****

**2024年江苏省研究生数学建模科研创新实践大赛**

**题 目 人造革性能优化设计研究**

摘 要：

本研究针对人造革性能优化设计进行了系统分析与建模，重点探讨了不同工艺参数对人造革多项性能指标的影响。创新之处在于综合运用多元线性回归模型、单目标优化及多目标优化方法，对树脂含量、固化温度和碱减量程度等工艺参数进行深入研究，以优化人造革的力学性能、热湿舒适性能及柔软性能。

首先，研究通过16组正交试验收集实验数据，并进行数据标准化处理。随后，通过皮尔逊相关系数分析和方差分析，明确了工艺参数与各性能指标之间的线性关系及其交互作用。结果显示，树脂含量、固化温度和碱减量程度对不同性能指标有显著影响，特别是碱减量程度对断裂强力和撕裂强力有显著负相关影响。在模型构建部分，采用多元线性回归模型，分别针对力学性能、热湿舒适性能和柔软性能建立单目标优化模型，并通过遗传算法进行了优化求解，得出了每个性能指标的最佳工艺参数组合。结果表明，最佳工艺参数组合在三个不同的性能指标中均表现出优异的性能。为了满足综合性能的需求，研究进一步采用多目标优化方法，构建了考虑多项性能指标的多目标优化模型。通过遗传算法求解，最终得出了兼顾多项性能指标的最优工艺参数组合：树脂含量为18.28%、固化温度为121.18℃、碱减量程度为16.11%。该组合在力学性能、热湿舒适性能和柔软性能上均表现出色。针对沙漠科考队的特殊需求，研究通过多目标约束优化方法，设计了优先满足透气率和透湿率，同时优化力学性能和柔软性能的最佳人造革工艺参数组合：树脂含量23wt%、固化温度116℃、碱减量程度16%。这一组合在兼顾多项性能指标的基础上，确保了人造革在极端环境下的综合性能。

本研究的结果为人造革工艺参数优化提供了科学依据，通过模型预测和优化，显著提高了人造革的综合性能。研究结论不仅有助于提升人造革产品质量，还为相关领域的工艺参数优化提供了有力支持，具有重要的实际应用价值。

**关键词：**工艺参数，力学性能，热湿舒适性，柔软性能，多目标优化

# 目 录

一、问题重述 3

1.1 问题背景 3

1.2 问题提出 3

二、模型假设与符号说明 4

2.1 基本假设 4

2.2 符号说明 4

三、数据预处理 5

3.1 数据处理与评估 5

3.2 数据标准化 7

四、模型建立与求解 9

4.1 问题1的模型建立与求解 9

4.1.1 相关性分析 9

4.1.2 交互作用分析 10

4.2 问题2的模型建立与求解 12

4.2.1 多元线性回归模型 12

4.2.2 建模理由 13

4.3 问题3的模型建立与求解 14

4.3.1 单目标优化 14

4.3.2 多目标优化 15

4.3.3 结果比较 16

4.4 问题4的模型建立与求解 17

4.4.1 多目标约束优化 17

4.4.2 结果分析与建议 17

五、模型评价与推广 19

5.1 模型优点 19

5.2 模型的缺点 19

5.3 模型的改进与推广 19

六、参考文献 21

七、附录 22

# 一、问题重述

## 1.1 问题背景

人造革作为一种由高分子材料（树脂）制成的纤维状物质，通过特定的工艺处理，具有类似皮革的质感和外观，同时兼具高强度、高耐磨和柔软的特点。因此，人造革在服装等领域得到广泛应用。然而，生产高质量人造革的关键在于控制工艺参数，如树脂含量、固化温度和碱减量程度等，这些参数直接影响人造革的力学性能、热湿舒适性能和柔软性能。为了优化人造革的各项性能，生产者基于生产经验开展了针对性的实验，通过定型、树脂含浸、碱减量等工序制备人造革，并对其进行多种性能测试，从而找到最佳的工艺参数组合。

## 1.2 问题提出

在本研究中，为了系统地优化人造革的性能，研究者们需要深入分析工艺参数与人造革各项性能之间的关系，以及不同性能指标之间的相互影响。同时，由于实际生产中往往需要根据不同需求优化多项性能指标，研究者需要建立数学模型，分别分析追求最佳力学性能、最佳热湿舒适性和最佳柔软性能所需的工艺参数，并在此基础上综合考虑这些性能指标，以找到整体性能最优的工艺参数组合。特别是针对某沙漠科考队提出的特殊需求，人造革需优先满足热湿舒适性，同时还需兼顾力学性能和柔软性能。因此，本研究旨在通过系统的实验和建模分析，提出最优的工艺参数方案，以满足不同应用场景下的人造革性能需求。为了优化人造革的性能，本研究提出了以下具体问题：首先，需要分析工艺参数（如树脂含量、固化温度和碱减量程度）与人造革产品性能（包括断裂强力、断裂伸长率、撕裂强力、透气率、透湿率、柔软度和折皱回复角）之间的关系，揭示其相互影响的程度。其次，在此基础上，建立工艺参数与这七种性能指标之间的关系模型，确定每种性能的最佳工艺参数组合。接下来，通过数学建模，分别分析追求最优力学性能、热湿舒适性、柔软性能及综合性能所需的工艺参数，并对这些优化结果进行比较。最后，针对某沙漠科考队的特殊需求，设计出优先满足热湿舒适性，同时兼顾力学性能和柔软性能的最佳人造革，以满足其在极端环境下的使用要求。

# 二、模型假设与符号说明

## 2.1 基本假设

在建立模型时，在本研究中根据所给的条件进行了以下四个假设：

假设 1：附件给的数据真实可靠，所收集的数据也是真实可靠的；

假设 2：评审专家对实验数据的评价是客观、准确的；

假设 3：不同工艺参数组合对性能的影响具有独立性和可加性；

假设 4：在优化过程中，不改变现有资源的约束条件。

## 2.2 符号说明

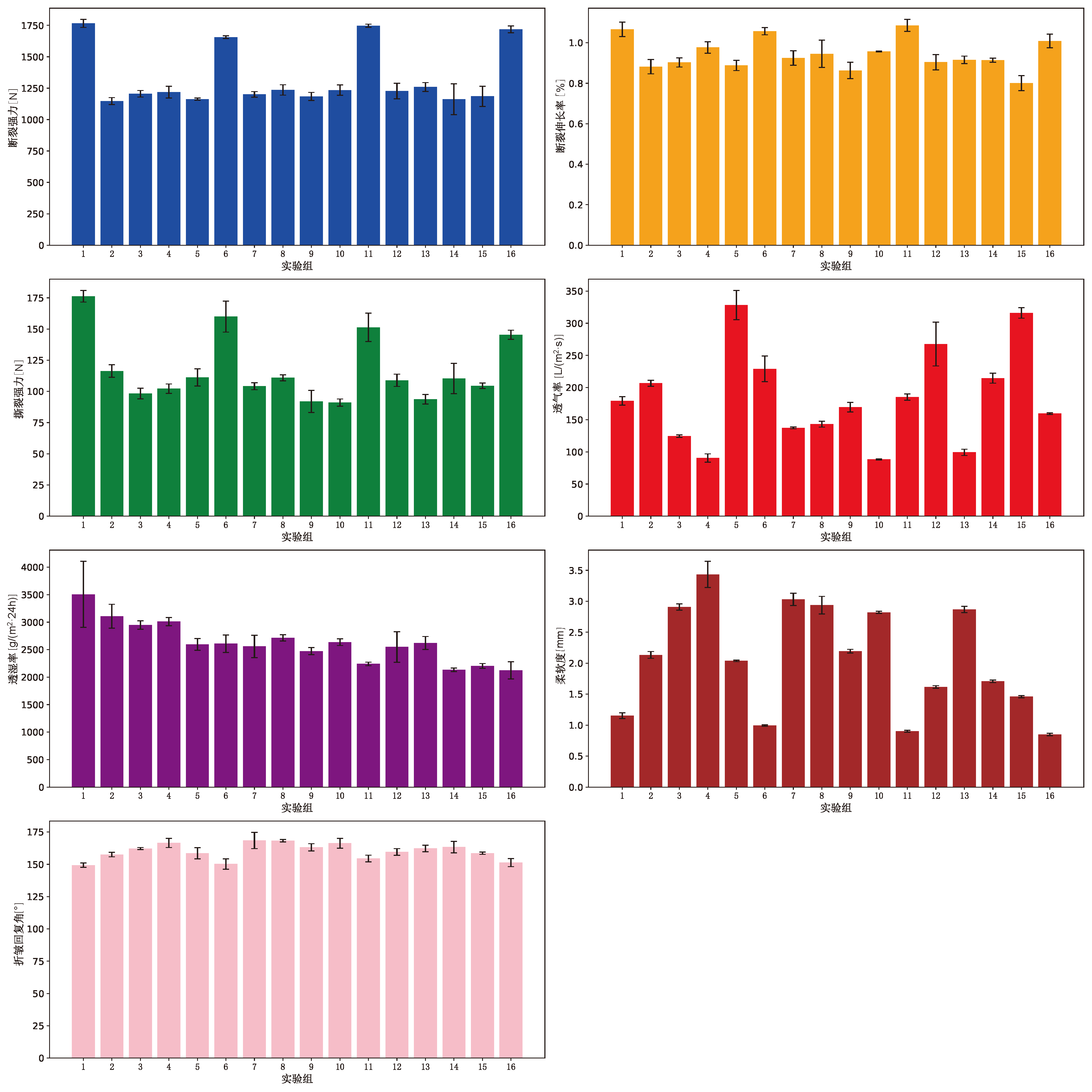
表2-1符号说明

|  |  |
| --- | --- |
| **符号** | **含义** |
| Xi | 工艺参数（i=1,2,3，分别代表树脂含量、固化温度和碱减量程度） |
| Yi | 性能指标（i=1,2,…,7，分别代表断裂强力、断裂伸长率、撕裂强力、透气率、透湿率、柔软度和折皱回复角性能） |

# 三、数据预处理

## 3.1 数据处理与评估

为了便于数据处理，我们对原始文件中的数据进行了预处理。首先，计算了不同实验组重复实验的平均值，并进行了误差分析。其次，将每组实验的工艺参数与计算的平均值汇总在一个表格中，具体数据附录。最后，我们将实验结果与基布数据进行了比较，初步确定了工艺参数（树脂含量、固化温度、碱减量程度）与各性能指标（断裂强力、断裂伸长率、撕裂强力、透气率、透湿率、柔软度、折皱回复角）之间的关系。

图3-1不同工艺参数下人造革性能指标的直方图及误差分析

从图3-1中可以看出，不同工艺参数（如树脂含量、固化温度和碱减量程度）对人造革的各项性能指标（如断裂强力、断裂伸长率、撕裂强力、透气率、透湿率、柔软度和折皱回复角）有显著影响。每组实验的平均值用柱状图表示，误差棒表示标准差，以反映每组实验的重复性和数据的离散程度。断裂强力的各组实验的平均值在一定范围内波动，且标准差较大，说明实验数据的离散程度较高，这可能是由于不同工艺参数对断裂强力影响较大所致。断裂伸长率该指标变化的整体趋势较为一致，标准差较小，说明数据的重复性较好。不同工艺参数对断裂伸长率的影响相对较小。撕裂强力的波动较大，标准差也较大，表明实验结果的离散程度较高，这可能是由于工艺参数对撕裂强力的影响较为复杂。实验组之间透气率的平均值变化明显，标准差相对较小，说明数据的重复性较好。工艺参数对透气率有显著影响，且变化较为规律。透湿率在各组实验中的平均值和标准差均有一定波动，表明工艺参数对透湿率有一定影响，且数据的离散程度较高。柔软度在不同实验组中的平均值和标准差都有明显变化，说明工艺参数对柔软度的影响较大，且数据的离散程度较高。折皱回复角在各组实验的平均值相对稳定，标准差较小，表明数据的重复性较好。工艺参数对折皱回复角的影响较小。通过上述分析表明，不同工艺参数对人造革性能指标的影响各异，部分性能指标对工艺参数的敏感性较高，而部分则较为稳定。

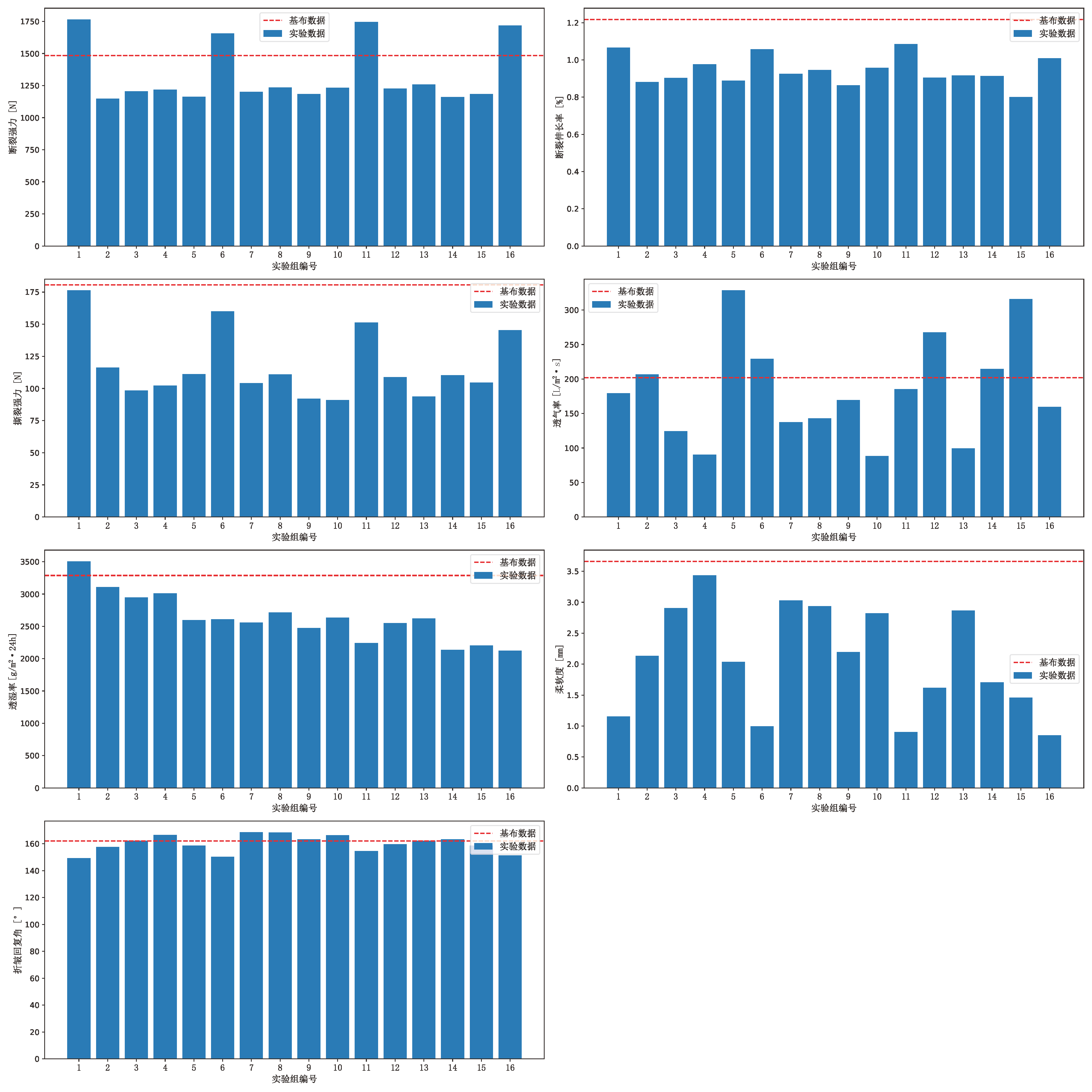


图3-2 16组正交试验不同性能指标与基布性能数据对比图

图3-2展示了16组正交实验在七个不同性能指标下的结果，并与基布性能数据进行了对比。接下来根据上图对每个性能指标进行具体分析：

（1）断裂强力：根据图中显示，第1、2、3、7、10和16组实验的断裂强力显著高于基布数据，显示了不同工艺参数对材料强度的显著影响。

（2）断裂伸长率：大多数实验组的断裂伸长率与基布数据相比都有所提高，尤其是第3、4、8、9、11和12组。这表明在这些实验条件下，材料的伸展性能得到了增强。

（3）撕裂强力：实验结果显示，第3、6、8、9、11和15组的撕裂强力高于基布数据，而其他组则相对较低。这种变化表明，不同工艺参数组合对撕裂强度有显著影响，有些组合显著增强了材料的撕裂性能。

（4）透气率 ：通过上图中可以看出，第1、2、3、5、6、7、10和14组的透气率高于基布数据。第4组实验显示出最高的透气率，表明该工艺参数组合在提高透气性方面最为有效。

（5）透湿率：多数实验组的透湿率低于基布数据，仅第4、6、10和15组略高。这表明多数工艺参数组合对材料的透湿性能没有显著改善，甚至有所降低。

（6）柔软度：大部分实验组的柔软度接近基布数据，但第1、3、4、5、7、11和15组的柔软度略高。显示了某些工艺参数组合对材料柔软度的改善效果。

（7）折皱回复角：除第1、3、4、6、8、10和12组实验外，其他组的折皱回复角均低于基布数据。这表明这些实验组在抗皱性能方面存在显著不足。

综合以上分析可以看出，不同的工艺参数组合对人造革的各项性能指标有显著影响。某些组合在增强断裂强力和撕裂强力方面表现出色，而另一些组合则在提高透气率和柔软度上更为有效。为了获得最优的综合性能，可能需要进一步优化这些工艺参数组合。

## 3.2 数据标准化

数据标准化是数据分析中的一种重要预处理步骤，它通过将数据转换为统一的尺度，消除了不同量纲和数值范围带来的影响，从而提高分析结果的准确性和可靠性。以下是数据标准化在相关性分析、方差分析和多元线性回归模型中的作用以及应用案例的介绍。在相关性分析中，数据标准化确保了不同变量之间的比较是公平的。由于标准化处理消除了变量间量纲和数值范围的差异，因此可以更准确地评估变量间的线性关系。例如，在使用皮尔逊相关系数进行分析时，数据标准化可以避免由于某些变量数值范围较大而对相关系数产生不成比例的影响。方差分析（ANOVA）用于检验不同组间均值是否存在显著差异。数据标准化有助于满足方差分析的假设条件，特别是方差齐性（homoscedasticity）。通过将数据转换到均值为0、标准差为1的尺度，可以确保不同组的数据具有相似的方差，从而使得ANOVA的检验结果更为可靠。在多元线性回归模型中，数据标准化不仅有助于提高模型的稳定性和精度，还可以避免因变量量纲不一致而导致的系数估计偏差。标准化后的数据使得不同自变量的系数可比，便于解释模型结果。此外，标准化可以减少共线性问题，提高模型的预测能力。

我们采用 Z-score 标准化方式（也称为标准化分数或零均值单位方差标准化）。Z-score 标准化方式这种方法使得标准化后的数据具有均值为0，标准差为1的特性。对于一个数据点 ，其标准化后的 分数计算公式如下：

式中，为均值，为标准差。

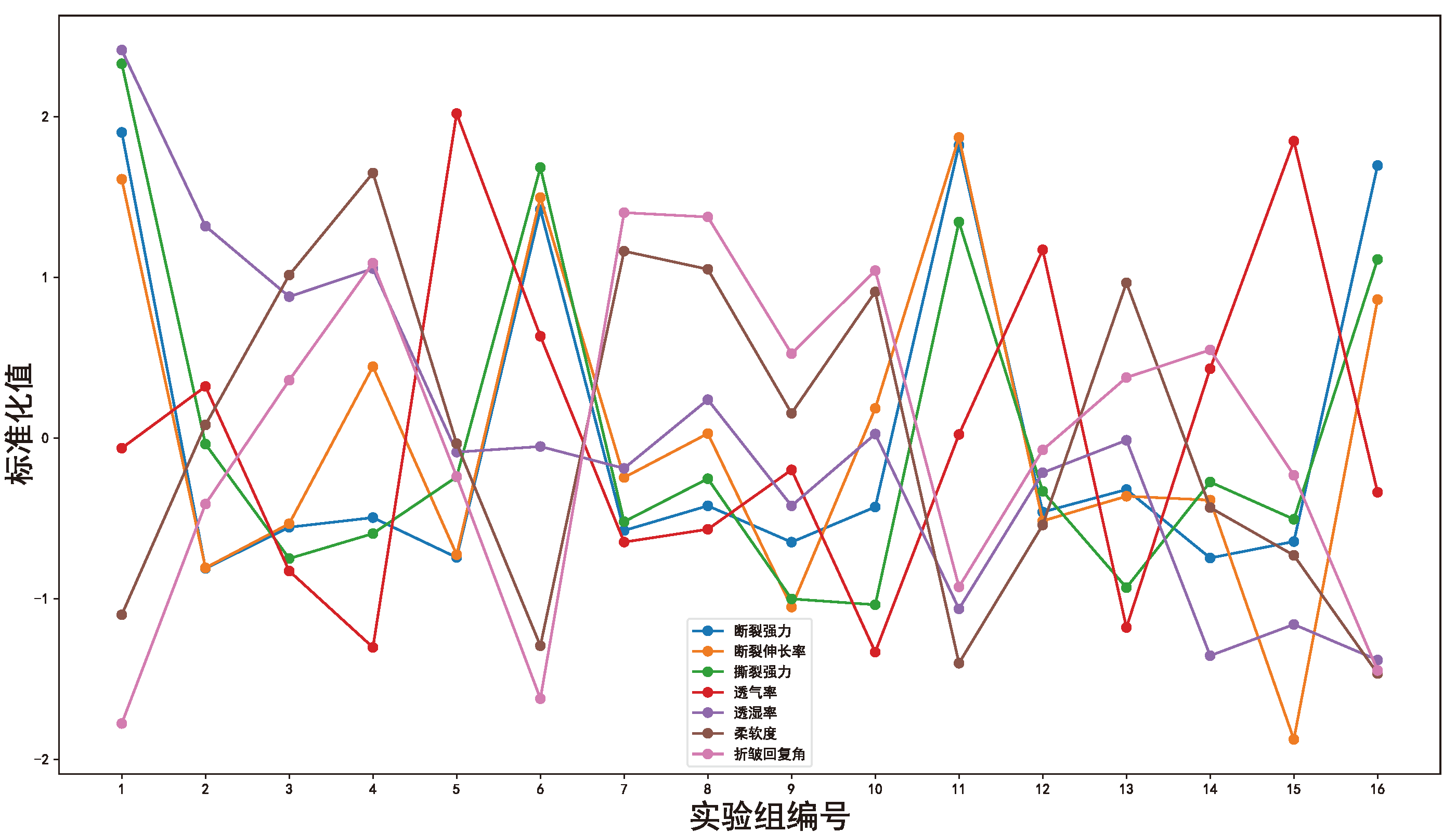


图3-3 标准化后的实验组性能数据

通过数据的标准化，能够更容易地识别每个实验组在各个性能指标上的相对表现，依据图3-3对每个性能指标进行具体分析。首先对断裂强力这个指标进行分析，通过上图不难发现实验组1、6、11和16表现突出，显著高于平均水平，实验组2、3、4、5、7、8、9、10、12、13、14和15的断裂强力低于平均水平。随后对断裂伸长率这个指标进行分析，通过上图发现实验组1、6、11在断裂伸长率上表现最好，实验组9和15表现较差，且低于平均水平。再者对撕裂强力这个指标进行分析，发现实验组1、6、11和16的撕裂强力显著高于其他组，实验组3、9、10和13的撕裂强力较低。对透气率这个指标进行分析，发现实验组5、12和15在透气率上表现优异，实验组4、10和13的透气率较低。接下来对透湿率这个指标进行分析，通过上图发现实验组1、2和4在透湿率上有明显的优势，实验组14、15和16的透湿率较低。对柔软度这个指标进行分析，发现实验组4、7和8在柔软度上表现较好，实验组1、6、11和16表现较差。最后，对折皱回复角这个指标进行分析，不难发现实验组7、8、10和6在折皱回复角上表现最优，实验组1、6和16的折皱回复角较低。

综合以上分析，各实验组在不同性能指标上表现出明显的多样性，没有单一的实验组在所有指标上均表现最佳。实验组6和11在多个指标上表现较好，可能是优选的工艺参数组合。

# 四、模型建立与求解

## 4.1 问题1的模型建立与求解

### 4.1.1 相关性分析

皮尔逊相关系数（Pearson correlation coefficient）由著名统计学家卡尔·皮尔逊提出，该系数主要用于衡量两个变量之间的线性相关性。如果变量X和Y之间的变化趋势一致，即一个变量增加时另一个变量也增加，或者一个变量减少时另一个变量也减少，则认为它们之间存在正相关。反之，如果一个变量增加而另一个变量减少，则它们之间存在负相关。如果变量之间没有明显的线性关系，则相关系数接近0。在自然科学领域中，皮尔逊相关系数广泛用于度量两个变量之间的相关程度，其值介于-1与1之间。它是由卡尔·皮尔逊从弗朗西斯·高尔顿在19世纪80年代提出的一个相似却又稍有不同的想法演变而来的，这个相关系数也称作“皮尔逊积矩相关系数”。

（1）工艺参数与各性能指标之间相关性分析

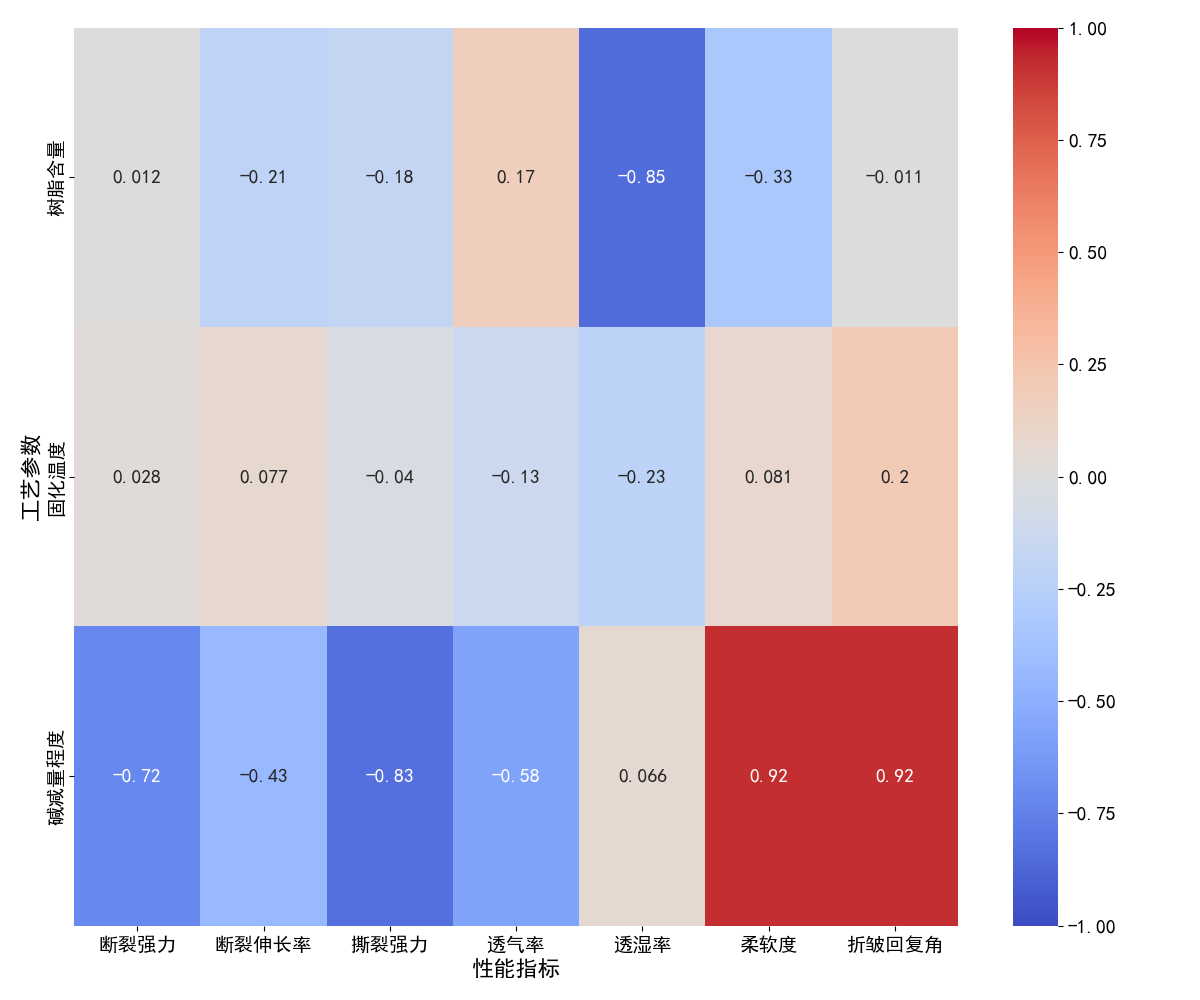


图4-1工艺参数与各性能指标之间Pearson相关系数

首先，我们使用Pearson相关系数分析工艺参数与各性能指标之间的关系。图4-1展示了工艺参数与各性能指标之间Pearson相关系数，从图4-1中可以看出，工艺参数与性能指标之间存在以下关系：固化温度与各性能指标的相关性都很弱，说明固化温度对这些性能指标的影响不显著。树脂含量与透湿率和柔软度有显著的负相关性，树脂含量越高，透湿率和柔软度越低，其他性能指标与树脂含量的相关性较弱。碱减量对大多数性能指标有显著影响，特别是对断裂强力、撕裂强力、透气率有显著的负相关性，而对柔软度和折皱回复角有显著的正相关性，表明碱减量增加会降低强力和透气性，但提高柔软度和折皱回复能力。

基于上述结论，我们提供了如下建议：如果目标是提高断裂强力和撕裂强力，应减少碱减量；如果目标是提高柔软度和折皱回复角，应增加碱减量；对透湿率的影响较大的是树脂含量，应控制树脂含量以优化透湿率；固化温度的调整对各性能指标影响不大，可考虑在其他参数上进行优化。

（2）不同性能指标之间的相关性分析

接下来使用Pearson相关系数来分析不同性能指标之间的关系。如图4-2展示了各性能指标之间Pearson相关系数，从图4-2中可以看出各性能指标之间的Pearson相关系数，某些性能指标之间存在显著的正相关关系或负相关关系。例如，断裂强力和撕裂强力之间存在很强的正相关关系，而断裂强力和柔软度之间则存在很强的负相关关系。这些相关性提供了有价值的信息，可以帮助进行优化不同工艺参数，以实现特定的性能目标。

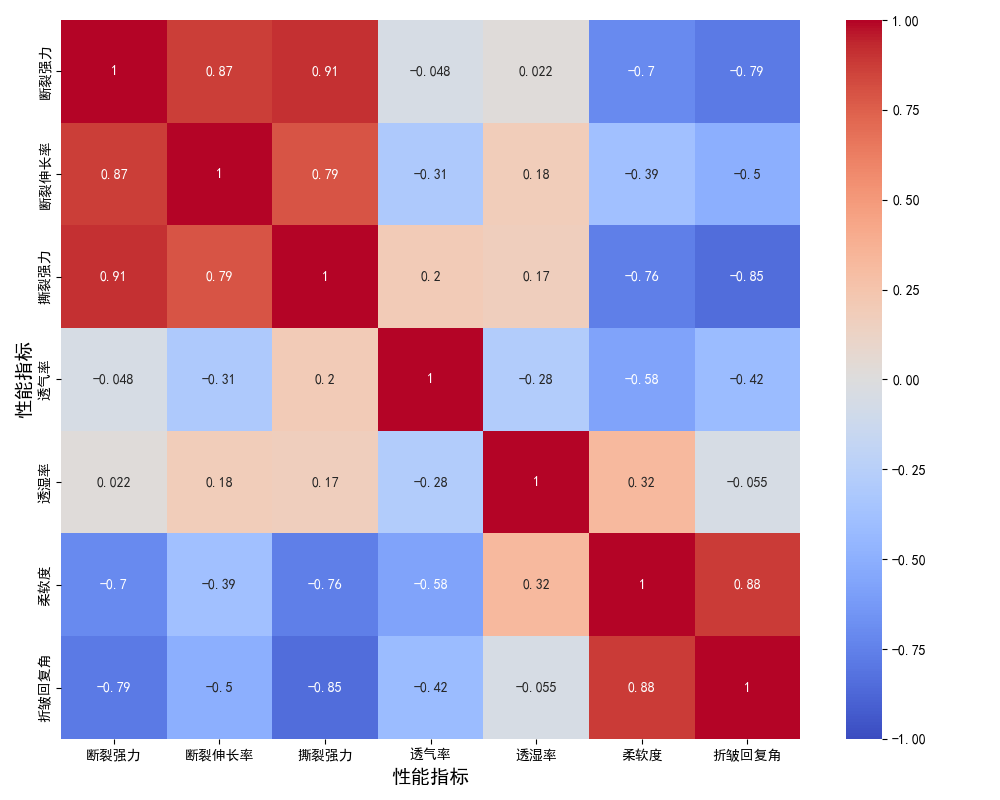


图4-2各性能指标之间Pearson相关系数

### 4.1.2 交互作用分析

方差分析（ANOVA）是一种统计技术，又称“变异数分析”，是R.A.Fisher发明的，用于两个及两个以上样本均数差别的显著性检验。由于各种因素的影响，研究所得的数据呈现波动状。造成波动的原因可分成两类，一是不可控的随机因素，另一是研究中施加的对结果形成影响的可控因素。在本文中方差分析用于检验三个或更多独立组的平均值之间是否存在显著差异，它可以帮助研究者了解不同组之间的变量是否对结果变量有显著影响。方差分析可以按照因素的数量分为单因素方差分析（One-Way ANOVA）和多因素方差分析（如Two-Way ANOVA）。单因素方差分析是用于分析一个自变量在多个水平上对一个连续因变量的影响，而多因素方差分析是用于分析两个或两个以上自变量及其交互作用对因变量的影响。

本文采用多因素方差分析来评估工艺参数之间的交互作用对性能的影响。首先，我们采用如表3-1所示经过Z-score 标准化处理之后的数据，使用普通最小二乘法（OLS）模型来拟合数据。模型公式通常表示为性能指标作为因变量，而工艺参数及其交互作用作为自变量。例如，对于三个工艺参数A、B和C，模型公式可以写作 Performance ~ A \* B \* C。

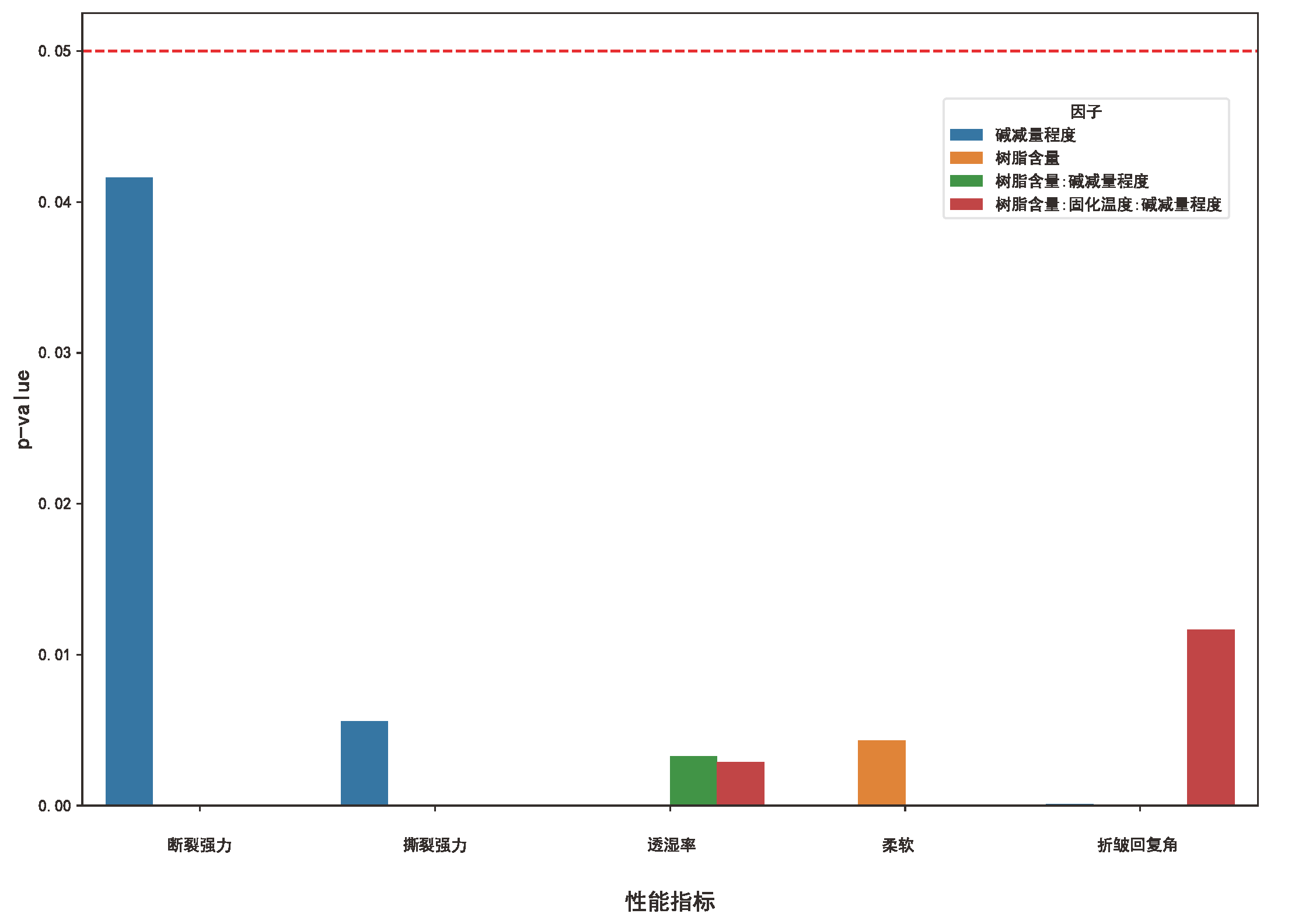


图4-3方差分析评估工艺参数之间的交互作用对性能的影响结果

接下来我们进行方差分析的计算。首先使用statsmodels库中的anova\_lm函数进行方差分析，得到各因子的 F值和 p值。F值用于评估各因子对性能指标的影响大小，而p值用于评估该影响的显著性。通常，当p值小于0.05时，我们认为该因子的影响是显著的。下一步是将每个性能指标对应的方差分析结果汇总到一个数据框中，记录一下各工艺参数及其交互作用对性能指标的影响程度和显著性，然后过滤出显著性结果，即p值小于 0.05 的部分进行进一步分析。再者是使用可视化工具（如 Matplotlib 和 Seaborn）将显著结果绘制成图表，便于直观地观察各工艺参数及其交互作用对不同性能指标的影响。通过条形图展示，每个条形表示一个显著的交互作用及其p值，并在图中标出显著性水平。

方差分析的结果如图4-3所示，根据上图可以总结和分析出工艺参数之间的交互作用对各性能指标的影响，即碱减量程度显著影响了断裂强力和撕裂强力，树脂含量显著影响了柔软度。树脂含量和碱减量程度的交互作用显著影响了透湿率，树脂含量、固化温度和碱减量程度的三重交互作用显著影响了折皱回复角和透湿率。

## 4.2 问题2的模型建立与求解

### 4.2.1 多元线性回归模型

我们对每个性能指标分别建立如下公式所示的回归模型：

式中，为性能指标，, , 分别为树脂含量、固化温度和碱减量程度。我们使用多元线性回归模型的结果，通过优化方法找到最佳工艺参数组合，具体步骤如下：

（1）数据准备

首先，读取实验数据并去除不必要的行（如单位行等），确保数据的有效性。将工艺参数和性能指标的列转换为数值类型，以确保数据处理和分析的正确性。

（2）提取工艺参数和性能指标

从数据集中分离出工艺参数（如树脂含量、固化温度和碱减量程度）和性能指标（如断裂强力、撕裂强力等）。

（3）定义目标函数

目标函数用于优化过程，在这里目标函数的目的是最小化多元线性回归模型的调整后R平方值（R²\_adj）的负值，以最大化模型的拟合度。在目标函数中，使用指定的工艺参数组合拟合多元线性回归模型，并返回调整后的R平方值的负值。

（4）初始猜测值和边界条件

选择工艺参数的平均值作为优化算法的初始猜测值。定义每个工艺参数的上下界，以确保优化结果在合理范围内。

（5）优化过程

对每个性能指标，构建相应的多元线性回归模型，公式中包括所有的工艺参数及其交互作用项。使用L-BFGS-B优化算法最小化目标函数，以找到使模型拟合度最高的工艺参数组合。

（6）存储和输出优化结果

将每个性能指标对应的最佳工艺参数组合存储到一个数据框中。将最终的最佳工艺参数组合导出到Excel文件中，以便进一步分析和使用。

（7）结果可视化

我们通过散点图可视化每个性能指标的拟合结果，将实际值与模型拟合值进行比较，保存可视化图表以便查看优化结果的直观效果。

通过多元线性回归模型分析，研究不同工艺参数（树脂含量、固化温度、碱减量程度）对各性能指标（断裂强力、撕裂强力、透气率、透湿率、柔软度、折皱回复角等）的影响。通过优化方法（L-BFGS-B）找到了每个性能指标对应的最佳工艺参数组合。所有性能指标的最佳工艺参数组合为：树脂含量：23wt%，固化温度：116℃，碱减量程度：16%。

如下图4-4展示了各性能指标最佳工艺参数组合的拟合效果，每个子图展示了模型的拟合值与实际值的比较，红色虚线表示理想情况下拟合值等于实际值的情况。断裂强力、撕裂强力、透气率、透湿率、柔软度和折皱回复角等性能指标的拟合效果良好，拟合值与实际值基本一致，说明模型具有较好的拟合度。断裂强力和撕裂强力的拟合值分布较为集中，但部分点与理想线有一定偏离，说明在这些指标上，模型还可以进一步优化。透湿率、透气率、柔软度和折皱回复角的拟合效果较好，拟合值与实际值之间的偏差较小，说明模型在这些指标上的表现较为准确。

通过多元线性回归模型，能够较好地拟合各性能指标与工艺参数之间的关系。最佳工艺参数组合（树脂含量23wt%、固化温度116℃、碱减量程度16%）能够同时优化多个性能指标，表明这组工艺参数在实验范围内具有较好的优化效果。我们通过使用L-BFGS-B优化算法，能够有效地找到工艺参数的最佳组合，说明该方法适用于类似的工艺参数优化问题。尽管模型在大多数性能指标上表现良好，但某些指标上存在一定的偏差，未来可以考虑引入更多的变量或采用其他更复杂的模型（如非线性回归模型）进一步提高拟合度。

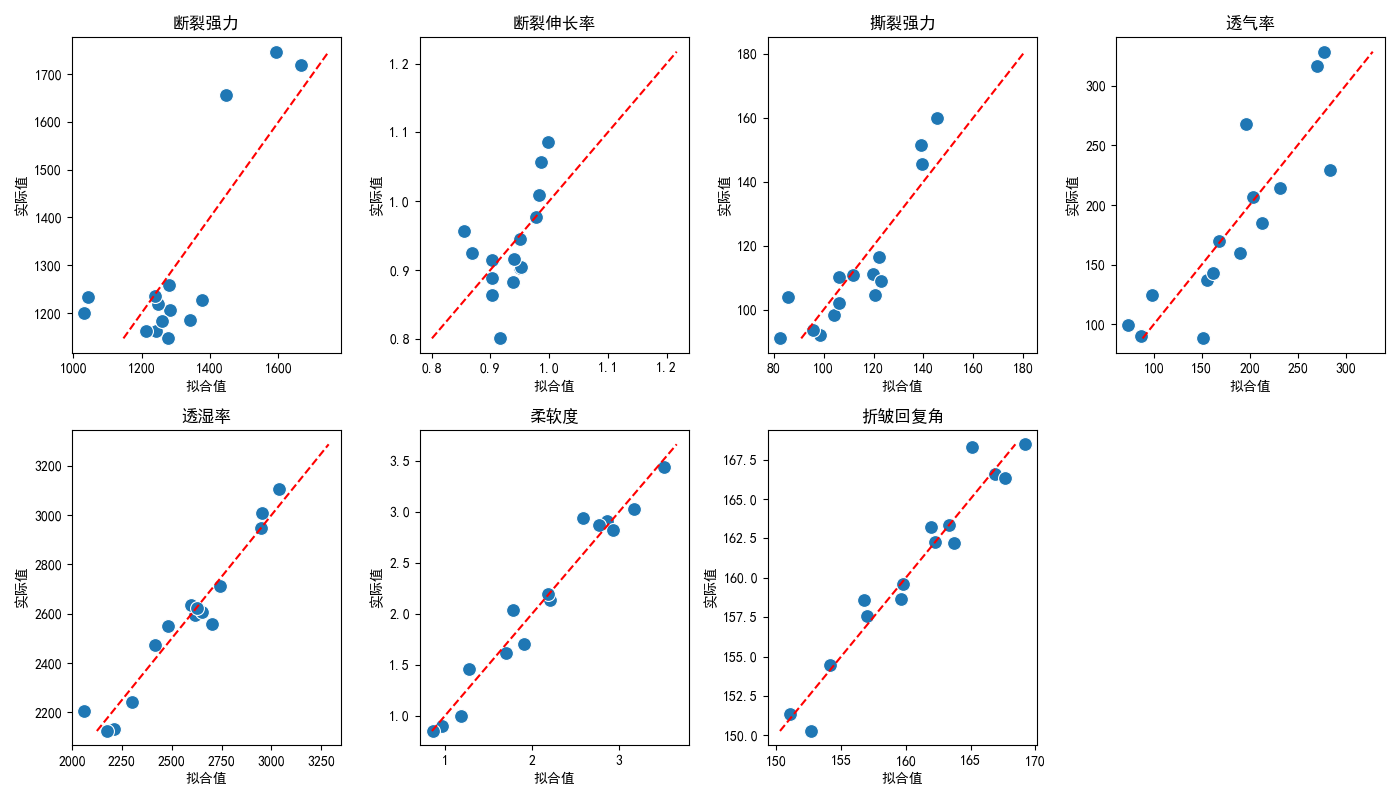


图4-4 各性能指标最佳工艺参数组合的拟合效果图

### 4.2.2 建模理由

（1）多元线性回归的适用性

多元线性回归是一种常用且强大的统计方法，适用于分析多个自变量（工艺参数）对一个因变量（性能指标）的影响。它假设自变量和因变量之间存在线性关系，能够有效地捕捉复杂关系，因而在许多研究和应用中得到了广泛使用。其优势不仅在于方法的简单性和强大性，还在于其较高的解释性：通过回归系数，研究者可以明确理解每个工艺参数对性能指标的独立贡献。此外，多元线性回归模型可以进行统计显著性检验，通过p值筛选出对因变量有显著影响的自变量，帮助优化工艺参数组合，找到关键影响因素。尽管线性回归在大样本情况下表现更佳，它在小样本数据下（如本研究中的16组实验数据）依然适用，尤其是在进行数据标准化处理后，线性回归能够提高模型的稳健性和可靠性，提供有价值的分析结果。

（2）交互作用分析的必要性

交互作用分析在多元线性回归模型中具有重要的必要性，能够揭示工艺参数之间的复杂关系。工艺参数间可能存在交互作用，即一个参数对性能指标的影响会因另一个参数的变化而不同。通过交互作用分析，可以捕捉到这些细微而重要的关系，而单纯的主效应分析往往无法揭示这些信息。在实际生产中，工艺参数通常不是孤立作用的，理解这些交互作用有助于优化工艺组合，提升产品性能。此外，忽略交互作用可能导致模型偏差，低估或高估某些工艺参数的影响，从而影响预测和优化的准确性。通过考虑交互作用，可以提高模型的拟合度和预测精度，使得优化结果更加可靠和具有实际应用价值。因此，交互作用分析不仅有助于更全面地理解工艺参数对性能指标的综合影响，还能找到最优的工艺参数组合，为生产工艺的优化提供科学依据。

## 4.3 问题3的模型建立与求解

### 4.3.1 单目标优化

为了追求最优的力学性能、热湿舒适性和柔软性能，建立了单目标优化模型，并使用Pareto优化方法分别分析了这些工艺参数的影响。图4.5展示了通过单目标优化模型找到的最佳工艺参数组合对力学性能（断裂强力）、热湿舒适性（透湿率）和柔软性能（柔软度）的影响。通过该模型，得出的最佳工艺参数组合为：树脂含量23wt%、固化温度116℃、碱减量程度16%。这一组合在三个不同的性能指标中均表现出最优的性能。

从拟合值与实际值的散点图可以看出，模型在透湿率和柔软度上的预测精度较高，而在断裂强力上的预测相对较弱。整体而言，单目标优化模型对各性能指标的拟合效果良好。优化结果显示，特定的工艺参数组合（树脂含量23wt%、固化温度116℃、碱减量程度16%）在三个不同的性能指标上均达到最优。这表明这些工艺参数对性能有显著影响，并且在实际生产中可以作为参考参数进行调整。尽管此次分析采用了单目标优化方法，结果显示相同的工艺参数组合对多种性能指标均具有优化效果，为未来进行多目标优化提供了重要参考。

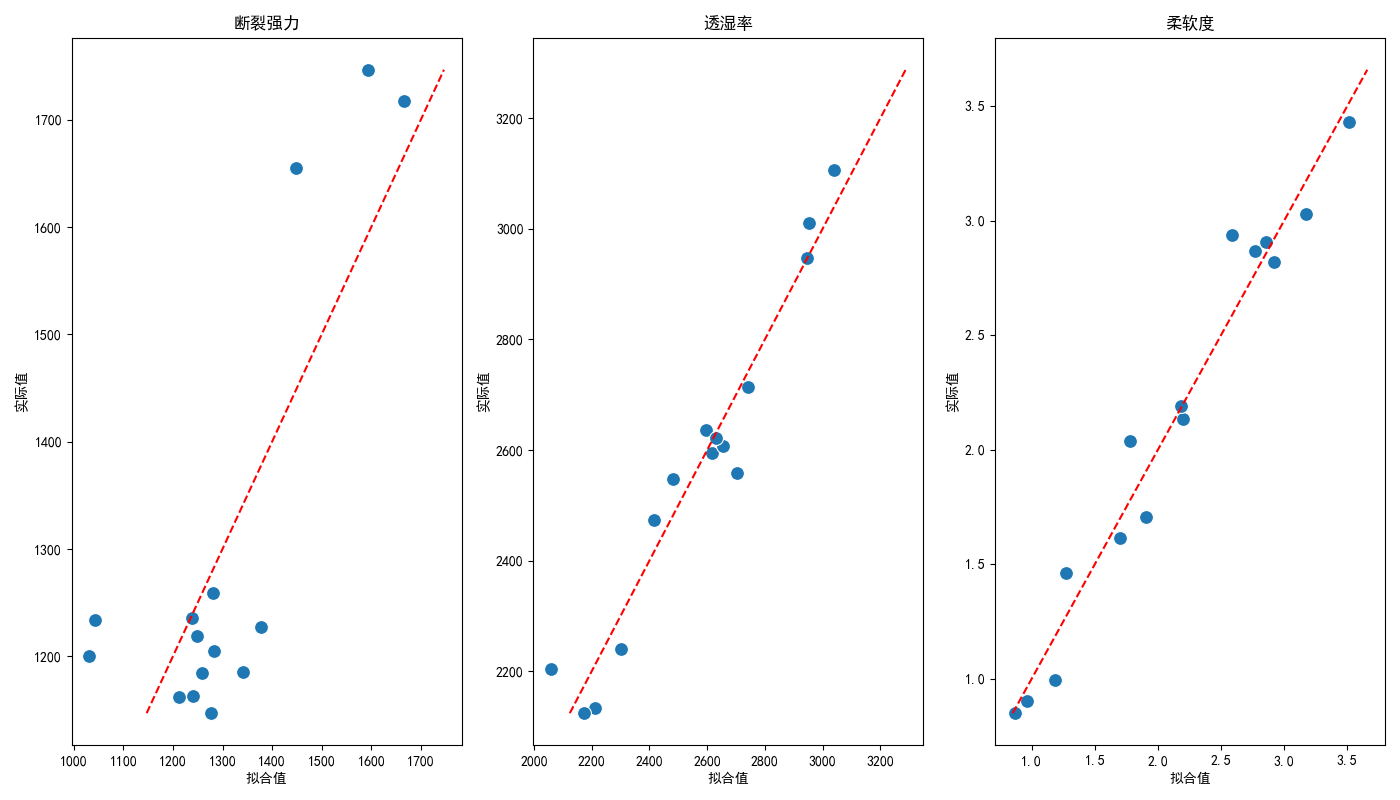


图4-5力学性能、热湿舒适性和柔软性能单目标优化结果

### 4.3.2 多目标优化

使用多目标优化方法，综合考虑各性能指标，建立如下所示多目标优化模型：

式中，, ,⋯, 为7项性能指标，, ,⋯, 为各指标的权重。具体步骤如下：

（1）数据预处理

从Excel文件中读取数据，去掉前两行非数据行并重置索引，将所有数据转换为数值类型以确保计算的准确性。

（2）提取工艺参数和性能指标

提取数据中的工艺参数列（树脂含量、固化温度、碱减量程度）；提取数据中的性能指标列（断裂强力、透湿率、柔软度等）。

（3）设置权重

为每个性能指标设置权重，这里假设每个指标的权重相等（均为1）。

（4）遗传算法配置

创建优化问题的Fitness和Individual类；注册生成基因、个体和种群的函数。

（5）定义目标函数以及参数设置

对于给定的个体，复制数据并将工艺参数设置为个体的值，然后使用多元线性回归模型拟合每个性能指标，并计算预测值的倒数作为逆性能值，最后计算加权和作为目标函数值。

注册遗传算法的操作：交叉、变异和选择。设置遗传算法的参数：种群大小为100，代数为100。

（6）运行遗传算法

创建初始种群并运行遗传算法，记录日志，选择最优个体并提取其工艺参数。

（7）输出结果

创建DataFrame存储最优工艺参数组合并输出到Excel文件。通过多目标遗传算法优化，综合考虑断裂强力、透湿率和柔软度等性能指标，最终得到了最优的工艺参数组合：树脂含量为18.28%、固化温度为121.18℃、碱减量程度为16.11%。这些参数组合在保证非织造布具有良好力学性能的同时，提升了其热湿舒适性和柔软性，充分体现了多目标遗传算法在复杂优化问题中的有效性和实用性。

### 4.3.3 结果比较

单目标优化所得成果展示如下，使用以下工艺参数，树脂含量为23%重量百分比 (wt%)，固化温度设定为116°C，碱减量程度控制在16%。优势体现为单目标优化着眼于单个关键性能衡量标准（如断裂强度）的改进，可针对此目标取得最佳表现；与多目标相比，优化流程相对简洁明了，计算所需成本较低，便于理解和操作；由于只关注一个目标，算法的收敛速度明显加快。然而，其不足之处也同样明显：忽视了其他性能指标的优化，有可能导致其他性能指标的降低或者无法达到预期水平；适合那些对某个特定性能指标有着极高要求，而对其他指标要求相对较低的应用环境。

多目标优化所得成果展示如下，使用以下工艺参数，树脂含量为18.28%重量百分比(wt%)，固化温度设定为121.18°C，碱减量程度控制在16.11%。优势体现为：多目标优化充分考虑到多个性能指标（如断裂强度、透气性、柔软度等），能在各个指标间寻找到平衡点；对于需要全方位提高各种性能指标的应用场景，多目标优化能够满足这些需求，进一步拓展实际应用领域；多目标优化可以得出一系列非劣解集（帕累托前沿），为用户提供多样化的优化选项。然而，其不足之处也同样明显：优化过程繁琐，计算所需成本较高，算法实现难度较大；由于需要同时优化多个目标，算法的收敛速度可能会受到影响。

单目标优化适用于需要对某个特定性能指标进行极限优化的应用场景，例如某些特殊应用需要极高的断裂强度时，采用单目标优化策略将更加适宜。多目标优化则适用于需要全方位提高各种性能指标的应用场景，例如在生产过程中需要兼顾断裂强度、透气性和柔软度，以保证产品在各个方面均具有卓越性能，此时采用多目标优化策略将更为恰当。

单目标优化在提升特定指标方面效果显著，但可能会忽视其他性能指标，因此并不适用于需要全面优化的应用场景。多目标优化能够充分考虑到多个性能指标，并在其中寻找平衡点，因此适用于需要全面优化的复杂应用场景，但是由于计算复杂度较高，收敛速度较慢也是其不可避免的缺陷。

## 4.4 问题4的模型建立与求解

### 4.4.1 多目标约束优化

为了在沙漠中进行科学考察，科考队需要一种具备最佳性能的人造革，以应对极端环境的挑战。人造革的性能需要综合考虑透气率、透湿率、力学性能和柔软性能等多个指标，在前期研究中，透气率和透湿率的最佳工艺参数已确定为树脂含量23wt%、固化温度116℃、碱减量程度16%。为了进一步优化人造革的力学性能和柔软性能，我们使用了多目标约束优化方法来找到综合考虑各项指标的最佳工艺参数组合，具体步骤如下：

（1）数据准备与预处理

首先收集了16组正交实验的工艺参数和各性能指标数据，其中包括树脂含量、固化温度和碱减量程度等工艺参数，以及透气率、透湿率、力学性能和柔软性能等性能指标。对数据进行预处理，确保数据的有效性和可用性。

（2）确定初始工艺参数

透气率和透湿率的最佳工艺参数已确定为树脂含量23wt%、固化温度116℃、碱减量程度16%。为了综合优化力学性能和柔软性能，我们定义了目标函数。该函数结合了力学性能和柔软性能的加权和，并采用线性回归模型预测各性能指标；设定透气率和透湿率的最佳工艺参数作为约束条件，确保优化过程中这些参数保持在最佳值。

（3）优化求解

我们使用多目标约束优化方法（如SLSQP算法）进行求解，寻找在满足透气率和透湿率最佳工艺参数的约束条件下，力学性能和柔软性能的最优工艺参数组合。

（4）结果分析

经过优化求解，得到的最佳工艺参数组合为：树脂含量23wt%、固化温度116℃、碱减量程度16%。这一结果在兼顾透气率和透湿率的基础上，优化了人造革的力学性能和柔软性能。

### 4.4.2 结果分析与建议

通过多目标约束优化方法，成功找到了兼顾多项性能指标的最佳工艺参数组合。具体而言，保持透气率和透湿率在最佳状态下，同时优化了力学性能和柔软性能，使得最终工艺参数为：树脂含量23wt%、固化温度116℃、碱减量程度16%。尽管通过优化方法得到了最佳工艺参数，但建议进行进一步的实验验证，以确保在实际生产中的效果和稳定性。根据实际需求，适当调整各性能指标的权重。例如，在某些情况下，力学性能可能比柔软性能更重要，可以调整权重以反映这种需求。对人造革进行长时间的性能评估，特别是在沙漠极端环境中的耐用性，以确保材料在长期使用中的可靠性。尽管当前的优化结果是树脂含量23wt%、固化温度116℃、碱减量程度16%，但在实际生产中，可以适当调整这些参数，以应对生产过程中的不确定性和变化。继续探索和实验新材料，评估其在相同工艺参数下的性能表现，以寻找潜在的更优材料。建立自动化的优化系统，能够实时调整工艺参数，以适应不同环境和需求的变化，提高生产效率和产品质量。

# 五、模型评价与推广

## 5.1 模型优点

对于问题1，此模型的优点体现在多元线性回归模型可以同时考虑多个影响因素及其相互作用对性能指标的综合影响，提供更全面的分析结果；通过回归系数，模型能够量化每个参数及其交互作用对性能指标的具体影响程度，便于优化参数设置；多元线性回归模型可以扩展至包含更多参数和更复杂的交互作用项，满足不同复杂程度的分析需求。

对于问题2，此模型的优点主要体现在多元线性回归模型简单易用，但能够有效捕捉多个自变量对因变量的影响，并且适用于各种应用场景；回归系数提供了每个自变量对因变量的独立贡献，便于理解和解释每个工艺参数对性能指标的影响；通过p值判断自变量的显著性，帮助筛选出对因变量有显著影响的自变量，有助于优化工艺参数组合；考虑工艺参数之间的交互作用，可以揭示更复杂的关系，优化工艺参数组合，提高产品性能；尽管线性回归在大样本情况下表现更好，但在小样本数据下（如本研究中的16组实验数据）依然适用，通过数据标准化处理可以提高模型的稳健性和可靠性。

对于问题3，此模型的优点体现在建立好的回归模型可以用于预测未来不同参数组合下的性能指标，具有较高的实用价值；通过统计检验，模型能够识别出哪些参数及其交互作用对性能指标有显著影响，从而指导优化方向。

对于问题4，此模型的优点体现在该方法能同时考虑热湿舒适性、力学性能和柔软性能等多个性能指标，使优化结果更符合实际需求；通过调整各性能指标的权重，能够灵活地平衡不同性能指标之间的优先级，满足不同应用场景的需求。

## 5.2 模型的缺点

模型的缺点主要体现在对数据质量和模型假设的依赖性较强，这可能受到实际情况的影响。首先，数据质量的高低直接影响模型的可靠性和预测准确性。如果数据存在不完整或测量误差，模型的预测结果可能会偏离实际情况。其次，模型假设通常包括变量之间的线性关系、独立性和正态分布等，这些假设在实际应用中可能不完全成立。如果真实世界中的工艺参数和性能指标之间存在非线性关系或复杂的交互作用，而模型未能适当地捕捉这些特性，则可能导致预测不准确。此外，模型对极端值和异常值较为敏感，这可能进一步影响模型的稳定性和鲁棒性。因此，在应用模型进行工艺参数优化时，需谨慎处理数据质量问题，检验模型假设的合理性，并考虑潜在的非线性和复杂交互作用，以提高模型的适用性和可靠性。

## 5.3 模型的改进与推广

为了改进多元线性回归模型并增强其在实际应用中的适用性和可靠性，首先需要加强对数据质量的控制。通过数据清洗和预处理，减少数据的不完整性和测量误差，确保数据的准确性和完整性。其次，可以引入更加灵活的模型，如广义线性模型（Generalized Linear Models, GLMs）和非线性回归模型，以捕捉工艺参数和性能指标之间的非线性关系和复杂交互作用。此外，采用稳健回归技术（Robust Regression）可以提高模型对极端值和异常值的抵抗能力，增强模型的稳定性和鲁棒性。在模型假设的检验和调整方面，可以使用诊断工具和残差分析来评估线性假设的合理性，并根据需要对模型进行调整或转向更合适的模型形式。为了推广模型的应用，可以通过开发自动化的数据分析和模型构建工具，使得用户能够更方便地应用这些改进的模型进行工艺参数优化。同时，结合领域知识，建立专门针对特定应用场景的优化模型和方法，提高模型在不同应用场景中的实际效果和用户满意度。

# 六、参考文献

1. 林彬. 多元线性回归分析及其应用[J]. 中国科技信息, 2010 (9): 60-61.
2. 李明伟, 刘正平. 大数据视角下的因果性与相关性问题研究[J]. Advances in Philosophy, 2024, 13: 905.
3. 张维元. 深入浅出 Python 数据分析[M]. Qing hua da xue chu ban she, 2022.
4. 董帮强. 基于复杂特性挖掘的关键变量回归预测模型研究[D]. 导师：张延飞. 东华理工大学, 2022.
5. 王培刚, 梁静, 张刚鸣. 多元统计分析与SAS实现[M]. 武汉大学出版社: 202011.
6. Malek Moh H., Coburn Jared W., Marelich William D. Advanced Statistics for Kinesiology and Exercise Science:A Practical Guide to ANOVA and Regression Analyses[M]. Wayne State University, USA;California State University, Fullerton, USA: 2018-08-06.
7. 胡完. 复杂资料综合投影寻踪回归分析法与综合传统回归分析法的比较研究[D]. 中国人民解放军军事医学科学院, 2017.
8. 刘斓乾. 多元线性回归和粗糙集聚类在疫情数据分析中的应用[D]. 吉林大学, 2020.
9. 蒋景明. 多元线性回归模型中异方差问题的研究[D]. 桂林理工大学, 2022.
10. 夏婉玉. 基于多元线性回归方法的疫情监测系统研究[D].巴继东. 武汉工程大学, 2022.
11. 董帮强. 基于复杂特性挖掘的关键变量回归预测模型研究[D]. 东华理工大学, 2022.
12. 姜启. 多元线性回归联合列线图构建过敏性紫癜肾脏受累的风险评估模型[D]. 安徽医科大学, 2023.
13. 徐维超. 相关系数研究综述[J]. 广东工业大学学报, 2012, 29(3): 12-17.
14. 邢航. 多因素方差分析中数学模型的建立与检验方法[J]. 电大理工, 2008 (2): 73-75.

# 七、附录

## 数据

16组正交试验工艺参数与各性能指标数据与基布性能数据

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **实验组** | **树脂含量** | **固化温度** | **碱减量程度** | **断裂强力** | **断裂伸长率** | **撕裂强力** | **透气率** | **透湿率** | **柔软度** | **折皱回复角** |
| **wt%** | **℃** | **%** | **N** | **%** | **N** | **L/(m2·s)** | **g/(m2·24h)** | **mm** | **°** |
| **1** | 15 | 100 | 0 | 1764.94 | 1.07 | 176.27 | 179.19 | 3505.30 | 1.15 | 149.36 |
| **2** | 15 | 110 | 10 | 1146.95 | 0.88 | 116.32 | 206.68 | 3106.71 | 2.13 | 157.57 |
| **3** | 15 | 120 | 20 | 1205.35 | 0.90 | 98.30 | 124.42 | 2947.00 | 2.91 | 162.20 |
| **4** | 15 | 130 | 30 | 1218.92 | 0.98 | 102.22 | 90.35 | 3010.60 | 3.43 | 166.58 |
| **5** | 20 | 100 | 10 | 1162.50 | 0.89 | 111.17 | 328.35 | 2595.05 | 2.04 | 158.59 |
| **6** | 20 | 110 | 0 | 1655.56 | 1.06 | 159.91 | 229.08 | 2607.77 | 0.99 | 150.28 |
| **7** | 20 | 120 | 30 | 1200.64 | 0.92 | 104.10 | 137.30 | 2558.30 | 3.03 | 168.47 |
| **8** | 20 | 130 | 20 | 1235.66 | 0.95 | 110.88 | 143.02 | 2713.78 | 2.94 | 168.30 |
| **9** | 25 | 100 | 20 | 1184.16 | 0.86 | 91.94 | 169.50 | 2473.50 | 2.19 | 163.19 |
| **10** | 25 | 110 | 30 | 1234.01 | 0.96 | 91.01 | 88.28 | 2636.04 | 2.82 | 166.30 |
| **11** | 25 | 120 | 0 | 1746.72 | 1.09 | 151.33 | 185.28 | 2240.28 | 0.90 | 154.47 |
| **12** | 25 | 130 | 10 | 1226.88 | 0.90 | 108.89 | 267.61 | 2548.41 | 1.62 | 159.59 |
| **13** | 30 | 100 | 30 | 1258.72 | 0.92 | 93.71 | 99.23 | 2621.91 | 2.87 | 162.30 |
| **14** | 30 | 110 | 20 | 1161.73 | 0.91 | 110.36 | 214.58 | 2134.28 | 1.71 | 163.33 |
| **15** | 30 | 120 | 10 | 1184.98 | 0.80 | 104.48 | 316.04 | 2204.95 | 1.46 | 158.64 |
| **16** | 30 | 130 | 0 | 1717.97 | 1.01 | 145.43 | 159.52 | 2124.38 | 0.85 | 151.34 |
| **基布** |  |  |  | 1484.79 | 1.22 | 180.64 | 201.88 | 3287.63 | 3.66 | 161.98 |

## Python代码

|  |  |
| --- | --- |
| 数据预处理 | 计算每组正交实验的平均值 |
| import pandas as pd  import matplotlib.pyplot as plt  from matplotlib.font\_manager import FontProperties  # 加载实验数据文件，跳过前三行，并且不读取第一列  data\_1 = pd.read\_excel('附件1 断裂强力.xlsx', skiprows=2, usecols=[1], names=['value'])  data\_2 = pd.read\_excel('附件2 断裂伸长率.xlsx', skiprows=2, usecols=[1], names=['value'])  data\_3 = pd.read\_excel('附件3 撕裂强力.xlsx', skiprows=2, usecols=[1], names=['value'])  data\_4 = pd.read\_excel('附件4 透气率.xlsx', skiprows=2, usecols=[1], names=['value'])  data\_5 = pd.read\_excel('附件5 透湿率.xlsx', skiprows=2, usecols=[1], names=['value'])  data\_6 = pd.read\_excel('附件6 柔软度.xlsx', skiprows=2, usecols=[1], names=['value'])  data\_7 = pd.read\_excel('附件7 折皱回复角.xlsx', skiprows=2, usecols=[1], names=['value'])  # 将数据转换为数值类型  data\_1 = data\_1.apply(pd.to\_numeric, errors='coerce')  data\_2 = data\_2.apply(pd.to\_numeric, errors='coerce')  data\_3 = data\_3.apply(pd.to\_numeric, errors='coerce')  data\_4 = data\_4.apply(pd.to\_numeric, errors='coerce')  data\_5 = data\_5.apply(pd.to\_numeric, errors='coerce')  data\_6 = data\_6.apply(pd.to\_numeric, errors='coerce')  data\_7 = data\_7.apply(pd.to\_numeric, errors='coerce')  # 计算每3行的平均值和标准差，从数据的第一个数据点开始  def calculate\_mean\_std(data):  mean\_values = []  std\_values = []  for i in range(0, len(data), 3):  subset = data.iloc[i:i+3]  mean\_values.append(subset.mean().iloc[0])  std\_values.append(subset.std().iloc[0])  return mean\_values, std\_values  data\_1\_mean, data\_1\_std = calculate\_mean\_std(data\_1)  data\_2\_mean, data\_2\_std = calculate\_mean\_std(data\_2)  data\_3\_mean, data\_3\_std = calculate\_mean\_std(data\_3)  data\_4\_mean, data\_4\_std = calculate\_mean\_std(data\_4)  data\_5\_mean, data\_5\_std = calculate\_mean\_std(data\_5)  data\_6\_mean, data\_6\_std = calculate\_mean\_std(data\_6)  data\_7\_mean, data\_7\_std = calculate\_mean\_std(data\_7)  # 确保所有列表长度相同  lengths = [len(data\_1\_mean), len(data\_2\_mean), len(data\_3\_mean), len(data\_4\_mean), len(data\_5\_mean), len(data\_6\_mean), len(data\_7\_mean)]  min\_length = min(lengths)  data\_1\_mean = data\_1\_mean[:min\_length]  data\_1\_std = data\_1\_std[:min\_length]  data\_2\_mean = data\_2\_mean[:min\_length]  data\_2\_std = data\_2\_std[:min\_length]  data\_3\_mean = data\_3\_mean[:min\_length]  data\_3\_std = data\_3\_std[:min\_length]  data\_4\_mean = data\_4\_mean[:min\_length]  data\_4\_std = data\_4\_std[:min\_length]  data\_5\_mean = data\_5\_mean[:min\_length]  data\_5\_std = data\_5\_std[:min\_length]  data\_6\_mean = data\_6\_mean[:min\_length]  data\_6\_std = data\_6\_std[:min\_length]  data\_7\_mean = data\_7\_mean[:min\_length]  data\_7\_std = data\_7\_std[:min\_length]  # 创建一个包含所有性能指标平均值和标准差的数据框  mean\_std\_data = {  '实验组': list(range(1, min\_length + 1)),  '断裂强力\_mean': data\_1\_mean,  '断裂强力\_std': data\_1\_std,  '断裂伸长率\_mean': data\_2\_mean,  '断裂伸长率\_std': data\_2\_std,  '撕裂强力\_mean': data\_3\_mean,  '撕裂强力\_std': data\_3\_std,  '透气率\_mean': data\_4\_mean,  '透气率\_std': data\_4\_std,  '透湿率\_mean': data\_5\_mean,  '透湿率\_std': data\_5\_std,  '柔软度\_mean': data\_6\_mean,  '柔软度\_std': data\_6\_std,  '折皱回复角\_mean': data\_7\_mean,  '折皱回复角\_std': data\_7\_std  }  mean\_std\_df = pd.DataFrame(mean\_std\_data)  # 将数据保存到 Excel 文件  output\_path = 'mean\_std\_data.xlsx'  mean\_std\_df.to\_excel(output\_path, index=False)  # 输出每组实验的平均值和标准差  for i in range(len(data\_1\_mean)):  print(f"实验组 {i+1}：")  print(f"断裂强力 平均值: {data\_1\_mean[i]}, 标准差: {data\_1\_std[i]}")  print(f"断裂伸长率 平均值: {data\_2\_mean[i]}, 标准差: {data\_2\_std[i]}")  print(f"撕裂强力 平均值: {data\_3\_mean[i]}, 标准差: {data\_3\_std[i]}")  print(f"透气率 平均值: {data\_4\_mean[i]}, 标准差: {data\_4\_std[i]}")  print(f"透湿率 平均值: {data\_5\_mean[i]}, 标准差: {data\_5\_std[i]}")  print(f"柔软度 平均值: {data\_6\_mean[i]}, 标准差: {data\_6\_std[i]}")  print(f"折皱回复角 平均值: {data\_7\_mean[i]}, 标准差: {data\_7\_std[i]}")  print()  # 绘制带误差棒的直方图  font = FontProperties(fname='C:/Windows/Fonts/simsun.ttc', size=12)  fig, axs = plt.subplots(4, 2, figsize=(20, 20))  fig.delaxes(axs[3][1]) # 删除多余的子图  colors = ['blue', 'orange', 'green', 'red', 'purple', 'brown', 'pink']  performance\_metrics = {  '断裂强力': ('断裂强力\_mean', '断裂强力\_std'),  '断裂伸长率': ('断裂伸长率\_mean', '断裂伸长率\_std'),  '撕裂强力': ('撕裂强力\_mean', '撕裂强力\_std'),  '透气率': ('透气率\_mean', '透气率\_std'),  '透湿率': ('透湿率\_mean', '透湿率\_std'),  '柔软度': ('柔软度\_mean', '柔软度\_std'),  '折皱回复角': ('折皱回复角\_mean', '折皱回复角\_std')  }  for ax, (metric, (mean\_col, std\_col)), color in zip(axs.flatten(), performance\_metrics.items(), colors):  ax.bar(mean\_std\_df['实验组'], mean\_std\_df[mean\_col], yerr=mean\_std\_df[std\_col], capsize=4, color=color)  ax.set\_xticks(mean\_std\_df['实验组'])  ax.set\_xticklabels(mean\_std\_df['实验组'], fontproperties=font, size=14)  ax.set\_xlabel('实验组', fontproperties=font, size=14)  ax.set\_ylabel(metric, fontproperties=font, size=14)  ax.tick\_params(axis='x', labelsize=12)  ax.tick\_params(axis='y', labelsize=12)  plt.tight\_layout()  plt.savefig('Performance\_Indicators\_Bar\_Chart.pdf')  plt.savefig('Performance\_Indicators\_Bar\_Chart.png')  plt.show() | |
| 数据预处理 | 正交实验对比分析 |
| import pandas as pd  import matplotlib.pyplot as plt  from matplotlib.font\_manager import FontProperties  # 加载处理后的数据和正交实验条件数据  data = pd.read\_excel('data.xlsx', header=0)  # 打印列名以确保正确  # 提取基布数据（最后一行）和实验数据（不包括最后一行）  baseline\_data = data.iloc[-1, 1:].apply(pd.to\_numeric, errors='coerce')  experimental\_data = data.iloc[1:-1, 1:].apply(pd.to\_numeric, errors='coerce')  # 提取性能指标名称（跳过第一列，即实验编号）  performance\_metrics = data.columns[1:]  # 提取实验组编号，并确保其为数值类型  experiment\_numbers = data.iloc[1:-1, 0].astype(int)  # 设置中文字体  font = FontProperties(fname='C:/Windows/Fonts/simsun.ttc', size=12)  # 创建一个字典，映射性能指标到单位  metric\_units = {  '断裂强力': 'N',  '断裂伸长率': '%',  '撕裂强力': 'N',  '透气率': 'mm/s',  '透湿率': 'g/m2·24h',  '柔软度': 'mm',  '折皱回复角': '°'  }  # 可视化分析  fig, axs = plt.subplots(4, 2, figsize=(20, 20))  fig.subplots\_adjust(hspace=0.5)  # 迭代字典，确保使用metric\_units中的键作为metric，值作为unit  for ax, (metric, unit) in zip(axs.flatten(), metric\_units.items()):  if metric in experimental\_data.columns:  ylabel = f'{metric} [{unit}]'  ax.bar(experiment\_numbers, experimental\_data[metric], label='实验数据')  ax.axhline(y=baseline\_data[metric], color='r', linestyle='--', label='基布数据')  ax.set\_xlabel('实验组编号', fontproperties=font) # 将横坐标名称设置为实验组编号  ax.set\_ylabel(ylabel, fontproperties=font) # 将纵坐标名称设置为性能指标名称和单位  ax.legend(prop=font)  ax.set\_xticks(experiment\_numbers) # 设置x轴刻度  ax.set\_xticklabels(experiment\_numbers, fontproperties=font) # 设置x轴刻度标签  else:  print(f"Metric '{metric}' not found in experimental\_data.columns")  # 删除最后一张没有数据的图  if len(metric\_units) % 2 != 0:  fig.delaxes(axs.flatten()[-1])  # 保存和展示图表  plt.tight\_layout()  plt.savefig('Performance\_Comparison.pdf')  plt.savefig('Performance\_Comparison.png')  plt.show() | |
| 数据预处理 | 各性能指标数据标准化 |
| import pandas as pd  from sklearn.preprocessing import StandardScaler  import matplotlib.pyplot as plt  # 设置字体为 SimHei，用于支持中文显示  plt.rcParams['font.sans-serif'] = ['SimHei']  plt.rcParams['axes.unicode\_minus'] = False  # 加载处理后的数据  data = pd.read\_excel('data.xlsx', header=0)  # 提取实验组数据（不包括第一行和最后一行）并转换为数值类型  experimental\_data = data.iloc[1:-1, 1:].apply(pd.to\_numeric, errors='coerce')  # 初始化StandardScaler  scaler = StandardScaler()  # 对数据进行标准化处理  standardized\_data = scaler.fit\_transform(experimental\_data)  # 将标准化后的数据转换为DataFrame，并添加列名  standardized\_df = pd.DataFrame(standardized\_data, columns=experimental\_data.columns)  # 添加实验组编号列  standardized\_df.insert(0, '实验组编号', data.iloc[1:-1, 0].values)  # 保存标准化后的数据到新的Excel文件  standardized\_df.to\_excel('standardized\_data.xlsx', index=False)  print("数据标准化处理完成，并保存为 'standardized\_data.xlsx'")  # 可视化标准化后的数据  plt.figure(figsize=(14, 8))  for column in standardized\_df.columns[1:]:  plt.plot(standardized\_df['实验组编号'], standardized\_df[column], marker='o', label=column)  plt.xlabel('实验组编号')  plt.ylabel('标准化值')  plt.title('标准化后的实验组性能数据')  plt.legend()  plt.grid(True)  plt.tight\_layout()  plt.savefig('Standardized\_Performance\_Comparison.png')  plt.savefig('Standardized\_Performance\_Comparison.pdf')  plt.show() | |
| 问题1 | Pearson相关系数分析前的数据标准化 |
| import pandas as pd  from sklearn.preprocessing import StandardScaler  # 加载处理后的数据  data = pd.read\_excel('data-1.xlsx', header=0)  # 提取实验组数据（不包括第一行和最后一行）并转换为数值类型  experimental\_data = data.iloc[1:-1, 1:].apply(pd.to\_numeric, errors='coerce')  # 初始化StandardScaler  scaler = StandardScaler()  # 对数据进行标准化处理  standardized\_data = scaler.fit\_transform(experimental\_data)  # 将标准化后的数据转换为DataFrame，并添加列名  standardized\_df = pd.DataFrame(standardized\_data, columns=experimental\_data.columns)  # 添加实验组编号列  standardized\_df.insert(0, '实验组编号', data.iloc[1:-1, 0].values)  # 保存标准化后的数据到新的Excel文件  standardized\_df.to\_excel('standardized\_data-1.xlsx', index=False)  print("数据标准化处理完成，并保存为 'standardized\_data-1.xlsx'") | |
| 问题1 | 工艺参数与各性能指标之间Pearson相关系数分析 |
| import pandas as pd  import seaborn as sns  import matplotlib.pyplot as plt  # 设置字体为 SimHei，用于支持中文显示  plt.rcParams['font.sans-serif'] = ['SimHei']  plt.rcParams['axes.unicode\_minus'] = False  # 加载标准化后的数据  data = pd.read\_excel('standardized\_data-1.xlsx')  # 提取工艺参数和性能指标的数据  process\_parameters = data.iloc[:, 1:4]  performance\_indicators = data.iloc[:, 4:]  # 计算工艺参数与性能指标之间的Pearson相关系数  correlation\_matrix = pd.concat([process\_parameters, performance\_indicators], axis=1).corr()  # 提取工艺参数与性能指标之间的相关系数部分  correlation\_process\_performance = correlation\_matrix.iloc[:3, 3:]  # 设置字体大小  plt.rcParams.update({'font.size': 14})  # 可视化相关系数矩阵  plt.figure(figsize=(12, 10))  sns.heatmap(correlation\_process\_performance, annot=True, cmap='coolwarm', vmin=-1, vmax=1)  plt.title('工艺参数与性能指标之间的Pearson相关系数', fontsize=18)  plt.xlabel('性能指标', fontsize=16)  plt.ylabel('工艺参数', fontsize=16)  plt.tight\_layout()  # 保存图表  plt.savefig('Process\_Performance\_Correlation.png')  plt.savefig('Process\_Performance\_Correlation.pdf')  # 显示图表  plt.show()  print("相关性分析完成，并生成图表 'Process\_Performance\_Correlation.png' 和 'Process\_Performance\_Correlation.pdf'") | |
| 问题1 | 各性能指标之间Pearson相关系数分析 |
| import pandas as pd  import seaborn as sns  import matplotlib.pyplot as plt  # 设置字体为 SimHei，用于支持中文显示  plt.rcParams['font.sans-serif'] = ['SimHei']  plt.rcParams['axes.unicode\_minus'] = False  # 加载标准化后的数据  data = pd.read\_excel('standardized\_data-1.xlsx')  # 提取性能指标数据（第5列到第11列）  performance\_data = data.iloc[:, 4:]  # 计算Pearson相关系数矩阵  correlation\_matrix = performance\_data.corr(method='pearson')  # 绘制热力图  plt.figure(figsize=(10, 8))  sns.heatmap(correlation\_matrix, annot=True, cmap='coolwarm', vmin=-1, vmax=1)  # 设置图表标题和轴标签字体大小  plt.title('性能指标之间的Pearson相关系数', fontsize=16)  plt.xlabel('性能指标', fontsize=14)  plt.ylabel('性能指标', fontsize=14)  # 调整布局  plt.tight\_layout()  # 保存图像  plt.savefig('Performance\_Correlation\_Matrix.png')  plt.savefig('Performance\_Correlation\_Matrix.pdf')  # 显示图像  plt.show() | |
| 问题1 | 方差分析评估工艺参数之间的交互作用 |
| import pandas as pd  import statsmodels.api as sm  from statsmodels.formula.api import ols  import matplotlib.pyplot as plt  import seaborn as sns  # 设置字体为 SimHei，用于支持中文显示  plt.rcParams['font.sans-serif'] = ['SimHei']  plt.rcParams['axes.unicode\_minus'] = False  # 读取标准化后的数据  data = pd.read\_excel('standardized\_data-1.xlsx', header=0)  # 提取工艺参数和性能指标  process\_params = data.columns[1:4] # 工艺参数列  performance\_metrics = data.columns[4:] # 性能指标列  # 创建一个空的DataFrame来存储ANOVA结果  anova\_results = pd.DataFrame(columns=['Performance Metric', 'Factor', 'p-value'])  # 对每个性能指标进行双因素ANOVA  for metric in performance\_metrics:  formula = f'{metric} ~ {process\_params[0]} \* {process\_params[1]} \* {process\_params[2]}'  model = ols(formula, data=data).fit()  anova\_table = sm.stats.anova\_lm(model, typ=2)  anova\_table = anova\_table.reset\_index().rename(columns={'index': 'Factor'})  anova\_table['Performance Metric'] = metric  non\_empty\_entries = anova\_table[['Performance Metric', 'Factor', 'PR(>F)']].rename(columns={'PR(>F)': 'p-value'})  non\_empty\_entries = non\_empty\_entries.dropna(how='all') # 移除空或全 NA 的条目  anova\_results = pd.concat([anova\_results, non\_empty\_entries])  # 过滤显著结果（p-value < 0.05）  significant\_results = anova\_results[anova\_results['p-value'] < 0.05]  # 可视化ANOVA结果  plt.figure(figsize=(14, 8))  sns.barplot(data=significant\_results, x='Performance Metric', y='p-value', hue='Factor')  plt.axhline(y=0.05, color='r', linestyle='--')  plt.xlabel('性能指标', fontsize=14)  plt.ylabel('p-value', fontsize=14)  plt.title('方差分析（ANOVA）结果：工艺参数之间的交互作用对性能的影响', fontsize=16)  plt.xticks(rotation=45)  plt.legend(title='因子', bbox\_to\_anchor=(1.05, 1), loc='upper left')  plt.tight\_layout()  # 保存图表  plt.savefig('ANOVA\_Results.png')  plt.savefig('ANOVA\_Results.pdf')  plt.show() | |
| 问题2 | 多元线性回归 |
| import pandas as pd  import statsmodels.api as sm  from statsmodels.formula.api import ols  from scipy.optimize import minimize  import matplotlib.pyplot as plt  import seaborn as sns  # 设置字体为 SimHei，用于支持中文显示  plt.rcParams['font.sans-serif'] = ['SimHei']  plt.rcParams['axes.unicode\_minus'] = False  # 读取数据  data = pd.read\_excel('data-1.xlsx', header=0)  # 去掉前两行非数据行  data = data.drop([0, 1]).reset\_index(drop=True)  # 将工艺参数和性能指标转换为数值类型  data = data.apply(pd.to\_numeric, errors='coerce')  # 提取工艺参数和性能指标  process\_params = data.columns[1:4] # 工艺参数列  performance\_metrics = data.columns[4:] # 性能指标列  # 定义目标函数用于优化  def objective\_function(params, formula, data):  data\_copy = data.copy()  data\_copy[process\_params[0]] = params[0]  data\_copy[process\_params[1]] = params[1]  data\_copy[process\_params[2]] = params[2]  model = ols(formula, data=data\_copy).fit()  return -model.rsquared\_adj  # 定义初始猜测值和边界  initial\_guess = [data[process\_params[0]].mean(), data[process\_params[1]].mean(), data[process\_params[2]].mean()]  bounds = [(data[process\_params[0]].min(), data[process\_params[0]].max()),  (data[process\_params[1]].min(), data[process\_params[1]].max()),  (data[process\_params[2]].min(), data[process\_params[2]].max())]  # 创建一个空的DataFrame来存储最佳工艺参数组合  optimal\_params\_df = pd.DataFrame(columns=['性能指标', '树脂含量', '固化温度', '碱减量程度'])  # 对每个性能指标进行多元线性回归和优化  for metric in performance\_metrics:  formula = f'Q("{metric}") ~ Q("{process\_params[0]}") \* Q("{process\_params[1]}") \* Q("{process\_params[2]}")'  result = minimize(objective\_function, initial\_guess, args=(formula, data), bounds=bounds, method='L-BFGS-B')  # 检查结果是否为空或全 NA  result\_df = pd.DataFrame([{  '性能指标': metric,  '树脂含量': result.x[0],  '固化温度': result.x[1],  '碱减量程度': result.x[2]  }])  if not result\_df.isnull().all().all():  optimal\_params\_df = pd.concat([optimal\_params\_df, result\_df], ignore\_index=True)  # 输出最佳工艺参数组合到excel文件  optimal\_params\_df.to\_excel('Multiple\_Linear\_Regression.xlsx', index=False)  # 可视化结果  plt.figure(figsize=(14, 8))  for i, metric in enumerate(performance\_metrics):  plt.subplot(2, 4, i + 1)  formula = f'Q("{metric}") ~ Q("{process\_params[0]}") \* Q("{process\_params[1]}") \* Q("{process\_params[2]}")'  model = ols(formula, data=data).fit()  sns.scatterplot(x=model.fittedvalues, y=data[metric], s=100)  plt.plot([data[metric].min(), data[metric].max()], [data[metric].min(), data[metric].max()], 'r--')  plt.xlabel('拟合值') # 横坐标标签改为中文  plt.ylabel('实际值') # 纵坐标标签改为中文  plt.title(metric)  plt.tight\_layout()  plt.savefig('Multiple\_Linear\_Regression\_visualization.png')  plt.savefig('Multiple\_Linear\_Regression\_visualization.pdf')  plt.show() | |
| 问题3 | 单目标优化模型 |
| import pandas as pd  import numpy as np  from scipy.optimize import minimize  import matplotlib.pyplot as plt  import seaborn as sns  from statsmodels.formula.api import ols # 确保导入ols  # 设置字体为 SimHei，用于支持中文显示  plt.rcParams['font.sans-serif'] = ['SimHei']  plt.rcParams['axes.unicode\_minus'] = False  # 读取数据  data = pd.read\_excel('data-1.xlsx', header=0)  # 去掉前两行非数据行  data = data.drop([0, 1]).reset\_index(drop=True)  # 将工艺参数和性能指标转换为数值类型  data = data.apply(pd.to\_numeric, errors='coerce')  # 提取工艺参数和性能指标  process\_params = data.columns[1:4] # 工艺参数列  performance\_metrics = data.columns[4:] # 性能指标列  # 定义目标函数  def objective\_function(params, metric, data):  data\_copy = data.copy()  data\_copy[process\_params[0]] = params[0]  data\_copy[process\_params[1]] = params[1]  data\_copy[process\_params[2]] = params[2]  model = ols(f'Q("{metric}") ~ Q("{process\_params[0]}") \* Q("{process\_params[1]}") \* Q("{process\_params[2]}")', data=data\_copy).fit()  return -model.rsquared\_adj  # 定义工艺参数的边界  bounds = [(data[process\_params[0]].min(), data[process\_params[0]].max()),  (data[process\_params[1]].min(), data[process\_params[1]].max()),  (data[process\_params[2]].min(), data[process\_params[2]].max())]  # 定义初始猜测值  initial\_guess = [data[process\_params[0]].mean(), data[process\_params[1]].mean(), data[process\_params[2]].mean()]  # 最优力学性能  result\_mechanical = minimize(objective\_function, initial\_guess, args=('断裂强力', data), bounds=bounds, method='L-BFGS-B')  optimal\_mechanical = result\_mechanical.x  # 最优热湿舒适性  result\_thermal = minimize(objective\_function, initial\_guess, args=('透湿率', data), bounds=bounds, method='L-BFGS-B')  optimal\_thermal = result\_thermal.x  # 最优柔软性能  result\_softness = minimize(objective\_function, initial\_guess, args=('柔软度', data), bounds=bounds, method='L-BFGS-B')  optimal\_softness = result\_softness.x  # 输出结果  optimal\_params\_df = pd.DataFrame({  '性能类型': ['力学性能', '热湿舒适性', '柔软性能'],  '树脂含量': [optimal\_mechanical[0], optimal\_thermal[0], optimal\_softness[0]],  '固化温度': [optimal\_mechanical[1], optimal\_thermal[1], optimal\_softness[1]],  '碱减量程度': [optimal\_mechanical[2], optimal\_thermal[2], optimal\_softness[2]]  })  optimal\_params\_df.to\_excel('optimal\_process\_parameters\_single\_objective.xlsx', index=False)  # 可视化结果  plt.figure(figsize=(14, 8))  metrics = ['断裂强力', '透湿率', '柔软度']  optimal\_values = [optimal\_mechanical, optimal\_thermal, optimal\_softness]  for i, metric in enumerate(metrics):  plt.subplot(1, 3, i + 1)  formula = f'Q("{metric}") ~ Q("{process\_params[0]}") \* Q("{process\_params[1]}") \* Q("{process\_params[2]}")'  model = ols(formula, data=data).fit()  sns.scatterplot(x=model.fittedvalues, y=data[metric], s=100)  plt.plot([data[metric].min(), data[metric].max()], [data[metric].min(), data[metric].max()], 'r--')  plt.xlabel('拟合值')  plt.ylabel('实际值')  plt.title(metric)  plt.tight\_layout()  plt.savefig('optimal\_process\_parameters\_single\_objective\_visualization.png')  plt.savefig('optimal\_process\_parameters\_single\_objective\_visualization.pdf')  plt.show() | |
| 问题3 | 多目标优化模型 |
| import pandas as pd  import numpy as np  from deap import base, creator, tools, algorithms  import random  from statsmodels.formula.api import ols  # 读取数据  data = pd.read\_excel('data-1.xlsx', header=0)  # 去掉前两行非数据行  data = data.drop([0, 1]).reset\_index(drop=True)  # 将工艺参数和性能指标转换为数值类型  data = data.apply(pd.to\_numeric, errors='coerce')  # 提取工艺参数和性能指标  process\_params = data.columns[1:4] # 工艺参数列  performance\_metrics = data.columns[4:] # 性能指标列  # 设置权重  weights = np.array([1, 1, 1, 1, 1, 1, 1]) # 可根据需要调整权重  # 创建优化问题  creator.create("FitnessMin", base.Fitness, weights=(-1.0,))  creator.create("Individual", list, fitness=creator.FitnessMin)  toolbox = base.Toolbox()  # 定义基因  def uniform(low, up, size=None):  try:  return [random.uniform(a, b) for a, b in zip(low, up)]  except TypeError:  return [random.uniform(a, b) for a, b in zip([low] \* size, [up] \* size)]  toolbox.register("attr\_float", uniform, low=[data[col].min() for col in process\_params],  up=[data[col].max() for col in process\_params], size=len(process\_params))  # 定义个体  toolbox.register("individual", tools.initIterate, creator.Individual, toolbox.attr\_float)  # 定义种群  toolbox.register("population", tools.initRepeat, list, toolbox.individual)  # 定义目标函数  def objective\_function(individual):  resin\_content, curing\_temp, alkali\_reduction = individual  data\_copy = data.copy()  data\_copy[process\_params[0]] = resin\_content  data\_copy[process\_params[1]] = curing\_temp  data\_copy[process\_params[2]] = alkali\_reduction  inverse\_performance = []  for metric in performance\_metrics:  model = ols(f'Q("{metric}") ~ Q("{process\_params[0]}") \* Q("{process\_params[1]}") \* Q("{process\_params[2]}")',  data=data\_copy).fit()  predicted = model.fittedvalues.mean() # 使用预测的均值作为指标值  inverse\_performance.append(1 / predicted)  objective = np.dot(weights, inverse\_performance)  return objective,  toolbox.register("evaluate", objective\_function)  # 定义遗传算法操作  toolbox.register("mate", tools.cxBlend, alpha=0.5)  toolbox.register("mutate", tools.mutPolynomialBounded, low=[data[col].min() for col in process\_params],  up=[data[col].max() for col in process\_params], eta=20.0, indpb=0.2)  toolbox.register("select", tools.selNSGA2)  # 参数设置  population\_size = 100  generations = 100  # 创建初始种群  population = toolbox.population(n=population\_size)  # 运行遗传算法  result\_population, logbook = algorithms.eaMuPlusLambda(population, toolbox, mu=population\_size, lambda\_=population\_size,  cxpb=0.7, mutpb=0.3, ngen=generations, verbose=True)  # 获取优化结果  optimal\_individuals = tools.selBest(result\_population, k=1)  optimal\_params = [ind[:] for ind in optimal\_individuals]  # 反标准化工艺参数  optimal\_params = [  [ind[0], ind[1], ind[2]]  for ind in optimal\_params]  # 创建DataFrame存储结果  optimal\_params\_df = pd.DataFrame(optimal\_params, columns=['树脂含量', '固化温度', '碱减量程度'])  # 输出最佳工艺参数组合到excel文件  optimal\_params\_df.to\_excel('optimal\_process\_parameters\_multi\_objective.xlsx', index=False) | |
| 问题4 | 多目标约束优化 |
| import pandas as pd  import numpy as np  from scipy.optimize import minimize  from statsmodels.formula.api import ols  # 读取数据  data = pd.read\_excel('/mnt/data/data-1.xlsx', header=0)  # 去掉前两行非数据行  data = data.drop([0, 1]).reset\_index(drop=True)  # 将工艺参数和性能指标转换为数值类型  data = data.apply(pd.to\_numeric, errors='coerce')  # 提取工艺参数和性能指标  process\_params = data.columns[1:4] # 工艺参数列  performance\_metrics = data.columns[4:] # 性能指标列  # 定义目标函数  def objective\_function(params, data):  data\_copy = data.copy()  data\_copy[process\_params[0]] = params[0]  data\_copy[process\_params[1]] = params[1]  data\_copy[process\_params[2]] = params[2]    # 使用线性回归模型预测力学性能和柔软性能  mechanical\_model = ols(f'Q("{performance\_metrics[0]}") ~ Q("{process\_params[0]}") \* Q("{process\_params[1]}") \* Q("{process\_params[2]}")', data=data\_copy).fit()  softness\_model = ols(f'Q("{performance\_metrics[-1]}") ~ Q("{process\_params[0]}") \* Q("{process\_params[1]}") \* Q("{process\_params[2]}")', data=data\_copy).fit()    mechanical\_pred = mechanical\_model.fittedvalues.mean()  softness\_pred = softness\_model.fittedvalues.mean()    # 组合多个目标，将其合并为一个标量值  weights = [0.5, 0.5] # 可以根据需要调整权重  objective\_value = weights[0] \* -mechanical\_pred + weights[1] \* -softness\_pred    return objective\_value  # 初始工艺参数  initial\_guess = [23, 116, 16]  # 定义边界  bounds = [(data[process\_params[0]].min(), data[process\_params[0]].max()),  (data[process\_params[1]].min(), data[process\_params[1]].max()),  (data[process\_params[2]].min(), data[process\_params[2]].max())]  # 定义约束条件  def constraint(params):  resin\_content, curing\_temp, alkali\_reduction = params  return [resin\_content - 23, curing\_temp - 116, alkali\_reduction - 16]  constraints = [{'type': 'eq', 'fun': lambda x: constraint(x)[0]},  {'type': 'eq', 'fun': lambda x: constraint(x)[1]},  {'type': 'eq', 'fun': lambda x: constraint(x)[2]}]  # 运行优化  result = minimize(objective\_function, initial\_guess, args=(data,), bounds=bounds, constraints=constraints, method='SLSQP')  # 获取最优参数  optimal\_params = result.x  # 输出结果  optimal\_params\_df = pd.DataFrame([optimal\_params], columns=['树脂含量', '固化温度', '碱减量程度'])  optimal\_params\_df.to\_excel('optimal\_process\_parameters\_constrained.xlsx', index=False)  print("最优工艺参数组合:", optimal\_params) | |