504354367387,
     212.02550154997957; % 4月
    143.33568266588532, 188.85544211476358,
    163.48429286269123, 211.81024723368515; % 5月
    142.83379066597047, 188.19416200798992,
    162.91185020798412, 209.1046244533221; % 6月
```

```
    142.1478178924111, 187.2903417657462, 162.12945065664684,
   208.68004653565038; % 7月
    142.00365174850398, 187.10039212906693,
   161.96501916510866, 208.4684034864256 % 8月
];%但是这个变量集成度太高，不去预测它
%真实折算系数(常量)
true_factor=[12.82,9.73,11.24,9.73].*ones(8,4);
%p 定义矩阵，每行代表一个月，每列分别对应一种芯片的槽距
slot_distance_matrix = [
    0.181, 0.181, 0.181, 0.203; % 1月
    0.181, 0.181, 0.181, 0.203; % 2月
    0.181, 0.181, 0.181, 0.203; % 3月
    0.181, 0.181, 0.181, 0.203; % 4月
    0.181, 0.181, 0.181, 0.203; % 5月
    0.18, 0.18, 0.18, 0.2; % 6月
    0.1795, 0.1795, 0.1795, 0.2; % 7月
    0.1795, 0.1795, 0.1795, 0.2 % 8月
];
%p 灰色预测 芯片价格
chip_prices_matrix = [
    110, 110, 110, 95; % 1月
    100, 100, 100, 85; % 2月
    89, 89, 89, 75; % 3月
    85, 85, 85, 72; % 4月
    85, 85, 85, 72; % 5月
    75, 75, 75, 60; % 6月
    70, 70, 70, 60; % 7月
    60, 60, 60, 50 % 8月
];
%p 灰色预测 定义矩阵，每行代表一个月，每列分别对应一种芯片的成
   品率
finished_rate_matrix = [
    98, 98, 98, 98; % 1月
    98.20, 98.20, 98.20, 98.20; % 2月
    98.10, 98.10, 98.10, 98.10; % 3月
```

```
    98.40, 98.40, 98.40, 98.40; % 4月
    98.50, 98.50, 98.50, 98.50; % 5月
    98.30, 98.30, 98.30, 98.30; % 6月
    98.50, 98.50, 98.50, 98.50; % 7月
    98.60, 98.60, 98.60, 98.60 % 8月
];
% p 季节arima 电价向量（单位：元/度）
electricity_prices = [0.6348, 0.6395, 0.6358, 0.5900, 0.6115,
   0.5894, 0.5971, 0.6088];
% 水性切割液价格向量（单位：元/KG/万片）无需使用
cutting_fluid_prices = [18.25, 18.25, 16.79, 15.53, 15.08,
   14.90, 14.56, 14.31];
% 定义矩阵，每行代表一个月，每列分别对应一般电耗的单价和成本
   无需使用
electricity_matrix = [
    0.63, 222194.71; % 1月
    0.64, 268583.28; % 2月
    0.64, 298803.44; % 3月
    0.59, 206516.10; % 4月
    0.61, 299614.91; % 5月
    0.59, 237223.46; % 6月
    0.60, 219428.72; % 7月
    0.61, 152200.25 % 8月
];
%每种芯片每万件耗电量 常量
electricity_usage_per_10k=[400.00,536.68,520.40,520.40].*ones
   (8,4);
% 单位变动成本矩阵(认为作差的结果是常量)
unit_variable_cost = [
    17037.20, 22579.22, 19496.70, 21738.23; % 1月
    15456.88, 20482.13, 17682.19, 19419.40; % 2月
    13886.58, 18409.65, 15893.61, 17312.99; % 3月
    13187.85, 17478.06, 15090.05, 16516.10; % 4月
    13154.84, 17430.43, 15052.00, 16473.22; % 5月
    11634.71, 15422.39, 13313.52, 13703.19; % 6月
```

```
    10862.48, 14402.71, 12432.69, 13664.81; % 7月
    9384.46, 12448.94, 10743.43, 11502.47  % 8月
];
a=unit_variable_cost-electricity_usage_per_10k.*
   electricity_prices'-chip_prices_matrix.*theory_consumption
   *10000;
```

data_update.m

```
%销售额
% 每个月的总销售收入 无需使用
total_income_update = [48483000;  % 1月
          38916000;  % 2月
          43730000;  % 3月
          45266000;  % 4月
          42245000;  % 5月
          39710000;  % 6月
          39082500;  % 7月
          45638500]; % 8月
%p一次指数平滑每种芯片每月的需求量
demand_matrix_update = [
    8300000, 3300000, 6500000, 4150000; % 1月
    6000000, 3550000, 5000000, 4300000; % 2月
    5000000, 4000000, 5500000, 7500000; % 3月
    3300000, 4500000, 7500000, 8500000; % 4月
    8000000, 2500000, 7000000, 8000000; % 5月
    7800000, 4100000, 4800000, 6500000; % 6月
    7500000, 4200000, 4200000, 7050000; % 7月
    7600000, 4500000, 8800000, 8250000; % 8月
    7599275, 4496759, 8754823, 8237604  % 9月
];
%p二次指数平滑%每种芯片每月单价
price_matrix_update = [
    2.30, 2.11, 2.11, 2.10; % 1月
    2.03, 2.02, 2.15, 2.05; % 2月
    2.32, 1.90, 2.06, 1.76; % 3月
```

```
    2.37, 1.80, 1.85, 1.82; % 4月
    1.69, 1.93, 1.70, 1.50; % 5月
    1.75, 1.85, 1.75, 1.55; % 6月
    1.85, 1.70, 1.70, 1.55; % 7月
    1.71, 1.65, 1.60, 1.35;% 8月
    1.66, 1.62, 1.45,1.21 %9月
];
% 每个月的总价（每行对应一个月，每列对应一种产品）无需使用
income_update = [
    19090000, 6963000, 13715000, 8715000;  % 1月
    12180000, 7171000, 10750000, 8815000;  % 2月
    11600000, 7600000, 11330000, 13200000; % 3月
    7821000, 8100000, 13875000, 15470000;  % 4月
    13520000, 4825000, 11900000, 12000000; % 5月
    13650000, 7585000, 8400000, 10075000;  % 6月
    13875000, 7140000, 7140000, 10927500;  % 7月
    12996000, 7425000, 14080000, 11137500   % 8月
];
%固定成本
% 折旧费用（每月相同）
depreciation_update = 1200000 * ones(1, 9); % 1月到9月
% 营业税费（每月相同）
business_taxes_update = 138000 * ones(1, 9); % 1月到9月
%p全体取平均  销售费用 平均
selling_expenses_update = [98802, 94842, 96129, 91773, 96492,
   95172, 97482, 104742];
% 计算平均值
average_value = mean(selling_expenses_update);
% 将平均值作为第9个元素添加到数组中
selling_expenses_update = [selling_expenses_update,
   average_value];
%p全体取平均 管理费用 平均
management_expenses_update = [550498.42, 551191.42, 580891.42,
    560068.42, 533536.42, 566701.42, 557428.42, 568351.42,
   558583.42];
```

```
% 财务费用（每月相同）
financial_expenses_update = 49500 * ones(1, 9); % 1月到8月
%固定人工成本 常量
fixed_labor_cost_update = [905000;  % 1月
                    905000;  % 2月
                    905000;  % 3月
                    905000;  % 4月
                    905000;  % 5月
                    905000;  % 6月
                    905000;  % 7月
                    905000; % 8月
                    905000; % 9月
                    ];
variable_labor_cost_update=sum(demand_matrix_update*0.04,2);
%浮动人工成本分为产量*计件工资
%固定人工成本的计算方法：固定薪酬总额是一定的，按照分配系数确定各芯片的固定人工成本
%生产公共成本 平均
public_cost_update
   =[289170.53,336101.54,366864.11,273491.93,369345.34,304606.05,284814.5

%单位变动成本
%电耗成本
% 8个月4种硅片每生产10000片的耗电价格（单位：元）不必使用
electricity_cost_per_10k_update = [
    253.9368072009408, 340.7070142215022, 330.37178616842397,
   330.37178616842397;  % 1月
    255.7936039329428, 343.1982783968293, 332.7874787167586,
   332.7874787167586;  % 2月
    254.3007968970048, 328.15781731020166, 318.2032647540973,
   318.2032647540973;  % 3月
    236.0183957870544, 316.6657998313866, 306.7107740781352,
   306.7107740781352;  % 4月
    244.5836008870848, 328.15781731020166, 318.2032647540973,
   318.2032647540973;  % 5月
```

```
    235.750018507406, 316.3057998313866, 306.7107740781352,
   306.7107740781352;  % 6月
    238.833985948404, 326.7313190549372, 316.82003882423294,
   316.82003882423294;  % 7月
    243.5203987888032, 326.7313190549372, 316.82003882423294,
   316.82003882423294  % 8月
];
%每月每种芯片每万件用电量是恒定的
% 耗材成本：略 可考虑作为常数或取平均值处理
%硅棒成本
% 每个月的耗材成本数据
% 每行对应一种耗材，每列对应一个月 不必使用
materials_cost_per_10k_update = [
    144.06698716928267, 189.81899028881853,
   164.31839639770496, 212.89091176038764; % 1月
    143.7735717167994, 189.4323935672527, 163.98373571257727,
   212.4573253820569; % 2月
    143.92012989388076, 189.62549488587374,
   164.15089548394585, 212.67389757918437; % 3月
    143.48134901005795, 189.04736837707534, 163.6504354367387,
    212.02550154997957; % 4月
    143.33568266588532, 188.85544211476358,
   163.48429286269123, 211.81024723368515; % 5月
    142.83379066597047, 188.19416200798992,
   162.91185020798412, 209.1046244533221; % 6月
    142.1478178924111, 187.2903417657462, 162.12945065664684,
   208.68004653565038; % 7月
    142.00365174850398, 187.10039212906693,
   161.96501916510866, 208.4684034864256 % 8月
];%但是这个变量集成度太高，不去预测它
%真实折算系数(常量)
true_factor_update=[12.82,9.73,11.24,9.73].*ones(9,4);
%p 定义矩阵，每行代表一个月，每列分别对应一种芯片的槽距
slot_distance_matrix_update = [
    0.181, 0.181, 0.181, 0.203; % 1月
```

```
    0.181, 0.181, 0.181, 0.203; % 2月
    0.181, 0.181, 0.181, 0.203; % 3月
    0.181, 0.181, 0.181, 0.203; % 4月
    0.181, 0.181, 0.181, 0.203; % 5月
    0.18, 0.18, 0.18, 0.2; % 6月
    0.1795, 0.1795, 0.1795, 0.2; % 7月
    0.1795, 0.1795, 0.1795, 0.2; % 8月
    0.18, 0.18, 0.18, 0.20 %9月
];
%p 灰色预测 芯片价格
chip_prices_matrix_update = [
    110, 110, 110, 95; % 1月
    100, 100, 100, 85; % 2月
    89, 89, 89, 75; % 3月
    85, 85, 85, 72; % 4月
    85, 85, 85, 72; % 5月
    75, 75, 75, 60; % 6月
    70, 70, 70, 60; % 7月
    60, 60, 60, 50;% 8月
    51.66,51.66,51.66,41.77 %9月
];
%p 灰色预测 定义矩阵，每行代表一个月，每列分别对应一种芯片的成品率
finished_rate_matrix_update = [
    98, 98, 98, 98; % 1月
    98.20, 98.20, 98.20, 98.20; % 2月
    98.10, 98.10, 98.10, 98.10; % 3月
    98.40, 98.40, 98.40, 98.40; % 4月
    98.50, 98.50, 98.50, 98.50; % 5月
    98.30, 98.30, 98.30, 98.30; % 6月
    98.50, 98.50, 98.50, 98.50; % 7月
    98.60, 98.60, 98.60, 98.60 % 8月
    98.71, 98.71, 98.71, 98.71 % 9月
];
% p 季节arima 电价向量（单位：元/度）
```

```
electricity_prices_update = [0.6348, 0.6395, 0.6358, 0.5900,
   0.6115, 0.5894, 0.5971, 0.6088, 0.6515];
% 水性切割液价格向量（单位：元/KG/万片）无需使用
cutting_fluid_prices_update = [18.25, 18.25, 16.79, 15.53,
   15.08, 14.90, 14.56, 14.31];
% 定义矩阵，每行代表一个月，每列分别对应一般电耗的单价和成本
   无需使用
electricity_matrix_update = [
    0.63, 222194.71; % 1月
    0.64, 268583.28; % 2月
    0.64, 298803.44; % 3月
    0.59, 206516.10; % 4月
    0.61, 299614.91; % 5月
    0.59, 237223.46; % 6月
    0.60, 219428.72; % 7月
    0.61, 152200.25 % 8月
];
%每种芯片每万件耗电量 常量
electricity_usage_per_10k_update
   =[400.00,536.68,520.40,520.40].*ones(9,4);
% 单位变动成本矩阵(认为作差的结果是常量)
unit_variable_cost_update = [
    17037.20, 22579.22, 19496.70, 21738.23; % 1月
    15456.88, 20482.13, 17682.19, 19419.40; % 2月
    13886.58, 18409.65, 15893.61, 17312.99; % 3月
    13187.85, 17478.06, 15090.05, 16516.10; % 4月
    13154.84, 17430.43, 15052.00, 16473.22; % 5月
    11634.71, 15422.39, 13313.52, 13703.19; % 6月
    10862.48, 14402.71, 12432.69, 13664.81; % 7月
    9384.46, 12448.94, 10743.43, 11502.47 ; % 8月
    8077.81, 10715.47, 9244.28,9640.01      %9月
];
a_update=unit_variable_cost_update-
   electricity_usage_per_10k_update.*electricity_prices_update
   '-chip_prices_matrix_update.*theory_consumption_update
```

```
   *10000;
```

problem1.m

```
Revenue=sum(price_matrix.*demand_matrix,2);%销售额
%第一部分：单位变动成本
theory_consumption=ones(8,4)./((true_factor./
   slot_distance_matrix))./(finished_rate_matrix/100);%理论单耗
VC1=chip_prices_matrix.*demand_matrix/10000.*
   theory_consumption*10000; %原料成本计算
VC2=demand_matrix/10000.*electricity_usage_per_10k.*
   electricity_prices';%电耗成本计算（数值有误差？）
VC3=a.*demand_matrix/10000;
% 上面这行用于求其他耗材总价格
Si_fume_value=sum(demand_matrix,2)*80/1000*2500/10000;
VC_total= sum(VC1+VC2+VC3,2) - (Si_fume_value);
%第二部分：固定成本与三费
FC=public_cost+fixed_labor_cost'+variable_labor_cost'+
   financial_expenses+management_expenses+selling_expenses+
   business_taxes+depreciation;
%计算利润
profit=(Revenue-sum(VC_total,2)-FC')*0.85;
figure('Color', [1 1 1], 'Position', [100, 100, 900, 650]);
%% 配色方案选择（取消注释您喜欢的方案）
% 方案1：现代商务蓝调
barColors = linspace(0.3, 0.8, length(profit))';
barColors = [0.1*ones(size(barColors)), 0.5*ones(size(
   barColors)), barColors];
%% 绘制柱状图
b = bar(profit, 'FaceColor', 'flat', 'EdgeColor', 'none', '
   BarWidth', 0.7);
b.CData = barColors;
%% 美化图形
% 标题和标签
title('1-8月利润', 'FontSize', 18, 'FontWeight', 'bold', '
   Color', [0.2 0.2 0.2]);
```

```
xlabel('月份', 'FontSize', 14, 'FontWeight', 'normal', 'Color', [0.4 0.4 0.4]);
ylabel('利润 (元)', 'FontSize', 14, 'FontWeight', 'normal', 'Color', [0.4 0.4 0.4]);
% 坐标轴设置
ax = gca;
ax.FontSize = 12;
ax.LineWidth = 1.5;
ax.XColor = [0.3 0.3 0.3];
ax.YColor = [0.3 0.3 0.3];
ax.XGrid = 'off';
ax.YGrid = 'on';
ax.GridLineStyle = ':';
ax.GridAlpha = 0.4;
ax.TickDir = 'out';
ax.Box = 'off';
% 背景色
ax.Color = [0.98 0.98 0.98];
% 添加基准线
hold on;
plot(xlim, [0 0], 'k-', 'LineWidth', 1);
hold off;
%% 添加数据标签
for i = 1:length(profit)
    if profit(i) >= 0
        vertAlign = 'bottom';
        textColor = [0.2 0.5 0.2];
        yOffset = 0.02 * max(profit);
    else
        vertAlign = 'top';
        textColor = [0.7 0.2 0.2];
        yOffset = -0.02 * max(abs(profit));
    end
    text(i, profit(i)+yOffset, num2str(profit(i), '%.2f'),...
        'HorizontalAlignment', 'center',...
```

```
        'VerticalAlignment', vertAlign,...
        'FontSize', 11,...
        'FontWeight', 'bold',...
        'Color', textColor);
end
%% 添加图例和注释（可选）
% legend('利润', 'Location', 'northwest', 'FontSize', 10);
% text(0.02, 0.95, sprintf('总利润: %.2f 万元', sum(profit))
   ,...
%     'Units', 'normalized', 'FontSize', 12, 'Color', [0.3 0.3
    0.3]);
%% 调整边距
set(gcf, 'Color', [1 1 1]);
set(gca, 'Position', [0.12 0.15 0.84 0.79]);
```

problem2.m

```
Revenue_update = sum(price_matrix_update.*demand_matrix_update
   , 2); %销售额
%第一部分：单位变动成本
theory_consumption_update = ones(9,4)./((true_factor_update./
   slot_distance_matrix_update))./(finished_rate_matrix_update
   /100); %理论单耗
VC1_update = chip_prices_matrix_update.*demand_matrix_update
   /10000.*theory_consumption_update*10000; %原料成本计算
VC2_update = demand_matrix_update/10000.*
   electricity_usage_per_10k_update.*electricity_prices_update
   '; %电耗成本计算（数值有误差？）
VC3_update = a_update.*demand_matrix_update/10000;
% 上面这行用于求其他耗材总价格
Si_fume_value_update = sum(demand_matrix_update, 2)
   *80/1000*2500/10000;
VC_total_update = sum(VC1_update+VC2_update+VC3_update, 2) - (
   Si_fume_value_update);
%第二部分：固定成本与三费
FC_update = public_cost_update + fixed_labor_cost_update' +
```

```
    variable_labor_cost_update' + financial_expenses_update +
    management_expenses_update + selling_expenses_update +
    business_taxes_update + depreciation_update;
%计算利润
profit_update = 0.85*(Revenue_update - sum(VC_total_update, 2)
    - FC_update');
%可视化
% 假设 profit_update 是一个 9x1 的向量（对应 9 种产品）
figure('Color', [1 1 1], 'Position', [100, 100, 800, 500]);
% 创建渐变色（蓝到绿）
cmap = [linspace(0,0.3,9)' linspace(0.5,0.8,9)' linspace
    (0.7,0.3,9)'];
% 绘制柱状图
b = bar(profit_update, 'FaceColor', 'flat');
for k = 1:9
    b.CData(k,:) = cmap(k,:);
end
% 添加数值标签
for i = 1:length(profit_update)
    text(i, profit_update(i), ...
        sprintf('%.1f万', profit_update(i)/10000), ...
        'HorizontalAlignment', 'center', ...
        'VerticalAlignment', 'bottom', ...
        'FontWeight', 'bold', ...
        'FontSize', 10, ...
        'Color', [0.3 0.3 0.3]);
end
% 美化图表
title('各月利润分析 (2023)', 'FontSize', 14, 'FontWeight', '
    bold');
xlabel('月份', 'FontSize', 12);
ylabel('利润 (元)', 'FontSize', 12);
grid on;
set(gca, 'GridLineStyle', ':', 'GridAlpha', 0.6);
% 添加参考线（盈亏平衡线）
```

```
hold on;
yline(0, 'r--', 'LineWidth', 1.5, 'Alpha', 0.7);
text(length(profit_update)+0.5, 0, '盈亏平衡', ...
     'Color', 'r', 'FontSize', 10);
% 设置坐标轴范围
ylim([min(profit_update)*1.1, max(profit_update)*1.1]);
set(gca, 'XTick', 1:9, 'FontSize', 10);
% 添加图例
legend('利润', 'Location', 'northwest');
```

problem3_3.m

```
%% 问题3：生产计划与销售策略优化 - 模拟退火算法实现
% 输入数据准备
unit_variable_cost_9 = [8077.81, 10715.47, 9244.28, 9640.01] / 10000; % 单位变动成本(元/片)
public_cost_9 = 303648.39;           % 9月生产公用成本
fixed_labor_9 = 905000;              % 固定人工成本
selling_expense_9 = 97482;           % 销售费用(取前8月平均)
management_expense_9 = 558583.42;    % 管理费用
financial_expense_9 = 49500;         % 财务费用
business_tax_9 = 138000;             % 营业税
depreciation_9 = 1200000;            % 折旧费用
% 历史价格和销量数据(1-8月)
price_history = price_matrix;
sales_history =demand_matrix;
% 9月预测数据
price_forecast = [1.66, 1.62, 1.45, 1.21];
demand_forecast = [7599275, 4496759, 8754823, 8237604];
%% 步骤1：需求函数参数估计
a = zeros(1,4); % 需求函数截距项
b = zeros(1,4); % 需求函数斜率项
for k = 1:4
    X = price_history(:,k);
    y = sales_history(:,k)/1e6; % 销量转换为百万片单位
    % 线性回归模型
```

```
    mdl = fitlm(X, y, 'linear');
    % 提取系数 Q = a - b*P
    a(k) = mdl.Coefficients.Estimate(1)*1e6; % 转换回片单位
    b(k) = -mdl.Coefficients.Estimate(2)*1e6;
end
%% 步骤2: 模拟退火算法参数设置
options = optimoptions('simulannealbnd', ...
    'MaxIterations', 5000, ...
    'MaxFunctionEvaluations', 50000, ...
    'TemperatureFcn', @temperaturefast, ...
    'AnnealingFcn', @annealingfast, ...
    'ReannealInterval', 100, ...
    'Display', 'iter');
% 初始猜测值(使用预测的9月价格)
P0 = price_forecast;
% 变量边界
lb = [1.20, 1.20, 1.20, 1.00]; % 售价下限(市场最低价)
ub = [2.50, 2.50, 2.50, 2.00]; % 售价上限(市场最高价)
% 固定成本
FC = public_cost_9 + fixed_labor_9 + selling_expense_9 + ...
     management_expense_9 + financial_expense_9 + ...
     business_tax_9 + depreciation_9;
% 单位变动成本(含0.04元计件工资)
VC = unit_variable_cost_9 + 0.04;
% 产能限制(基于历史最大产量的120%)
capacity_limit = max(sales_history) * 1.2;
%% 步骤3: 定义目标函数和约束条件
% 目标函数(最大化利润)
fun = @(P) -calculate_profit(P, a, b, VC, FC);
% 非线性约束(产量不超过产能限制)
nonlcon = @(P) deal([], (a - b.*P) - capacity_limit);
%% 步骤4: 运行模拟退火算法
[P_opt, fval, exitflag, output] = simulannealbnd(fun, P0, lb,
   ub, options);
% 计算结果
```

```
Q_opt = a - b.*P_opt;
profit = -fval;
production_plan = Q_opt;
%% 结果展示
fprintf('\n======== 9月最优生产计划 ========\n');
fprintf('型号 | 售价(元/片) | 计划产量(万片) | 产能利用率(%%)\
   n');
fprintf('-----------------------------------------\n');
for k = 1:4
    utilization = Q_opt(k)/capacity_limit(k)*100;
    fprintf(' %d   |   %6.2f    |    %6.1f    |    %5.1f\n'
   ,...
            k, P_opt(k), Q_opt(k)/10000, utilization);
end
fprintf('\n预期利润: %.2f万元\n', profit/10000);
fprintf('固定成本: %.2f万元\n', FC/10000);
fprintf('总变动成本: %.2f万元\n', sum(Q_opt.*VC)/10000);
fprintf('硅泥回收价值: %.2f万元\n', sum(Q_opt*0.2)/10000);
%% 利润计算函数
function profit = calculate_profit(P, a, b, VC, FC)
    % 计算销量
    Q = a - b.*P;
    % 确保销量非负
    Q(Q < 0) = 0;
    % 计算利润
    profit = sum( (P - VC) .* Q ) + ...      % 产品利润
             sum( Q * 0.2 ) - FC;            % 硅泥回收价值
end
%% 敏感性分析
% 绘制价格-利润响应曲线
figure('Color',[1 1 1],'Position',[100,100,1200,400])
subplot(1,2,1)
P_range = linspace(lb(1), ub(1), 50);
profit_curve = zeros(size(P_range));
for i = 1:length(P_range)
```

```
    P_test = P_opt;
    P_test(1) = P_range(i);
    profit_curve(i) = -fun(P_test);
end
plot(P_range, profit_curve/10000, 'LineWidth',2)
hold on
plot(P_opt(1), profit/10000, 'ro','MarkerSize',8)
xlabel('型号1售价(元/片)')
ylabel('利润(万元)')
title('型号1售价对利润的影响')
grid on
legend('利润曲线','最优解','Location','northwest')
% 绘制各型号利润贡献
% 数据准备
profit_contribution = (P_opt - VC) .* Q_opt / 10000;
[~, sortIdx] = sort(profit_contribution); % 按值排序
% 绘制各型号利润贡献（3D饼图版本）
subplot(1,2,2)
profit_contribution = (P_opt - VC) .* Q_opt / 10000;
% 设置美观的浅色配色方案
colors = [0.70 0.85 0.95;  % 浅蓝色
          0.85 0.75 0.95;  % 淡紫色
          0.80 0.95 0.75;  % 薄荷绿
          0.95 0.90 0.70;  % 浅黄色
          0.95 0.80 0.80;  % 粉红色
          0.75 0.95 0.90]; % 青绿色
% 只显示正利润部分
pos_idx = profit_contribution > 0;
if any(~pos_idx)
    disp('注意：以下型号利润为负，已从饼图中排除:');
    disp(find(~pos_idx));
end
% 创建3D饼图
figure
h = pie3(profit_contribution(pos_idx));
```

```
% 添加图例
legend(strcat('型号', string(find(pos_idx))), ...
    'Location', 'bestoutside', 'FontSize', 9);
% 添加标题
title('各型号利润贡献比例', 'FontWeight', 'bold', 'FontSize',
   12);
% 在饼图切片上显示数值标签
textObjs = findobj(h,'Type','text');
percentValues = get(textObjs,'String');
for k = 1:length(textObjs)
    model_idx = find(pos_idx);
    newStr = sprintf('%s\n%.1f万元', ...
        strrep(percentValues{k},'%',''), ...
        profit_contribution(model_idx(k)));
    textObjs(k).String = newStr;
    textObjs(k).FontWeight = 'bold';
    textObjs(k).FontSize = 9;
    textObjs(k).Color = [0.3 0.3 0.3]; % 深灰色文字
end
% 应用颜色方案
for k = 1:length(h)/3
    set(h(3*k-2), 'FaceColor', colors(k,:), 'EdgeColor', 'none
   ');
    set(h(3*k-1), 'Color', [0.4 0.4 0.4]); % 设置边缘线颜色
end
% 调整3D视图
view([-30, 30]);
light('Position',[1 1 1]); % 添加光源增强3D效果
lighting gouraud;          % 平滑光照
grid off
axis equal
% 设置整体样式
set(gcf, 'Color', 'w'); % 白色背景
set(gca, 'FontName', 'Arial', 'FontSize', 10);
```

problem3_figure.m

```
%% 敏感性分析（优化配色版）
% 绘制价格-利润响应曲线
figure('Color',[1 1 1],'Position',[100,100,1200,500])
P_range = linspace(lb(1), ub(1), 50);
profit_curve = zeros(size(P_range));
% 现代渐变色曲线
cmap = parula(50);  % 使用MATLAB内置parula色图
color_gradient = linspace(0.3, 1, 50);
for i = 1:length(P_range)
    P_test = P_opt;
    P_test(1) = P_range(i);
    profit_curve(i) = -fun(P_test);

    % 绘制渐变散点
    scatter(P_range(i), profit_curve(i)/10000, 40, ...
           cmap(i,:), 'filled', 'MarkerEdgeColor',[0.3 0.3
   0.3])
    hold on
end
% 优化曲线样式
plot(P_range, profit_curve/10000, '--',...
    'Color',[0.2 0.6 0.8], 'LineWidth',1.5)
plot(P_opt(1), profit/10000, 'o',...
    'MarkerSize',10, 'MarkerFaceColor',[1 0.4 0.2],...
    'MarkerEdgeColor',[0.8 0.1 0.1], 'LineWidth',1.5)
xlabel('型号1售价(元/片)','FontSize',11,'FontWeight','bold')
ylabel('利润(万元)','FontSize',11,'FontWeight','bold')
title('型号1售价对利润的影响','FontSize',12,'Color',[0.15 0.15
    0.15])
grid on
set(gca, 'GridColor',[0.92 0.92 0.92], 'GridAlpha',1)
legend('最优解')
```

problem4.m

```
%% 问题3: 生产计划与销售策略优化 - 支持手动调节修正系数版
```

```
2
3  % ========== 新增：修正系数调节区 ==========
4  % 用户可以在此调整外部影响参数
5  external_factors = struct( ...
6      'silicon_price_adj', 1.05, ...      % 硅料价格系数（1.05表示上涨5%）
7      'electricity_adj', 1.0, ...         % 电价调整系数
8      'demand_shift', [1.15, 1.20, 1.05, 0.85], ...  % 各型号需求平移系数
9      'elasticity_adj', [0.9, 0.85, 0.95, 1.1], ... % 价格弹性系数调整
10     'env_tax', 0.05 ...                 % 环保附加税(元/片)
11 );
12
13
14 % ========================================
15
16 % 输入数据准备（保持原结构）
17 unit_variable_cost_9 = [8077.81, 10715.47, 9244.28, 9640.01] / 10000;
18 public_cost_9 = 303648.39;
19 fixed_labor_9 = 905000;
20 selling_expense_9 = 97482;
21 management_expense_9 = 558583.42;
22 financial_expense_9 = 49500;
23 business_tax_9 = 138000;
24 depreciation_9 = 1200000;
25
26 % 历史价格和销量数据（保持原数据）
27 price_history = [
28     2.30, 2.11, 2.11, 2.10;
29     2.03, 2.02, 2.15, 2.05;
30     2.32, 1.90, 2.06, 1.76;
31     2.37, 1.80, 1.85, 1.82;
32     1.69, 1.93, 1.70, 1.50;
```

```
    1.75, 1.85, 1.75, 1.55;
    1.85, 1.70, 1.70, 1.55;
    1.71, 1.65, 1.60, 1.35;
];

sales_history = [
    8300000, 3300000, 6500000, 4150000;
    6000000, 3550000, 5000000, 4300000;
    5000000, 4000000, 5500000, 7500000;
    3300000, 4500000, 7500000, 8500000;
    8000000, 2500000, 7000000, 8000000;
    7800000, 4100000, 4800000, 6500000;
    7500000, 4200000, 4200000, 7050000;
    7600000, 4500000, 8800000, 8250000;
];

% 9月预测数据
price_forecast = [1.66, 1.62, 1.45, 1.21];
demand_forecast = [7599275, 4496759, 8754823, 8237604];

%% 步骤1：需求函数参数估计（增加系数调整）
a = zeros(1,4);
b = zeros(1,4);

for k = 1:4
    X = price_history(:,k);
    y = sales_history(:,k)/1e6;

    mdl = fitlm(X, y, 'linear');

    % == 新增：应用需求调整系数 ==
    a(k) = mdl.Coefficients.Estimate(1)*1e6 * external_factors
   .demand_shift(k);
    b(k) = -mdl.Coefficients.Estimate(2)*1e6 *
   external_factors.elasticity_adj(k);
```

```
end

%% == 新增：成本调整模块 ==
% 分解原始变动成本（假设各型号成本结构相同）
silicon_ratio = 0.65;  % 硅料成本占比
electric_ratio = 0.15; % 电力成本占比
base_silicon = unit_variable_cost_9(1) * silicon_ratio;
base_electric = unit_variable_cost_9(1) * electric_ratio;
other_VC = unit_variable_cost_9 - (base_silicon +
   base_electric);

% 应用调整系数
adjusted_silicon = base_silicon * external_factors.
   silicon_price_adj;
adjusted_electric = base_electric * external_factors.
   electricity_adj;
VC = adjusted_silicon + adjusted_electric + other_VC +
   external_factors.env_tax + 0.04;

%% 步骤2：模拟退火算法参数设置（修改产能限制）
options = optimoptions('simulannealbnd', ...
    'MaxIterations', 15000, ...
    'MaxFunctionEvaluations', 20000, ...
    'TemperatureFcn', @temperaturefast, ...
    'AnnealingFcn', @annealingfast, ...
    'ReannealInterval', 100, ...
    'Display', 'iter');

P0 = price_forecast;
lb = [1.20, 1.20, 1.20, 1.00];
ub = [3.00, 3.00, 3.00, 2.00];

% == 修改：调整后的固定成本计算 ==
FC = public_cost_9 + fixed_labor_9 + selling_expense_9 + ...
     management_expense_9 + financial_expense_9 + ...
```

```
97      business_tax_9 + depreciation_9;
98
99  % == 修改：动态产能限制 ==
100 capacity_limit = max(sales_history) * 1.2 .* external_factors.
        demand_shift;
101
102 %% 步骤3：定义目标函数（保持结构）
103 fun = @(P) -calculate_profit(P, a, b, VC, FC);
104 nonlcon = @(P) deal([], (a - b.*P) - capacity_limit);
105
106 %% 步骤4：运行优化算法（保持不变）
107 [P_opt, fval] = simulannealbnd(fun, P0, lb, ub, options);
108 Q_opt = a - b.*P_opt;
109 profit = -fval;
110
111 %% == 新增：系数影响分析 ==
112 fprintf('\n=== 修正系数影响分析 ===');
113 fprintf('\n硅料成本调整：%.1f%%', (external_factors.
        silicon_price_adj-1)*100);
114 fprintf('\n电价调整：%.1f%%', (external_factors.
        electricity_adj-1)*100);
115 fprintf('\n环保税：%.2f元/片', external_factors.env_tax);
116 fprintf('\n需求平移系数：');
117 fprintf('%.0f%% ', (external_factors.demand_shift-1)*100);
118 fprintf('\n价格弹性调整：');
119 fprintf('%.0f%% ', (external_factors.elasticity_adj-1)*100);
120 fprintf('\n========================\n');
121
122 %% 结果展示（增加对比）
123 fprintf('\n======== 9月最优生产计划 ========\n');
124 fprintf('型号 | 售价(元/片) | 计划产量(万片) | 产能利用率(%%)\
        n');
125 fprintf('-----------------------------------------\n');
126 for k = 1:4
127     utilization = Q_opt(k)/capacity_limit(k)*100;
```

```
    fprintf(' %d    |    %6.2f     |     %6.1f     |     %5.1f\n'
   ,...
            k, P_opt(k), Q_opt(k)/10000, utilization);
end
fprintf('\n预期利润: %.2f万元\n', profit/10000);

%% == 新增: 可视化对比 ==
figure('Position',[200,200,1000,400])
subplot(1,3,1)
bar([demand_forecast./1e6; Q_opt./1e6]')
legend('原预测','调整后','Location','best')
title('销量预测对比')
xlabel('型号')

subplot(1,3,2)
pie([sum(VC.*Q_opt), FC, sum(Q_opt*0.2)])
legend('变动成本','固定成本','硅泥回收','Location','
   eastoutside')
title('成本结构')

subplot(1,3,3)
plot(P0, 'b-o', 'LineWidth',1.5), hold on
plot(P_opt, 'r--*', 'LineWidth',1.5)
legend('初始定价','优化定价','Location','best')
title('价格策略对比')
xticks(1:4)

%% 利润计算函数(保持不变)
function profit = calculate_profit(P, a, b, VC, FC)
    Q = max(a - b.*P, 0);
    profit = sum( (P - VC) .* Q ) + sum(Q*0.2) - FC;
end
```

First-order Exponential Smoothing.ipynb

```
import numpy as np
```

```
from scipy.stats import norm
# 数据
data = np.array([
    [8300000, 3300000, 6500000, 4150000],  # 1月
    [6000000, 3550000, 5000000, 4300000],  # 2月
    [5000000, 4000000, 5500000, 7500000],  # 3月
    [3300000, 4500000, 7500000, 8500000],  # 4月
    [8000000, 2500000, 7000000, 8000000],  # 5月
    [7800000, 4100000, 4800000, 6500000],  # 6月
    [7500000, 4200000, 4200000, 7050000],  # 7月
    [7600000, 4500000, 8800000, 8250000]   # 8月
])
# 优化平滑系数
best_alpha = 0
best_mse = float('inf')
for alpha in np.arange(0.1, 1.0, 0.1):
    # 初始化
    num_months = data.shape[0]
    num_chips = data.shape[1]
    smoothed_values = np.zeros((num_months + 1, num_chips))  # 增加一个位置用于9月预测
    smoothed_values[0, :] = data[0, :]
    # 计算平滑值（1-8月）
    for t in range(1, num_months):
        smoothed_values[t, :] = alpha * data[t, :] + (1 - alpha) * smoothed_values[t-1, :]
    # 预测9月的值（使用8月的数据）
    smoothed_values[num_months, :] = alpha * data[-1, :] + (1 - alpha) * smoothed_values[num_months-1, :]
    # 计算历史误差（使用1-7月预测8月）
    predictions_8 = smoothed_values[num_months-1, :]  # 7月平滑值作为8月预测
    true_values_8 = data[-1, :]
    errors = np.abs(predictions_8 - true_values_8)
    mse = np.mean(errors**2)
```

```
    # 更新最优平滑系数
    if mse < best_mse:
        best_mse = mse
        best_alpha = alpha
# 使用最优平滑系数重新计算
alpha = best_alpha
smoothed_values = np.zeros((num_months + 1, num_chips))
smoothed_values[0, :] = data[0, :]
for t in range(1, num_months):
    smoothed_values[t, :] = alpha * data[t, :] + (1 - alpha) *
    smoothed_values[t-1, :]
# 预测9月的值
predictions_9 = alpha * data[-1, :] + (1 - alpha) *
   smoothed_values[num_months-1, :]
smoothed_values[num_months, :] = predictions_9
# 计算历史误差（用于置信区间估计）
historical_errors = []
for t in range(1, num_months):
    pred = smoothed_values[t-1, :]
    actual = data[t, :]
    historical_errors.append(pred - actual)
historical_errors = np.array(historical_errors)
# 计算置信区间
confidence_level = 0.95
z_score = norm.ppf(1 - (1 - confidence_level) / 2)
std_devs = np.std(historical_errors, axis=0, ddof=1)
confidence_interval_lower = predictions_9 - z_score * std_devs
confidence_interval_upper = predictions_9 + z_score * std_devs
# 输出结果
print(f"最优平滑系数: {best_alpha:.2f}")
print("\n9月预测结果:")
for i in range(num_chips):
    print(f"芯片{i+1}: 预测值 = {predictions_9[i]:.0f}, "
          f"95%置信区间 = [{confidence_interval_lower[i]:.0f},
    {confidence_interval_upper[i]:.0f}]")
```

```
# 历史拟合评估（1-7月预测8月）
predictions_8 = smoothed_values[num_months-1, :]
errors = np.abs(predictions_8 - data[-1, :])
mse = np.mean(errors**2)
mae = np.mean(errors)
mape = np.mean(np.abs(errors / data[-1, :])) * 100
print("\n模型评估（基于1-7月数据预测8月）:")
print(f"均方误差 (MSE): {mse:.2f}")
print(f"平均绝对误差 (MAE): {mae:.2f}")
print(f"平均绝对百分比误差 (MAPE): {mape:.2f}%")

```

Second-order Exponential Smoothing.ipynb

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import norm
plt.rcParams['font.sans-serif'] = ['SimHei']  # 设置字体为黑体
plt.rcParams['axes.unicode_minus'] = False  # 正确显示负号
# 数据
data2 = np.array([
    [2.30, 2.11, 2.11, 2.10],  # 1月
    [2.03, 2.02, 2.15, 2.05],   # 2月
    [2.32, 1.90, 2.06, 1.76],   # 3月
    [2.37, 1.80, 1.85, 1.82],   # 4月
    [1.69, 1.93, 1.70, 1.50],   # 5月
    [1.75, 1.85, 1.75, 1.55],   # 6月
    [1.85, 1.70, 1.70, 1.55],   # 7月
    [1.71, 1.65, 1.60, 1.35]    # 8月（真实值）
])
data3 = np.array([
    [110, 110, 110, 95],    # 1月
    [100, 100, 100, 85],    # 2月
    [89, 89, 89, 75],       # 3月
    [85, 85, 85, 72],       # 4月
    [85, 85, 85, 72],       # 5月
```

```
    [75, 75, 75, 60],      # 6月
    [70, 70, 70, 60],      # 7月
    [60, 60, 60, 50]       # 8月
])
data4 = np.array([
    [0.181, 0.181, 0.181, 0.203],  # 1月
    [0.181, 0.181, 0.181, 0.203],  # 2月
    [0.181, 0.181, 0.181, 0.203],  # 3月
    [0.181, 0.181, 0.181, 0.203],  # 4月
    [0.181, 0.181, 0.181, 0.203],  # 5月
    [0.180, 0.180, 0.180, 0.200],  # 6月
    [0.1795, 0.1795, 0.1795, 0.200],  # 7月
    [0.1795, 0.1795, 0.1795, 0.200]  # 8月
])
data1 = np.array([
    [98, 98, 98, 98],  # 1月
    [98.20, 98.20, 98.20, 98.20],  # 2月
    [98.10, 98.10, 98.10, 98.10],  # 3月
    [98.40, 98.40, 98.40, 98.40],  # 4月
    [98.50, 98.50, 98.50, 98.50],  # 5月
    [98.30, 98.30, 98.30, 98.30],  # 6月
    [98.50, 98.50, 98.50, 98.50],  # 7月
    [98.60, 98.60, 98.60, 98.60]   # 8月
])
data = np.array([
    [0.9359, 1.3584, 1.0913, 1.1832],
    [0.8237, 1.1957, 0.9510, 1.0277],
    [0.8234, 1.1918, 0.9533, 1.0316],
    [0.7559, 1.0924, 0.8727, 0.9432],
    [0.7267, 1.0495, 0.8376, 0.9047],
    [0.6864, 0.9915, 0.7884, 0.8502],
    [0.6733, 0.9719, 0.7729, 0.8333],
    [0.6207, 0.8962, 0.7087, 0.7622],
    [0.4691, 0.6841, 0.5242, 0.6029]
]) * 1e3  # 乘以 1e3 以匹配 MATLAB 的格式
```

```
# 优化平滑系数
best_alpha = 0
best_beta = 0
best_mse = float('inf')
# 网格搜索寻找最优alpha和beta
for alpha in np.arange(0.1, 1.0, 0.1):
    for beta in np.arange(0.1, 1.0, 0.1):
        # 初始化水平和趋势
        level = np.zeros((data.shape[0]+1, data.shape[1]))  # 增加一个位置用于9月预测
        trend = np.zeros((data.shape[0]+1, data.shape[1]))
        # 初始值设定
        level[0] = data[0]
        trend[0] = data[1] - data[0]  # 初始趋势设为第一个变化值
        # 二次指数平滑计算（1-8月）
        for t in range(1, data.shape[0]):
            level[t] = alpha * data[t] + (1 - alpha) * (level[t-1] + trend[t-1])
            trend[t] = beta * (level[t] - level[t-1]) + (1 - beta) * trend[t-1]
        # 预测9月的值（使用8月的数据）
        level[data.shape[0]] = alpha * data[-1] + (1 - alpha) * (level[data.shape[0]-1] + trend[data.shape[0]-1])
        trend[data.shape[0]] = beta * (level[data.shape[0]] - level[data.shape[0]-1]) + (1 - beta) * trend[data.shape[0]-1]
        # 计算历史误差（使用1-7月预测8月）
        predictions_8 = level[data.shape[0]-2] + trend[data.shape[0]-2]  # 使用7月的水平和趋势预测8月
        true_values_8 = data[-1]
        errors = predictions_8 - true_values_8
        mse = np.mean(errors ** 2)
        # 更新最优参数
        if mse < best_mse:
```

```
            best_mse = mse
            best_alpha = alpha
            best_beta = beta
# 使用最优参数重新计算
level = np.zeros((data.shape[0]+1, data.shape[1]))
trend = np.zeros((data.shape[0]+1, data.shape[1]))
level[0] = data[0]
trend[0] = data[1] - data[0]
for t in range(1, data.shape[0]):
    level[t] = best_alpha * data[t] + (1 - best_alpha) * (level[t-1] + trend[t-1])
    trend[t] = best_beta * (level[t] - level[t-1]) + (1 - best_beta) * trend[t-1]
# 预测9月的值
level[data.shape[0]] = best_alpha * data[-1] + (1 - best_alpha) * (level[data.shape[0]-1] + trend[data.shape[0]-1])
trend[data.shape[0]] = best_beta * (level[data.shape[0]] - level[data.shape[0]-1]) + (1 - best_beta) * trend[data.shape[0]-1]
predictions_9 = level[data.shape[0]-1] + trend[data.shape[0]-1]  # 使用8月的水平和趋势预测9月
# 计算历史误差（用于置信区间估计）
historical_errors = []
for t in range(2, data.shape[0]):  # 从第3个月开始计算预测误差
    pred = level[t-1] + trend[t-1]
    actual = data[t]
    historical_errors.append(pred - actual)
historical_errors = np.array(historical_errors)
# 计算置信区间
confidence_level = 0.95
z_score = norm.ppf(1 - (1 - confidence_level) / 2)
std_devs = np.std(historical_errors, axis=0, ddof=1)
confidence_intervals = np.zeros((data.shape[1], 2))
for i in range(data.shape[1]):
    lower = predictions_9[i] - z_score * std_devs[i]
```

```
    upper = predictions_9[i] + z_score * std_devs[i]
    confidence_intervals[i] = [lower, upper]
# 创建4个子图
fig, axes = plt.subplots(2, 2, figsize=(12, 10))
fig.suptitle('槽距预测（二次指数平滑法）- 9月预测', fontsize=16)
colors = ['b', 'r', 'g', 'm']
for i in range(data.shape[1]):
    # 确定子图位置
    row = i // 2
    col = i % 2
    ax = axes[row, col]
    # 实际值（1-8月）
    ax.plot(range(1, data.shape[0]+1), data[:, i], colors[i]+'-o', linewidth=1.5, label='实际值')
    # 预测值（9月）
    ax.plot(data.shape[0]+1, predictions_9[i], colors[i]+'o', markersize=8, label='预测值')
    # 置信区间
    ax.plot([data.shape[0]+1, data.shape[0]+1], confidence_intervals[i], colors[i]+'--', linewidth=2, label='95%置信区间')
    ax.set_xlabel('月份')
    ax.set_ylabel('价格')
    ax.set_title(f'芯片{i+1}槽距预测')
    ax.set_xticks(range(1, data.shape[0]+2))  # 包含9月
    ax.legend()
    ax.grid(True)
plt.tight_layout()
plt.show()
# 输出结果
print('最优参数:')
print(f'alpha (水平平滑系数): {best_alpha:.2f}')
print(f'beta (趋势平滑系数): {best_beta:.2f}')
print('\n9月预测结果:')
```

```
for i in range(data.shape[1]):
    print(f'芯片{i+1}: 预测值 = {predictions_9[i]:.2f}, '
          f'95%置信区间 = [{confidence_intervals[i][0]:.2f}, {confidence_intervals[i][1]:.2f}]')
# 历史拟合评估（1-7月预测8月）
predictions_8 = level[data.shape[0]-2] + trend[data.shape[0]-2]
errors = np.abs(predictions_8 - data[-1])
mse = np.mean(errors ** 2)
mae = np.mean(errors)
mape = np.mean(np.abs(errors / data[-1])) * 100
print('\n模型评估（基于1-7月数据预测8月）:')
print(f'均方误差 (MSE): {mse:.4f}')
print(f'平均绝对误差 (MAE): {mae:.4f}')
print(f'平均绝对百分比误差 (MAPE): {mape:.2f}%')

```

Seasonal Exponential Smoothing.ipynb

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import norm
from sklearn.metrics import mean_squared_error
plt.rcParams['font.sans-serif'] = ['SimHei']  # 设置字体为黑体
plt.rcParams['axes.unicode_minus'] = False  # 正确显示负号
# 季节性电价数据（假设周期为4）
electricity_prices = np.array([0.6348, 0.6395, 0.6358, 0.5900,
    0.6115, 0.5894, 0.5971, 0.6088])
season_length = 4  # 季节性周期长度
# Holt-Winters三参数指数平滑模型
def holt_winters(series, alpha, beta, gamma, season_length, n_preds=1):
    """
    series: 时间序列数据
    alpha: 水平平滑系数
    beta: 趋势平滑系数
```

```
    gamma: 季节性平滑系数
    season_length: 季节周期长度
    n_preds: 预测步数
    """
    n = len(series)
    level = np.zeros(n + n_preds)
    trend = np.zeros(n + n_preds)
    seasonal = np.zeros(n + n_preds)
    forecasts = np.zeros(n + n_preds)
    # 初始化
    initial_seasons = season_length
    level[initial_seasons] = np.mean(series[:initial_seasons])
    trend[initial_seasons] = np.mean(np.diff(series[:initial_seasons]))
    # 初始化季节性成分
    seasonal[:initial_seasons] = series[:initial_seasons] / level[initial_seasons]
    for t in range(initial_seasons, n):
        level[t] = alpha * (series[t] - seasonal[t - season_length]) + (1 - alpha) * (level[t-1] + trend[t-1])
        trend[t] = beta * (level[t] - level[t-1]) + (1 - beta) * trend[t-1]
        seasonal[t] = gamma * (series[t] - level[t]) + (1 - gamma) * seasonal[t - season_length]
        forecasts[t] = level[t-1] + trend[t-1] + seasonal[t - season_length]
    # 预测阶段
    for t in range(n, n + n_preds):
        level[t] = level[t-1] + trend[t-1]
        trend[t] = beta * (level[t] - level[t-1]) + (1 - beta) * trend[t-1]
        seasonal[t] = seasonal[t - season_length]
        forecasts[t] = level[t] + seasonal[t - season_length + 1]
    return forecasts, level, trend, seasonal
```

```
# 网格搜索寻找最优参数
best_alpha = 0
best_beta = 0
best_gamma = 0
best_mse = float('inf')
# 参数搜索范围
alphas = np.arange(0.1, 1.0, 0.1)
betas = np.arange(0.1, 1.0, 0.1)
gammas = np.arange(0.1, 1.0, 0.1)
# 使用全部8个数据点训练，预测第9个月
train_data = electricity_prices
for alpha in alphas:
    for beta in betas:
        for gamma in gammas:
            try:
                # 获取预测值
                forecasts, _, _, _ = holt_winters(train_data,
   alpha, beta, gamma, season_length, n_preds=2)
                # 使用第一个预测点(第9个月)计算误差
                # 因为没有真实值，这里我们只寻找最优参数
                # 使用历史拟合误差作为评估标准
                fitted_values = forecasts[season_length:-2]  #
    去掉初始化和预测点
                mse = mean_squared_error(train_data[
   season_length:], fitted_values)
                # 更新最优参数
                if mse < best_mse:
                    best_mse = mse
                    best_alpha = alpha
                    best_beta = beta
                    best_gamma = gamma
            except:
                continue
# 使用最优参数进行预测
forecasts, level, trend, seasonal = holt_winters(
```

```
    electricity_prices, best_alpha, best_beta, best_gamma,
    season_length, n_preds=2)
prediction_9 = forecasts[-2]  # 第9个月的预测值
prediction_10 = forecasts[-1]  # 第10个月的预测值(备用)
# 计算置信区间
confidence_level = 0.95
z_score = norm.ppf(1 - (1 - confidence_level) / 2)
# 使用历史误差的标准差作为估计
historical_errors = []
for t in range(season_length, len(electricity_prices)):
    historical_errors.append(forecasts[t] - electricity_prices
    [t])
std_error = np.std(historical_errors, ddof=1)
lower_9 = prediction_9 - z_score * std_error
upper_9 = prediction_9 + z_score * std_error
# 可视化
plt.figure(figsize=(12, 6))
plt.plot(range(1, len(electricity_prices)+1),
    electricity_prices, 'b-o', linewidth=2, label='实际值')
plt.plot(len(electricity_prices)+1, prediction_9, 'ro',
    markersize=8, label='第9个月预测值')
plt.plot([len(electricity_prices)+1, len(electricity_prices)
    +1], [lower_9, upper_9], 'r--', linewidth=2, label='95%置信
    区间')
# 添加预测点标记
plt.text(len(electricity_prices)+1.1, prediction_9, f'{
    prediction_9:.4f}', ha='left', va='center')
plt.xticks(range(1, len(electricity_prices)+2))  # 包含第9个月
plt.xlabel('月份')
plt.ylabel('电价')
plt.title('季节性电价预测 (Holt-Winters三参数指数平滑) - 预测
    第9个月')
plt.legend()
plt.grid(True)
plt.show()
```

```
# 输出结果
print('最优参数:')
print(f'alpha (水平平滑系数): {best_alpha:.2f}')
print(f'beta (趋势平滑系数): {best_beta:.2f}')
print(f'gamma (季节平滑系数): {best_gamma:.2f}')
print(f'第9个月预测值: {prediction_9:.4f}')
print(f'95%置信区间: [{lower_9:.4f}, {upper_9:.4f}]')

```

Gray Forcasting.ipynb

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import norm

# 设置字体和负号显示
plt.rcParams['font.sans-serif'] = ['SimHei']  # 设置字体为黑体
plt.rcParams['axes.unicode_minus'] = False  # 正确显示负号

# GM(1,1)模型函数
def gm11(x0, predict_step=1):
    n = len(x0)
    x1 = np.cumsum(x0)
    z1 = (x1[:-1] + x1[1:]) / 2.0
    B = np.column_stack((-z1, np.ones_like(z1)))
    Y = x0[1:].reshape(-1, 1)
    a, b = np.linalg.lstsq(B, Y, rcond=None)[0]
    # 计算拟合值
    fit_x1 = (x0[0] - b/a) * np.exp(-a * np.arange(n)) + b/a
    fit_x0 = np.zeros(n)
    fit_x0[0] = x0[0]
    for i in range(1, n):
        fit_x0[i] = fit_x1[i] - fit_x1[i-1]
    # 预测未来predict_step个点
    predict_x1 = (x0[0] - b/a) * np.exp(-a * np.arange(n +
  predict_step)) + b/a
```

```
    predict_x0 = np.zeros(n + predict_step)
    predict_x0[0] = x0[0]
    for i in range(1, n + predict_step):
        predict_x0[i] = predict_x1[i] - predict_x1[i-1]
    return fit_x0, predict_x0[-predict_step:]

# 第一段代码：季节性电价预测（灰色预测）
electricity_prices = np.array([0.6348, 0.6395, 0.6358, 0.5900,
    0.6115, 0.5894, 0.5971, 0.6088])
train_data = electricity_prices[:-1]
true_value = electricity_prices[-1]

# 使用GM(1,1)进行预测
fit_values, prediction = gm11(train_data, predict_step=1)
prediction_value = prediction[0]

# 计算残差和置信区间
residuals = train_data[1:] - fit_values[1:]
std_error = np.std(residuals, ddof=1)
z_score = norm.ppf(0.975)
lower = prediction_value - z_score * std_error
upper = prediction_value + z_score * std_error

# 绘制结果
plt.figure(figsize=(10, 6))
plt.plot(range(1, len(electricity_prices)+1),
   electricity_prices, 'b-', linewidth=2, label='实际值')
plt.plot(len(electricity_prices), prediction_value, 'ro',
   markersize=8, label='预测值')
plt.plot([len(electricity_prices), len(electricity_prices)], [
   lower, upper], 'r--', linewidth=2, label='95%置信区间')
plt.plot(len(electricity_prices), true_value, 'kx', markersize
   =8, label='真实值')
plt.text(len(electricity_prices)+0.1, prediction_value, f'{
   prediction_value:.4f}', ha='left', va='center')
```

```
plt.text(len(electricity_prices)+0.1, true_value, f'{
   true_value:.4f}', ha='left', va='center')
plt.xlabel('时间点')
plt.ylabel('电价')
plt.title('季节性电价预测（灰色预测）')
plt.legend()
plt.grid(True)
plt.show()

error = prediction_value - true_value
mse = error ** 2
mae = np.abs(error)
mape = np.abs(error / true_value) * 100

print('预测结果:')
print(f'预测值: {prediction_value:.4f}')
print(f'真实值: {true_value:.4f}')
print(f'95%置信区间: [{lower:.4f}, {upper:.4f}]')
print(f'均方误差 (MSE): {mse:.6f}')
print(f'平均绝对误差 (MAE): {mae:.6f}')
print(f'平均绝对百分比误差 (MAPE): {mape:.2f}%')

# 第二段代码：成品率预测（灰色预测）
data1 = np.array([
    [0.9359, 1.3584, 1.0913, 1.1832],
    [0.8237, 1.1957, 0.9510, 1.0277],
    [0.8234, 1.1918, 0.9533, 1.0316],
    [0.7559, 1.0924, 0.8727, 0.9432],
    [0.7267, 1.0495, 0.8376, 0.9047],
    [0.6864, 0.9915, 0.7884, 0.8502],
    [0.6733, 0.9719, 0.7729, 0.8333],
    [0.6207, 0.8962, 0.7087, 0.7622]
]) * 1e3  # 乘以 1e3 以匹配 MATLAB 的格式

data4 = np.array([
```

```
    [2.30, 2.11, 2.11, 2.10],  # 1月
    [2.03, 2.02, 2.15, 2.05],   # 2月
    [2.32, 1.90, 2.06, 1.76],   # 3月
    [2.37, 1.80, 1.85, 1.82],   # 4月
    [1.69, 1.93, 1.70, 1.50],   # 5月
    [1.75, 1.85, 1.75, 1.55],   # 6月
    [1.85, 1.70, 1.70, 1.55],   # 7月
    [1.71, 1.65, 1.60, 1.35]    # 8月
])
data = np.array([
    [110, 110, 110, 95],   # 1月
    [100, 100, 100, 85],   # 2月
    [89, 89, 89, 75],      # 3月
    [85, 85, 85, 72],      # 4月
    [85, 85, 85, 72],      # 5月
    [75, 75, 75, 60],      # 6月
    [70, 70, 70, 60],      # 7月
    [60, 60, 60, 50]       # 8月
])
data3 = np.array([
    [0.181, 0.181, 0.181, 0.203],  # 1月
    [0.181, 0.181, 0.181, 0.203],  # 2月
    [0.181, 0.181, 0.181, 0.203],  # 3月
    [0.181, 0.181, 0.181, 0.203],  # 4月
    [0.181, 0.181, 0.181, 0.203],  # 5月
    [0.180, 0.180, 0.180, 0.200],  # 6月
    [0.1795, 0.1795, 0.1795, 0.200],  # 7月
    [0.1795, 0.1795, 0.1795, 0.200]  # 8月
])
data2  = np.array([
    [98, 98, 98, 98],  # 1月
    [98.20, 98.20, 98.20, 98.20],  # 2月
    [98.10, 98.10, 98.10, 98.10],  # 3月
    [98.40, 98.40, 98.40, 98.40],  # 4月
    [98.50, 98.50, 98.50, 98.50],  # 5月
```

```
    [98.30, 98.30, 98.30, 98.30],  # 6月
    [98.50, 98.50, 98.50, 98.50],  # 7月
    [98.60, 98.60, 98.60, 98.60]   # 8月
])

true_values = data[-1, :]
predictions = []
confidence_intervals = []

for i in range(data.shape[1]):
    chip_data = data[:-1, i]
    fit_vals, pred = gm11(chip_data, predict_step=1)
    pred_value = pred[0]
    predictions.append(pred_value)
    residuals = chip_data[1:] - fit_vals[1:]
    std_err = np.std(residuals, ddof=1) if len(residuals) > 0
   else 0
    lower = pred_value - z_score * std_err
    upper = pred_value + z_score * std_err
    confidence_intervals.append([lower, upper])

predictions = np.array(predictions)
errors = predictions - true_values
mse = np.mean(errors ** 2)
mae = np.mean(np.abs(errors))
mape = np.mean(np.abs(errors / true_values)) * 100

# 绘制结果
fig, axes = plt.subplots(2, 2, figsize=(12, 10))
fig.suptitle('成品率预测（灰色预测）', fontsize=16)
colors = ['b', 'r', 'g', 'm']
for i in range(data.shape[1]):
    row = i // 2
    col = i % 2
    ax = axes[row, col]
```

```
    ax.plot(range(1, data.shape[0] + 1), data[:, i], colors[i], linewidth=1.5, label='实际值')
    ax.plot(data.shape[0], predictions[i], colors[i] + 'o', markersize=8, label='预测值')
    ax.plot([data.shape[0], data.shape[0]], confidence_intervals[i], colors[i] + '--', linewidth=2, label='95%置信区间')
    ax.plot(data.shape[0], true_values[i], 'kx', markersize=8, label='真实值')
    ax.set_xlabel('月份')
    ax.set_ylabel('价格')
    ax.set_title(f'芯片{i+1}成品率预测')
    ax.legend()
    ax.grid(True)
plt.tight_layout()
plt.show()

print('第八个月的预测值:')
print(predictions)
print('真实值:')
print(true_values)
print('95%置信区间:')
for i in range(data.shape[1]):
    print(f'芯片{i+1}: [{confidence_intervals[i][0]:.2f}, {confidence_intervals[i][1]:.2f}]')
print('均方误差 (MSE):', mse)
print('平均绝对误差 (MAE):', mae)
print('平均绝对百分比误差 (MAPE):', mape)

# 第三段代码：简单灰色预测
data = np.array([
    [8300000, 3300000, 6500000, 4150000],
    [6000000, 3550000, 5000000, 4300000],
    [5000000, 4000000, 5500000, 7500000],
    [3300000, 4500000, 7500000, 8500000],
```

```
    [8000000, 2500000, 7000000, 8000000],
    [7800000, 4100000, 4800000, 6500000],
    [7500000, 4200000, 4200000, 7050000],
    [7600000, 4500000, 8800000, 8250000]
])

true_values = data[-1, :]
predictions = []
confidence_intervals = []

for i in range(data.shape[1]):
    chip_data = data[:-1, i]
    fit_vals, pred = gm11(chip_data, predict_step=1)
    pred_value = pred[0]
    predictions.append(pred_value)
    residuals = chip_data[1:] - fit_vals[1:]
    std_err = np.std(residuals, ddof=1) if len(residuals) > 0
   else 0
    lower = pred_value - z_score * std_err
    upper = pred_value + z_score * std_err
    confidence_intervals.append([lower, upper])

predictions = np.array(predictions)
errors = predictions - true_values
mse = np.mean(errors ** 2)
mae = np.mean(np.abs(errors))
mape = np.mean(np.abs(errors / true_values)) * 100

print("第八个月的预测值:", predictions)
print("真实值:", true_values)
print("误差:", errors)
print("均方误差 (MSE):", mse)
print("平均绝对误差 (MAE):", mae)
print("平均绝对百分比误差 (MAPE):", mape)
print("95% 置信区间下限:", [ci[0] for ci in
```

```
    confidence_intervals])
220 print("95% 置信区间上限:", [ci[1] for ci in
    confidence_intervals])
221
```