所属类别	2022 年“华数杯”全国大学生数学建模竞赛	参赛编号
本科组		CM2202599

基于多元回归的插层熔喷非织造材料的性能研究

摘要

本文研究了影响插层熔喷非织造材料的工艺参数、结构变量和产品性能之间的关系和变化规律。

对于问题一：本文首先进行了数据清洗，处理了异常值和缺失值。将六个指标按照是否插层各自分为两组，并对两组数据进行 **Wilcoxon 符号秩检验**，发现插层前后的厚度、孔隙率、压缩回弹性率和过滤效率有所增大，而过滤阻力和透气性没有显著变化。考虑到不同组实验的接收距离和热空气速度不同，本文将插层率按照大小分组，与接收距离、热空气速度作为控制变量，以结构变量和产品性能作为响应变量，分别进行**多因素方差分析**。发现插层率对除过滤阻力外的五个指标均有影响。

对于问题二：本文先对 data3 中来自同一实验的数据取平均。以接收距离和热空气速度为自变量，以结构变量为因变量，进行**多项式回归分析**。其中厚度与自变量为线性关系，孔隙率、压缩回弹性率与自变量为二次关系。回归得到的 R^2 均在 **0.96** 以上，回归效果好。再用上述方法得到的表达式来预测问题 2 的表格数据，预测结果见表 9。

对于问题三：为了得到结构变量与产品性能的关系，本文首先采用**主成分分析法**，将三个结构变量降维成两个主成分。考虑到实验数据在边缘值的突变，本文利用两个主成分对三个产品性能变量进行**分段多项式回归分析**，得到主成分与产品性能之间的关系，进而得到结构变量与产品性能的关系。再利用**皮尔逊相关系数检验**分别检验结构变量之间和产品性能之间的相关关系。得到厚度与孔隙率、厚度与压缩回弹性、过滤效率与透气性之间存在相关关系。本文查询文献得到**厚度与孔隙率之间的表达式**，根据该表达式进行参数拟合，得到的 R^2 达到 **0.95**。另外两组关系则同样采用**多项式回归**，结果见表 14表 15。

为了求解使得过滤效率达到最大的工艺参数，本文对传统的一元阻滞增长模型进行改进，结合过滤效率的物理意义，推广得到**二元阻滞增长模型**，用于描述过滤效率先快速增长，后趋于平稳的现象。通过拟合参数，得到过滤效率关于工艺参数的表达式，其 R^2 为 **0.8379**，具有很高的拟合精度。再以过滤效率最高为规划目标，以接收距离和热空气速度的范围为约束进行**非线性规划**，求解出当**接收距离为 10.1720cm，热空气速度为 2000r/min 时，过滤效率达到最大值 99.9469%**。

对于问题四：本题为**双目标规划**问题。本文首先利用**二次多项式回归**，得到过滤阻力关于接收距离和热空气速度的函数表达式，其 R^2 为 **0.87**。之后，先对过滤效率与过滤阻力进行标准化与正向化，再对其进行加权组合，从而得到目标函数。根据题意，本文将接收距离、热空气速度、厚度和压缩回弹性率的相关限制作为上述规划问题的约束条件。进行规划求解之前，本文绘制了约束条件的可行域图像如图 10，从而对约束条件进行简化。利用 Python 求解规划问题，得到当**接收距离为 17.9781cm，热空气速度为 1149.6201r/min** 时，可以满足过滤效率尽可能高和过滤阻力尽可能小的规划目标。此时的**过滤效率为 92.6366%，过滤阻力为 28.9473Pa**。

最后，本文通过调整问题四的两个目标的权值大小进行了**灵敏度分析**，得到了相关指标随权值大小改变的变化趋势。验证了模型的稳定性。

关键字：**插层熔喷　多元回归模型　二元阻滞增长模型　多因素方差分析**

一、 问题重述

1.1 问题背景

熔喷非织造材料是通过熔喷非织造技术生产的一种超细纤维过滤材料，由于其纤维极细，具有很好的屏蔽和过滤作用，在医学领域常用于生产防护服和口罩等医用产品。而由于传统的熔喷非织造技术生产出来的产品的压缩回弹性差，科学家在此基础上改进出新的熔喷技术——插层熔喷技术。该技术在传统熔喷设备的基础上多加一台机器用于将短纤维进行梳理后鼓入熔喷纤维流当中进行复合，最终形成插层熔喷复合非织造材料[1]。

由于该技术的工艺参数较多，参数之间存在相互关联，直接分析其工艺参数与最终产品性能的关系存在较大困难。如果可以分别建立工艺参数与结构变量、结构变量和最终产品性能之间的关系模型，那么可以根据工艺参数决定结构变量，结构变量决定最终产品性能的关系，从而给出产品性能的调控机制的理论基础。

1.2 问题提出

问题一：通过研究按照不同插层率插层后结构变量和产品性能的变化规律，分析插层率是否对变化产生影响。

问题二：通过研究工艺参数与结构变量之间的关系，对题目给出的 8 组工艺参数，预测相应的结构变量数据。

问题三：通过研究结构变量与产品性能之间的关系、结构变量之间的关系和产品性能之间的关系，结合第二问，给出使产品过滤效率达到最高的工艺参数。

问题四：在接收距离尽量不大于 100cm，热空气速度尽量不大于 200r/min，厚度尽量不超过 3mm，压缩回弹性率尽量不低于 85% 的条件下，给出能够使过滤效率尽量高同时过滤阻力尽量小的工艺参数。

二、 问题分析

2.1 问题一的分析

根据题意，需要对不同插层率插层后的结构变量和产品性能的变化规律进行探究，为此，本文对题目给出的数据进行初步可视化，以便于查看是否存在异常值和缺失值，结合对插层前后结构变量和产品性能的改变，直观地给出插层率是否影响变化的定性结论。

由于在题目给出的表 data1 和表 data2 中，不同测试组之间的接收距离或者热空气速度不同，无法直接通过将不同测试组之间的插层率进行比较和回归分析得出定量结论。于是本文通过将接受距离和热空气速度纳入考虑范围，将这三个量作为控制变量，将 6 个指标（结构变量和产品性能的指标）作为观测变量，对 6 个观测变量分别进行多因素方差分析，给出插层率对指标变化的定量结论，并进一步给出可能存在的插层率与其他控制变量的交互作用对指标变化的定量结论。

2.2 问题二的分析

由于题目给出的表 data3 是在插层率固定的条件下对同一组工艺参数进行了三次实验的实验数据，本文首先判断该数据集是否存在异常值和缺失值，然后在对每一组工艺

参数的三组实验数据取平均数，得到新的数据集，并在该数据集上进行接下来的分析与求解。

该问需要研究 2 个工艺参数（接收距离和热空气速度）和 3 个结构变量（厚度、孔隙率和压缩回弹性率）之间的关系，由于本题的数据较少，本文采用传统的回归算法来求解，而不是使用机器学习的算法求解。

通过采用多项式回归分析的方法，以 2 个工艺参数为自变量，每组工艺参数对应的 3 个结构变量为因变量，最高次数为 2 次，分别进行多项式回归分析。然后根据得到的分析结果，分别代入题目给出的 8 组工艺参数，得到相应的预测结构变量，并以表格的形式给出。

2.3 问题三的分析

根据题意，可以将该问题分解成四个子问题，分别是确定结构变量与产品性能之间的关系、确定结构变量之间的关系和产品性能之间的关系、利用已有的关系和第二问的预测方程求解过滤效率最高的工艺参数。

针对第一个子问题，由于结构变量和产品性能都有三个指标，同时确定 3 对 3 的关系较为困难，本文采用逐一确定的方法，确定单个产品性能指标其与结构变量的关系。而又由于结构变量有三个，而且结构变量之间还存在关联，本文采用主成分分析法，从三个结构变量中提取出两个主成分，再利用提取出的两个主成分与单个产品性能指标进行非线性回归分析以求解主成分与单个产品性能指标之间的关系。

针对第二个子问题，由于实验数据取自于正态总体，并且连续以及服从线性关系，本文对结构变量和产品性能分别进行皮尔逊相关系数检验，根据检验得到的 p 值判断结构变量之间和产品性能之间的相关关系，对具有相关关系的结构变量或产品性能进行进一步的回归分析得到其定量的关系。

针对第三个子问题，由于第一个子问题已经得到结构变量与产品性能的关系，同时问题二中得到了工艺参数与结构变量之间的关系，通过将这两组关系联立，可以得到工艺参数与产品性能之间的关系。本文利用其中工艺参数与过滤效率之间的关系，将该问题转化成方程求最值的问题。通过绘制工艺参数与过滤效率的散点图，本文发现其存在类似 Sigmoid 函数的增长趋势，受到一元阻滞增长模型的启发，本文使用改进的一元阻滞增长模型对工艺参数和过滤效率的关系进行建模。再利用已知数据对该模型进行参数拟合，最后利用的得到的参数，以实际工艺参数的限制范围为约束条件，以过滤效率最大为目标函数，进行非线性规划，最后得到过滤效率的最大值以及取最大值时的工艺参数。

2.4 问题四的分析

根据题意可以判断出该问题是一道非线性规划问题，而且题目给出了约束条件和目标函数。其目标函数有两个，一是：过滤效率最大；二是：过滤阻力最小。为此本文进行多目标规划，将两个目标函数标准化最后进行线性组合，转换成单目标规划问题，求解过滤效率尽可能大同时过滤阻力尽可能小时的工艺参数，以及此时的过滤效率与过滤阻力。

三、 模型假设

（1）实验数据构成的总体是一个正态总体。
（2）热风速度均匀且接收距离无扰动。
（3）喷熔过程中没有杂质影响。

四、 符号说明

符号	含义	单位
pvalue	假设检验的 p 值	/
A, B, C	多因素方差分析的因素	/
$f(x, y)$	回归分析的因变量	/
p	回归分析的各项系数	/
R^2	回归分析的拟合优度	/
F	主成分分析的单指标值	/
R	主成分分析的相关系数矩阵	/
X, Y	皮尔逊相关系数检验的样本数据集	/
r_{XY}	皮尔逊相关系数	/
E	过滤效率	%
H	过滤阻力	Pa
r	阻滞增长模型待定参数	/

五、 模型的建立和求解

5.1 问题一模型的建立与求解

5.1.1 初步数据可视化及异常值处理

为了直观感受插层对结构变量和产品性能变化带来影响，本文将题目给出的数据进行了初步的可视化，得到以下图 1。

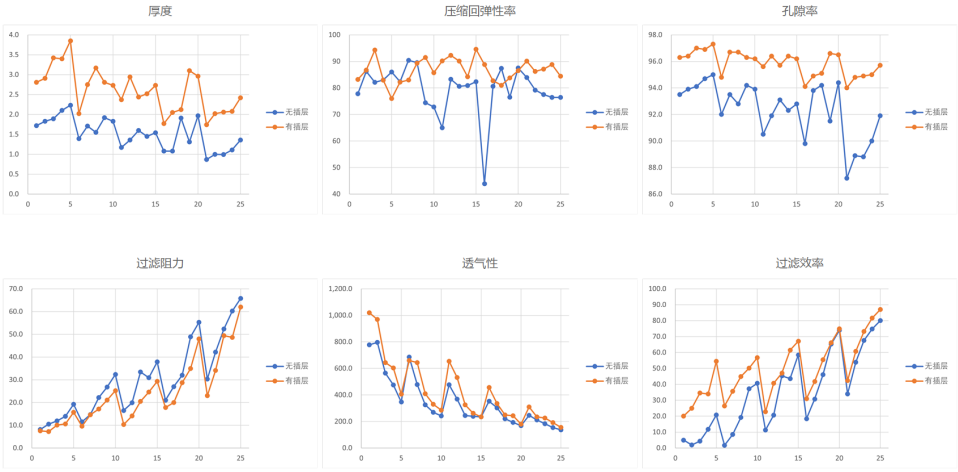


图 1　数据初步可视化

从上图可以看出插层对结构变量和产品性能各指标的值大小有提升或下降，但同时发现对于压缩回弹率这一结构变量而言，不进行插层的数据当中，第 16 组的数据过低，在相同热空气速度的条件下，改变接收距离和插层率带来的影响应当是连续的，对比第 1、6、16、21 组数据，可以发现，第 16 组数据比其他的数据小了将近一半，本文认为该数据可能是由于记录错误导致的异常值。为了更加明显地看出该数据地极端情况，本文在此给出压缩回弹性率的盒须图，如下图 2所示。

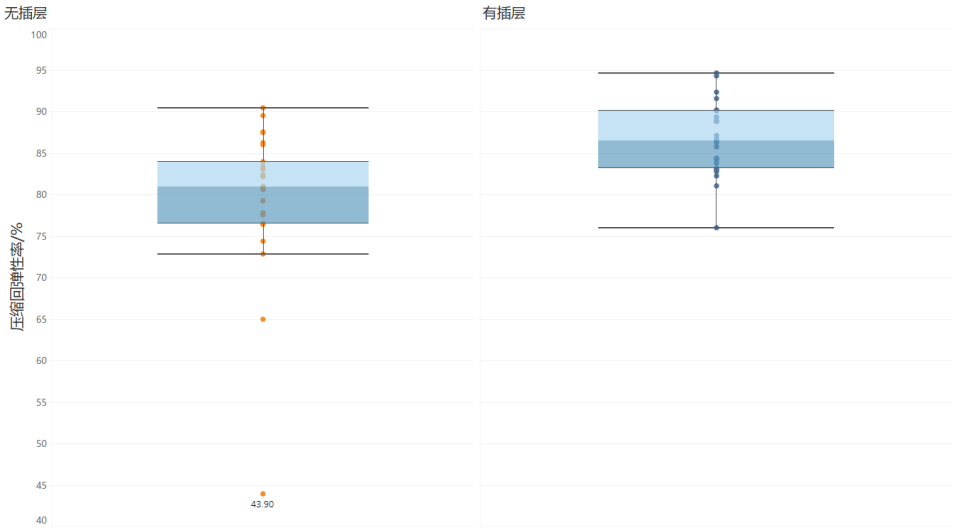


图 2　压缩回弹性率的盒须图

通过上述盒须图可以发现，第 16 组数据（即数值为 43.90 的点）在下须范围以外，虽然还有一个点（即第 6 组数据）也在下须范围以外，但是其偏离总体的程度不大，本文认为第 6 组数据是偶然误差，不做处理；而第 16 组数据是过失误差，本文用线性插值的方法，即相邻两组实验数据（第 11 组和第 21 组）的平均值来代替原有的异常值。

5.1.2 定性研究

本文通过对同一组实验样本插层前后的实验数据进行威尔科克森符号秩检验，来判断插层前后实验数据是否有明显变化。

威尔科克森符号秩检验

该检验是一种非参数检验，但在使用“非参数”一词时，通常意味着总体数据没有正态分布，并不意味着对总体一概不知。通过威尔科克森符号秩检验，可以在不假定数据服从正态分布的前提下，判断出相应的数据总体分布是否相同。

针对该问题，本文将“H_0：插层前后的实验数据是相似的”作为原假设，相应的“H_1：插层前后的实验数据是不同的”作为备择假设，使用威尔科克森符号秩检验进行假设检验。

利用 Python 中的 scipy.stats.ranksums 函数进行编程求解，本文得到单个指标威尔科克森符号秩检验的 p 值，如下表 1所示。

表 1　单个指标的威尔科克森符号秩检验 p 值

指标	pvalue	指标	pvalue	指标	pvalue
厚度	3.0331e-8	孔隙率	9.2880e-9	压缩回弹性率	0.0004
过滤阻力	0.1775	过滤效率	0.0313	透气性	0.2483

根据上表可以发现除过滤阻力和透气性的 p 值大于 0.05 之外，其余指标的 p 值都小于 0.05，即对于大部分的指标，可以拒绝原假设，对于过滤阻力和透气性，无法拒绝原假设。

本文认为对于过滤阻力和透气性这两个指标，虽然他们的 p 值大于 0.05，但是由于实验中不只是插层率在改变，还有接收距离和热空气速度两个变量存在，所以不能直接下结论认为对于过滤阻力和透气性这两个指标，插层率对其改变是没有影响的。而对于

其他的指标，根据威尔科克森符号秩检验，可以得到以下结论：插层率对这些指标的改变有影响，插层后厚度、压缩回弹性率、孔隙率和过滤效率显著增大，而过滤阻力和透气性在插层前后无显著变化。

为了更加清楚的反映插层率对指标变化的影响，本文接下来进行定量分析，通过多因素方差分析，建立插层率、接收距离和热空气速度三个控制变量与单个指标观测量的方差分析模型。

5.1.3 定量分析

数据预处理

为了便于后续分析和编程，本文先将题目所给的数据中的表 data2 与表 data1 联结，将单组数据的实验环境（即 data2 中的接收距离和热风速度）作为控制变量填入表 data1，与插层率并列。对于没有进行插层操作的实验，本文根据题目背景，将插层率设置为 0%。

通过上述操作得到新表 data1$^{\#}$，以下列出前三组数据在 data1$^{\#}$ 中的表示形式作为示例，如下表 2所示。

表 2　表 data1$^{\#}$ 示例

组号	编号	···	插层率（%）	接收距离（cm）	热风速度（r/min）
1	1#	···	0	40	800
	2#	···	36.44	40	800
2	1#	···	0	40	900
	2#	···	24.74	40	900
3	1#	···	0	40	1000
	2#	···	31.45	40	1000

多因素方差分析

本文在表 data1$^{\#}$ 的基础之上，建立以插层率、接收距离和热空气速度为控制变量，以单个指标为观测变量的多元多因素方差分析模型。

a. 明确观测变量和控制变量

由于本文要建立三因素方差分析模型，需要明确单个因素的水平个数。本文对插层率进行如下分类，见下表 3

表 3　插层率分类

0	$(0,10]$	$(10,20]$	$(20,30]$	$(30,40]$	$(40,\infty]$
25 组	3 组	9 组	6 组	5 组	2 组

对于接收距离和热空气速度，由于题目中给出的数据本身存在分类，在此直接使用其分层（即表 data2 中的测试环境）。

综上，设插层率为因素 A，接收距离为因素 B，热空气速度为因素 C。插层率因素有 6 个水平 $A_1, A_2, \cdots, A_6$，接受距离因素共有 5 个水平 $B_1, B_2, \cdots, B_5$，热空气速度因素共有 5 个水平 $C_1, C_2, \cdots, C_5$。对因素 A、B、C 的每一个水平的一组数据 $(A_i, B_j, C_k), (i = 1, 2, \cdots, 6; j = 1, 2, \cdots, 5; k = 1, 2, \cdots, 5)$ 只进行一次试验，得到 $L\,(= 6 \times 5 \times 5)$ 个试验结果 X_{ijk}。

b. 确定原假设和备择假设

检验各个因素样本的均值是否相等，如果相等，那么代表单个观测变量在控制变量改变前后没有显著差异，提出原假设为：

$$\begin{aligned} H_{0A}:\ &\mu_{1jk}=\mu_{2jk}=\cdots=\mu_{6jk},(j=1,2,\cdots,5;k=1,2,\cdots,5)\\ H_{0B}:\ &\mu_{i1k}=\mu_{i2k}=\cdots=\mu_{i5k},(i=1,2,\cdots,6;k=1,2,\cdots,5)\\ H_{0C}:\ &\mu_{ij1}=\mu_{ij2}=\cdots=\mu_{ij5},(i=1,2,\cdots,6;j=1,2,\cdots,5) \end{aligned} \tag{1}$$

备择假设为：

$$\begin{aligned} H_{1A}:\ &\mu_{1jk},\mu_{2jk},\cdots,\mu_{6jk}\text{不全相等}\\ H_{1B}:\ &\mu_{i1k},\mu_{i2k},\cdots,\mu_{i5k}\text{不全相等}\\ H_{1C}:\ &\mu_{ij1},\mu_{ij2},\cdots,\mu_{ij5}\text{不全相等} \end{aligned} \tag{2}$$

c. 构建模型

针对该问题，三因素控制变量分别为：插层率、接收距离和热空气速度。根据假设，有 $X_{ijk}\sim N\left(\mu_{ijk},\sigma^2\right)$ (μ_{ijk}和σ^2未知)，记 $X_{ijk}-\mu_{ijk}=\varepsilon_{ijk}$ ，即有 $\varepsilon_{ijk}=X_{ijk}-\mu_{ijk}\sim N\left(0,\sigma^2\right)$，故 $X_{ijk}-\mu_{ijk}$ 可视作随机误差，从而建立如下数学模型：

$$\begin{cases} X_{ijk}=\mu_{ijk}+\varepsilon_{ijk},(i=1,2,\cdots 6;j=1,2,\cdots,5;k=1,2,\cdots,5)\\ \varepsilon_{ijk}\sim N\left(0,\sigma^2\right),\left(\mu_{ijk},\sigma^2\text{未知},\varepsilon_{ijk}\text{相互独立}\right) \end{cases} \tag{3}$$

引入以下变量：

$$\begin{aligned} \mu&=\frac{1}{6\times5\times5}\sum_{i=1}^{6}\sum_{j=1}^{5}\sum_{k=1}^{5}\mu_{ijk}\\ \mu_{i\cdot\cdot}&=\frac{1}{5\times5}\sum_{j=1}^{5}\sum_{k=1}^{5}\mu_{ijk},(i=1,2,\cdots,6)\\ \mu_{\cdot j\cdot}&=\frac{1}{6\times5}\sum_{i=1}^{6}\sum_{k=1}^{5}\mu_{ijk},(j=1,2,\cdots,5)\\ \mu_{\cdot\cdot k}&=\frac{1}{6\times5}\sum_{i=1}^{6}\sum_{j=1}^{5}\mu_{ijk},(k=1,2,\cdots,5)\\ \alpha_i&=\mu_{i\cdot\cdot}-\mu,(i=1,2,\cdots,6)\\ \beta_j&=\mu_{\cdot j\cdot}-\mu,(j=1,2,\cdots5)\\ \gamma_k&=\mu_{\cdot\cdot k}-\mu,(k=1,2,\cdots,5) \end{aligned} \tag{4}$$

可知，$\sum_{i=1}^{6}\alpha_i=0$，$\sum_{j=1}^{5}\beta_j=0$，$\sum_{k=1}^{5}\gamma_k=0$。其中，μ 为总平均值，α_i 为插层率水平 A_i 的效应，β_j 为接收距离水平 B_j 的效应，γ_k 为热空气速度水平 C_k 的效应，且有 $\mu_{ijk}=\mu_{ijk}+\alpha_i+\beta_j+\gamma_k$。

于是上述模型可以转化成

$$\begin{cases} X_{ijk}=\mu_{ijk}+\varepsilon_{ijk},(i=1,2,\cdots,6;j=1,2,\cdots,5;k=1,2,\cdots,5)\\ \varepsilon_{ijk}\ N\left(0,\sigma^2\right),\left(\mu_{ijk},\sigma^2\text{未知},\varepsilon_{ijk}\text{相互独立}\right)\\ \sum_{i=1}^{6}\alpha_i=0,\sum_{j=1}^{5}\beta_j=0,\sum_{k=1}^{5}\gamma_k=0 \end{cases} \tag{5}$$

d. 模型求解

将总偏差平方和进行分解处理，有

$$\begin{aligned}
\bar{X} &= \frac{1}{6\times5\times5}\sum_{i=1}^{6}\sum_{j=1}^{5}\sum_{k=1}^{5}X_{ijk} \\
\bar{X}_{i\cdot\cdot} &= \frac{1}{5\times5}\sum_{j=1}^{5}\sum_{k=1}^{5}X_{ijk},(i=1,2,\cdots,6) \\
\bar{X}_{\cdot j\cdot} &= \frac{1}{6\times5}\sum_{i=1}^{6}\sum_{k=1}^{5}X_{ijk},(j=1,2,\cdots,5) \\
\bar{X}_{\cdot\cdot k} &= \frac{1}{6\times5}\sum_{i=1}^{6}\sum_{j=1}^{5}X_{ijk},(k=1,2,\cdots,5)
\end{aligned} \tag{6}$$

分解总偏差平方和：

$$\begin{aligned}
S_T &= \sum_{i=1}^{6}\sum_{j=1}^{5}\sum_{k=1}^{5}\left(X_{ijk}-\bar{X}\right)^2 \\
&= \sum_{i=1}^{6}\sum_{j=1}^{5}\sum_{k=1}^{5}\left[\begin{array}{c}\left(\bar{X}_{i\cdot\cdot}-\bar{X}\right)+\left(\bar{X}_{\cdot j\cdot}-\bar{X}\right)+\left(\bar{X}_{\cdot\cdot k}-\bar{X}\right)\\+\left(X_{ijk}-\bar{X}_{i\cdot\cdot}-\bar{X}_{\cdot j\cdot}-\bar{X}_{\cdot\cdot k}+2\bar{X}\right)\end{array}\right]^2
\end{aligned} \tag{7}$$

根据上式，可知 S_T 的展开式中交叉项的乘积为零，所以有

$$S_T = S_A + S_B + S_C + S_E \tag{8}$$

其中

$$\begin{aligned}
S_A &= \sum_{i=1}^{6}\sum_{j=1}^{5}\sum_{k=1}^{5}\left(X_{i\cdot\cdot}-\bar{X}\right)^2 = s\sum_{i=1}^{6}\left(X_{i\cdot\cdot}-\bar{X}\right)^2 \\
S_B &= \sum_{i=1}^{6}\sum_{j=1}^{5}\sum_{k=1}^{5}\left(X_{\cdot j\cdot}-\bar{X}\right)^2 = r\sum_{j=1}^{5}\left(X_{\cdot j\cdot}-\bar{X}\right)^2 \\
S_C &= \sum_{i=1}^{6}\sum_{j=1}^{5}\sum_{k=1}^{5}\left(X_{\cdot\cdot k}-\bar{X}\right)^2 = t\sum_{k=1}^{5}\left(X_{\cdot\cdot k}-\bar{X}\right)^2 \\
S_E &= \sum_{i=1}^{6}\sum_{j=1}^{5}\sum_{k=1}^{5}\left(X_{ijk}-X_{i\cdot\cdot}-X_{\cdot j\cdot}-X_{\cdot\cdot k}+2\bar{X}\right)^2
\end{aligned} \tag{9}$$

S_E 为误差平方和，S_A、S_B、S_C 分别为插层率因素、接收距离因素、热空气速度因素的偏差平方和。

类似易证，当 H_{0A}、H_{0B}、H_{0C} 成立时，$\frac{S_A}{\sigma^2}$、$\frac{S_B}{\sigma^2}$、$\frac{S_C}{\sigma^2}$、$\frac{S_E}{\sigma^2}$、$\frac{S_T}{\sigma^2}$ 以此服从自由度为 $(r-1)(t-1)$、$(s-1)(t-1)$、$(s-1)(r-1)$、$(s-1)(r-1)(t-1)$ 的卡方分布，且 S_A、S_B、S_C、S_E、S_T 相互独立。

本文使用 Python 的 statsmodels 库中的 ols 和 anova_lm 函数进行三因素方差分析（具体代码见附录），可得以下结果。（此处只列出部分值得讨论和分析的结果，全部的结果请见附录。）

到此，本文即可回答问题一中的插层率与结构变量和产品性能变化是否有关的问题。根据上表 4的 PR 值可以发现，插层率对过滤阻力的改变不存在显著影响，但对其他的指标存在显著影响。并且插层率和接收距离在对过滤效率的影响上存在交互作用。

表 4　方差分析结果

观测变量	控制变量	df	sum_sq	mean_sq	F	PR(>F)
厚度	插层率	1.0	13.1497	13.1497	127.4963	2.6438e-14
孔隙率	插层率	1.0	123.6365	123.6365	75.9460	5.7664e-11
压缩回弹性率	插层率	1.0	173.9754	173.9754	4.9694	0.0312
过滤阻力	插层率	1.0	36.3207	36.3207	2.4140	1.2776e-1
过滤效率	插层率	1.0	3430.4134	3430.4134	107.6217	3.7056e-13
透气性	插层率	1.0	3.2156e+04	3.2156e+04	7.1584	1.0591e-02
过滤效率	插层率: 接收距离	1.0	706.7732	706.7732	22.1735	2.7238e-05

本文接下来对结果进行进一步分析，试图挖掘进一步的数据特征。

e. 结果分析

1、厚度结果分析

由上表 4可以看出插层率对厚度的 P 值 <0.05，表明插层率对于厚度有显著影响，对此本文进一步进行事后检验得到以下结果。

表 5　厚度的事后检验结果

样本组 1	样本组 2	meandiff	p 值	是否拒绝原假设
0	1	0.9929	0.017	True
0	2	0.9945	0.001	True
0	3	1.0929	0.001	True
0	4	1.3992	0.001	True
0	5	1.0912	0.0364	True

上表只列出插层率为 0 和有插层率不为 0 的事后检验结果，可以看出可以拒绝原假设，即可得到插层率的有无对厚度的改变存在显著影响。对于其他的不同插层率之间的比较，由于无法拒接原假设，且组合次数多，在此不进行展示（可见附表），可以得出插层率的大小对厚度的改变没有显著影响。

同时本文给出两样本组之间的 meandiff，例如表 5的第一行数据表示在 95% 的置信度下，第 1 组的厚度比第 0 组的厚度大 0.9929mm。

2、孔隙率结果分析

同理对孔隙率进行结果分析，由表 4可以看出，插层率对孔隙率的 P 值 <0.05，表明插层率对于孔隙率有显著影响，对此本文进行事后检验得到以下结果。

表 6　孔隙率的事后检验结果

样本组 1	样本组 2	meandiff	p 值	是否拒绝原假设
0	1	3.4775	0.0172	True
0	2	3.2386	0.001	True
0	3	3.6258	0.001	True
0	4	3.8928	0.001	True

上表列出事后检验可以拒绝原假设的样本组之间的检验结果，可以看出对于绝大部分插层率为 0（即没有插层）的组与插层率不为 0 的组的事后检验，可以拒绝原假设，即可得插层率的有无对孔隙率的变化存在显著影响，具体的影响可通过两组样本之间的 meandiff 看出，例如上表 6中的第一行数据表示在 95% 的置信度下，第 1 组的孔隙率比第 0 组的孔隙率大 3.4775%。

表 6中没有列出的样本组之间的事后检验，都无法拒绝原假设，即可得到插层率的大小对于孔隙率的变化没有显著影响。

总结

通过上述分析和计算，本文得出以下结论：是否插层对除过滤阻力之外的指标变化都有显著影响，插层率的大小对的结构变量和产品性能的数据变化没有显著影响。

与此同时，本文通过查阅相关文献 [2] 得知：进行插层之后，卷曲弹性高，力学性能优异的 PET 能够起到支撑作用，将较大的孔隙分割为更细小的孔隙，从而提高材料的孔隙率与过滤效率，提高容尘性能。并且由于 PET 本身具有一定的卷曲度，使得插层喷熔非织造材料的厚度明显大于单一的喷熔非织造材料。

5.2 问题二模型的建立与求解

经检验，表 data3 中的数据不存在异常值和缺失值，于是本文按照问题分析，对每组工艺参数的三组实验数据取平均值，得到新的 25×8 的表 data3$^{\#}$，接下来的分析与求解都在该表上进行。

由于问题一在分析插层率的影响时进行了多元线性回归分析，而且针对厚度这一结构变量得到了较好的拟合优度，本文在此继续采用多元线性回归对厚度建立回归模型；对于孔隙率和 y 压缩回弹性率，本文以接收距离和热空气速度作为回归因子，分别以孔隙率和压缩回弹性率作为响应变量，建立多元二次回归模型

$$f(x,y) = p_{00} + p_{10}x + p_{01}y + p_{20}x^2 + p_{11}xy + p_{02}y^2 \tag{10}$$

其中 $f(x,y)$ 为单个响应变量的值，x 为接收距离，y 为热空气速度，p 为各项系数。对于厚度的多元线性回归模型，直接令 p_{20}, p_{11}, p_{02} 为 0 即可。

通过 Python 中的 curve_fit 函数编程求解具体的系数 p，得到以下结果，见表 7。

表 7　回归方程系数

响应变量	p_{00}	p_{10}	p_{01}	p_{20}	p_{11}	p_{02}
厚度	-0.8602	0.0542	0.0018	0.0000	0.0000	0.0000
孔隙率	7.5064e+1	3.5197e-1	2.6050e-2	-1.5417e-3	-1.7536e-4	-8.8912e-6
压缩回弹性率	4.3958e+1	1.1539	5.8802e-2	-2.0722e-2	2.0664e-5	-3.1080e-5

同时本文对回归方程的拟合优度 R^2 进行计算，得到以下结果，见表 8。

表 8　拟合优度

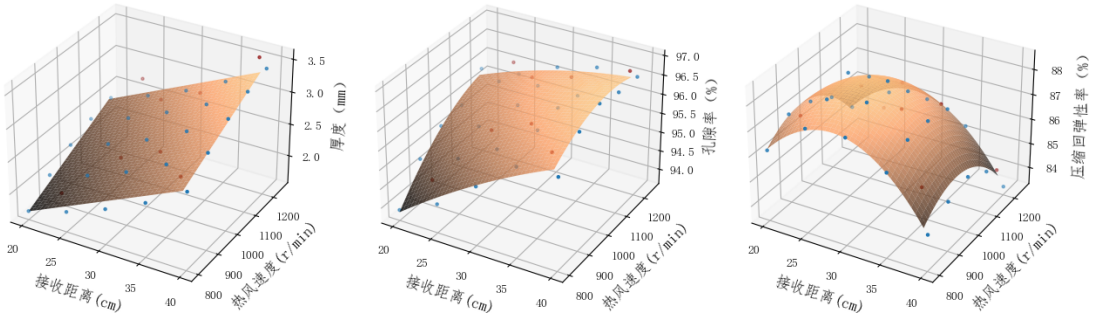
响应变量	R^2
厚度	0.9757
孔隙率	0.9654
压缩回弹性率	0.9894

由上表可以看出，拟合优度 R^2 都大于 0.95，可以认为得到的回归方程有较好拟合效果。本文利用得到的回归方程，对题目给出的 8 组工艺参数进行预测，得到以下结果，见表 9。

同时，本文对预测的结果和模型的回归拟合效果进行了可视化，分别得到以下示意图 3。

表 9　结构变量预测数据

接收距离（cm）	热风速度（r/min）	厚度（mm）	孔隙率（%）	压缩回弹性率（%）
38	850	2.76	96.27	86.08
33	950	2.68	96.23	87.93
28	1150	2.77	96.26	87.20
23	1250	2.69	95.97	85.07
38	1250	3.50	96.55	83.80
33	1150	3.05	96.54	86.77
28	950	2.41	95.77	88.38
23	850	1.95	94.63	87.46



(a) 厚度　　(b) 孔隙率　　(c) 压缩回弹性率

图 3　回归拟合结果和预测结果示意图

此外，本文通过查阅文献 [2]，得知：接收距离增大时，通过插层熔喷技术得到的纤维粘合性较差，会导致纤网的蓬松性提升，纤维直径增大，孔隙率随之增大。这与本文通过回归分析得出的结论是一致的。

5.3 问题三模型的建立与求解

5.3.1 子问题一的主成分分析模型

根据问题分析，结构变量之间存在关联关系，为了找到这其中起到支配作用的因素以便于后续的回归分析，本文对三个结构变量进行主成分分析，步骤如下：

Step1. 为消除不同的结构变量之间量纲不同带来的影响，首先对结构变量进行标准化处理

本部分涉及到的指标一共涉及的样本对象有 25 个，对应的指标有 3 个，令第 j 个样本的第 i 个指标值为 F_{ij}，将各个指标值进行标准化，记为 $\tilde{F}_{ij}$，其计算公式如下：

$$\tilde{F}_{ij} = \frac{F_{ij} - \bar{F}_i}{s_i} \tag{11}$$

其中，$\bar{F}_i$ 为第 i 个指标的均值，s_i 表示第 i 个指标的标准差。

Step2. 计算标准化后数据的相关系数矩阵

为了计算相关系数矩阵，需要先给出相关系数的计算方法。记第 i 个指标与第 j 个指标的相关系数为 r_{ij}，其计算公式如下：

$$r_{ij} = \frac{\sum_{k=1}^{25} \tilde{F}_{ik}\tilde{F}_{jk}}{25-1}, (i,j = 1,2,3) \tag{12}$$

则相关系数矩阵可以表示为 $R=(r_{ij})_{3\times3}$。

Step3. 计算相关系数矩阵的特征值和特征向量

令相关系数矩阵 R 的特征值为 $\lambda_1\geq\lambda_2\geq\lambda_3\geq0$，特征向量为 ξ_1,ξ_2,ξ_3。其中，特征向量 $\xi_i=(\mu_{i1},\mu_{i2},\cdots,\mu_{i25})^T$，利用特征向量组成新的指标变量：

$$\begin{cases} Y_1=\mu_{11}\tilde{F}_{11}+\mu_{12}\tilde{F}_{12}+\cdots+\mu_{125}\tilde{F}_{125} \\ Y_2=\mu_{21}\tilde{F}_{21}+\mu_{22}\tilde{F}_{22}+\cdots+\mu_{225}\tilde{F}_{225} \\ Y_3=\mu_{31}\tilde{F}_{31}+\mu_{32}\tilde{F}_{32}+\cdots+\mu_{325}\tilde{F}_{325} \end{cases} \tag{13}$$

其中，Y_i 为第 i 个主成分。

Step4. 确定最终的主成分

根据上述步骤可以得到三个结构变量的主成分，接下来本文利用 SPSS 统计软件计算三个主成分的特征值和累计贡献率，见下

表 10　主成分的特征值和累计贡献率

主成分	提取平方和载入		
	特征值	贡献率%	累计贡献率%
1	2.249132	74.9711	74.9711
2	0.707176	23.5725	98.5436
3	0.043691	1.4564	100.0000

在累计方差为 100% 的前提下，本文通过分析获得 3 个主成分，这 3 个主成分当中，主成分 1 和主成分 2 的贡献率较大，两个主成分的累计贡献率达到 98.5436%，可以认为这两个主成分是结构变量最重要的指标。本文接下来就利用得到的主成分 1 和主成分 2 进行非线性回归分析，求解两个主成分与单个产品性能之间的关系。

5.3.2 子问题一的非线性回归分析模型

为了求解两个主成分与单个产品性能指标之间的关系，本文采用非线性回归分析来求解以单个指标性能为因变量，以两个主成分为自变量的非线性回归方程。

通过初步构建回归方程，本文发现：对于过滤效率和透气性，实验数据在边界值的多项式回归曲线回归效果不佳，这可能是因为实验条件到达一定程度后，其内部机理可能发生变化，考虑使用分段讨论的方式将过滤效率和透气性的边界点进行单独的线性回归。而对于其余大部分点，仍然进行与上文相同的非线性回归分析。得到结果如下。

对于过滤效率和过滤阻力，本文采用以下非线性方程进行回归分析：

$$f(x,y)=p_{00}+p_{10}x+p_{01}y+p_{20}x^2+p_{11}xy \tag{14}$$

其中，$f(x,y)$ 表示过滤效率或过滤阻力（即因变量），x 表示主成分 1，y 表示主成分 2，p 表示各项的系数。

注意到，上式没有使用 y^2 参与回归分析，这是因为本文在尝试使用之后拟合优度反而下降，于是这里使用 x,y,x^2,xy 参与回归分析。得到以下结果，见表 11。

对于透气性，本文采用以下非线性方程进行回归分析：

$$f(x,y)=p_{00}+p_{10}x+p_{01}y+p_{20}x^2+p_{11}xy+p_{02}y^2 \tag{15}$$

其中，$f(x,y)$ 表示透气性（即因变量），x 表示主成分 1，y 表示主成分 2，p 表示各项系数。

表 11　过滤效率和过滤阻力的非线性回归分析结果

因变量	p_{00}	p_{10}	p_{01}	p_{20}	p_{11}	R^2
过滤阻力	29.1259	2.8736	-0.4385	0.7435	-0.3329	0.9011
过滤效率	50.1578	2.5436	-1.9378	-0.6129	1.1354	0.9023

表 12　透气性的非线性回归分析结果

p_{00}	p_{10}	p_{01}	p_{20}	p_{11}	p_{02}	R^2
427.9688	-18.9344	14.8663	5.4853	-19.7279	-3.4439	0.8007

通过 Python 中的 curve_fit 函数编程求解具体的系数和拟合优度，得到以下结果，见表 12。

由上述两表可以看出，拟合优度达到了 80% 以上，可以认为得到的回归方程有较好拟合效果，能够较好的反映两个主成分与单个产品性能之间的关系，于是对于大部分点，本文以这三个二元二次方程作为子问题一结构变量与产品性能之间关系的解答。

对于小部分边界点，本文在此给出其单独的分段讨论。

对于过滤效率：

当接收距离 ≤20 且热空气速度 ≥1000 时，进行二元线性回归，有：

$$f(x,y) = 1648.39x + 481.85y + 1618.29 \tag{16}$$

对于透气性：

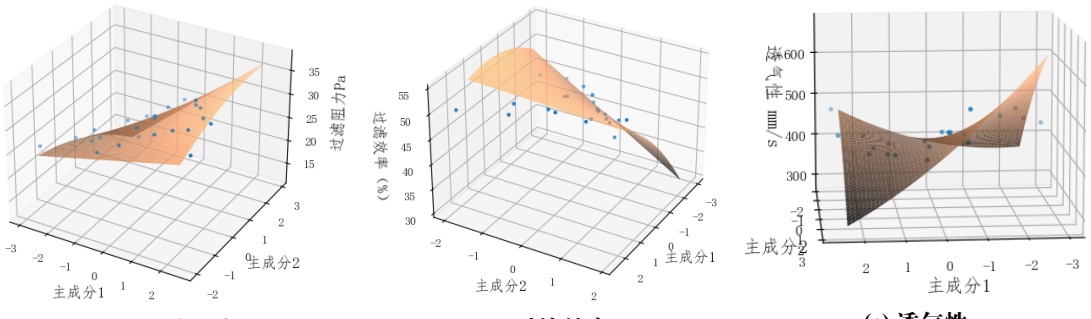
当接收距离 ≥40 且热空气速度 ≤900 时，进行二元线性回归，有：

$$f(x,y) = -1585.22x + 2166.66y + 3754.1 \tag{17}$$

当接收距离 ≤20 且热空气速度 ≥1100 时，进行二元线性回归，有：

$$f(x,y) = -45090x - 13243y - 41865 \tag{18}$$

此外，本文对模型的回归拟合效果进行了可视化，分别得到以下示意图 4。



(a) 过滤阻力　(b) 过滤效率　(c) 透气性

图 4　回归拟合结果示意图

5.3.3 子问题二的皮尔逊相关系数检验

由于实验数据取自于正态总体，且题目给出的实验数据是连续且服从线性关系的。因此本文可以通过皮尔逊（Pearson）相关系数检验以确定结构变量（或产品性能）之间

的相关性，对具有相关性的结构变量（或产品性能）进行非线性回归分析从而获得结构变量（或产品性能）之间的量化关系。

a. 确定原假设和备择假设

本文对三个结构变量（或产品性能）的组合分别进行皮尔逊相关系数检验，则令单次检验选择的结构变量分别为 $X:\{X_1,X_2,\cdots,X_{25}\}$ 和 $Y:\{Y_1,Y_2,\cdots,Y_{25}\}$，计算单次检验的 Pearson 相关系数 r_{XY}，其计算公式如下：

$$r_{XY}=\frac{Cov\left(X,Y\right)}{S_XS_Y} \tag{19}$$

其中 $Cov\left(X,Y\right)$ 表示两个样本数据的协方差，S_X 表示样本数据 X 的样本标准差，S_Y 表示样本数据 Y 的样本标准差。

根据单次检验的 Pearson 相关系数，给出原假设 H_0 为 $H_0:r_{XY}=0$，则其备择假设 H_1 为 $H_1:r_{XY}\neq0$。

b. 计算求解

本文通过 Python 编程求解 6 组皮尔逊相关系数（三个结构变量两两组合加上三个产品性能两两组合），并计算相应的 p 值，如果 p 值小于 0.05，则说明在 95% 的置信水平下拒绝原假设。通过计算，得到以下结果，见表 13。

表 13　Pearson 相关系数检验结果

组合	相关系数	pvalue
厚度与孔隙率	0.9336	9.6780e-12
厚度与压缩回弹性率	-0.5342	0.0060
孔隙率与压缩回弹性率	-0.3509	0.0855
过滤阻力与过滤效率	0.3909	0.0534
过滤阻力与透气性	-0.3873	0.0558
过滤效率与透气性	-0.7893	2.7317e-6

由上表可以看出厚度与孔隙率、厚度与压缩回弹性率、过滤效率与透气性的 p 值小于 0.05，可以认为在 95% 的置信水平下拒绝原假设，即厚度与孔隙率有线性相关关系、厚度与压缩回弹性率有线性相关关系、过滤效率与透气性有线性相关关系。

同时，为了更加清晰的看出这 6 组数据的相关性，本文通过热力图的形式对 6 组数据的相关系数进行可视化，得到下图 5。

5.3.4 子问题二的定量关系求解

根据上述皮尔逊相关系数检验的结果可知，厚度与孔隙率有线性相关关系、厚度与压缩回弹性率有线性相关关系、过滤效率与透气性有线性相关关系。本文对这三组具有相关关系的变量进行回归分析，给出其定量关系。

厚度与孔隙率

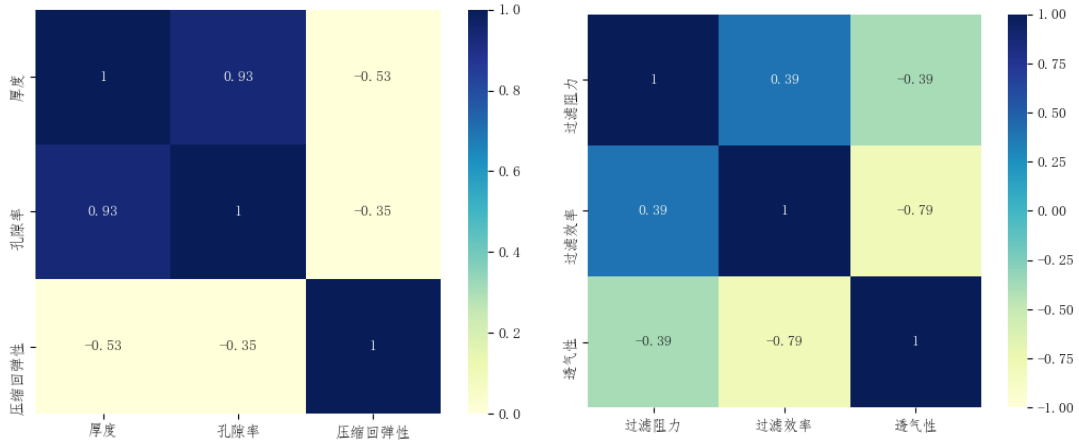
根据文献 [2] 中对材料孔隙率的计算公式可知：

$$n=\left(1-\frac{m}{\rho\delta}\right)\times100\% \tag{20}$$

其中 n 为孔隙率，m 为样品单位面积质量，ρ 为纤维密度，δ 为样品厚度。

从上述公式可知，厚度与孔隙率存在以下关系：

$$n=\left(1-\frac{\lambda}{\delta}\right)\times100\% \tag{21}$$



(a) 结构变量热力图　　(b) 产品性能热力图

图 5　相关关系热力图

其中 $\lambda = \frac{\rho}{m}$ 为需要通过拟合确定的系数。

根据式 (21)，本文通过参数拟合，确定 $\lambda = 0.1038$，且拟合优度 $R^2 = 0.9511$。

本文在此给出其回归曲线图，如图 6a。

拟合优度大于 95%，可以认为通过拟合得到的方程具有较好的拟合效果，并且通过上述回归曲线图能够看出拟合效果较好，能够较好的反映厚度与孔隙率的关系。

厚度与压缩回弹性率

本文通过非线性回归分析求解厚度与压缩回弹性率的定量关系。将厚度作为自变量 x，压缩回弹性率作为因变量 y，用以下一元二次方程表示其关系：

$$y = p_2 x^2 + p_1 x + p_0 \tag{22}$$

其中，p 为各项系数。

通过 Python 中的 curve_fit 函数编程求解具体的系数和拟合优度，得到以下结果，见表 14。

表 14　厚度与压缩回弹性率的回归结果

p_2	p_1	p_0	R^2
-1.998	9.166	76.73	0.459

本文在此给出其回归曲线图，如图 6b。

由上述图片可以看出回归曲线中的散点比较离散，同时通过表 14看出 R^2 的值并不高，说明得到的回归方程只适用于解释厚度与压缩回弹性率之间的关系，并不能用于预测。

过滤效率与透气性

本文通过非线性回归分析求解过滤效率与透气性的定量关系。将过滤效率作为自变量 x，透气性作为因变量 y，用以下一元一次方程表示其关系：

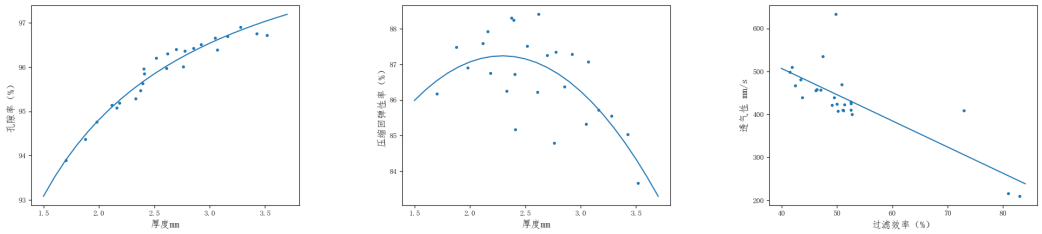
$$y = p_1 x + p_0 \tag{23}$$

其中，p 为各项系数。

通过 Python 中的 curve_fit 函数编程求解具体的系数和拟合优度，得到以下结果，见表 15。

表 15　过滤效率与透气性的回归结果

p_1	p_0	R^2
-6.104	751.3	0.623



(a) 厚度与孔隙率　(b) 厚度与压缩回弹性率　(c) 过滤效率与透气性

图 6　回归曲线图

本文在此给出其回归曲线图，如图 11b。

由上述图片可以看出回归曲线中的个别散点与总体偏离较远，同时通过表 15看出 R^2 的值并不高，说明得到的回归方程只适用于解释过滤效率与透气性之间的关系，并不能用于预测。

5.3.5 子问题三的工艺参数与过滤效率关系模型的建立

根据问题分析，需要对工艺参数与过滤效率的关系进行数学刻画，本文首先根据已知数据绘制出两个工艺参数（接收距离和热空气速度）与过滤效率的散点图图 7:

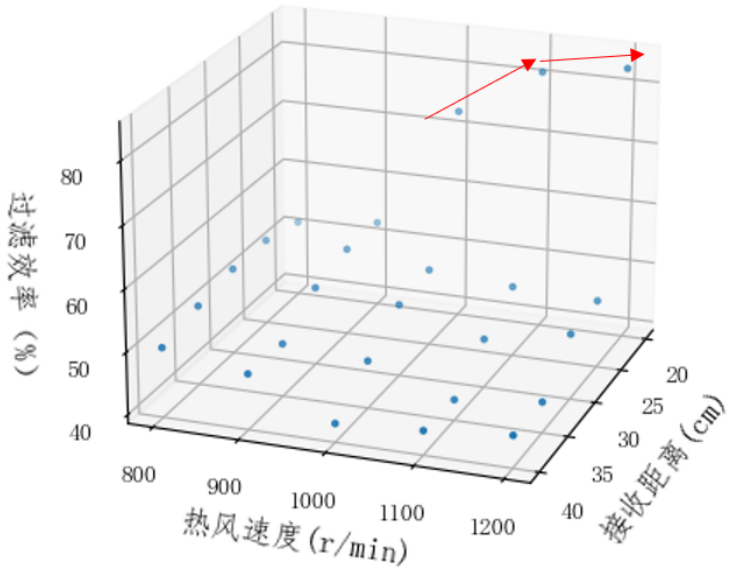


图 7　工艺参数与过滤效率的散点图

根据上图可以看出在已知数据的工艺参数最大值附近，出现陡增和上升趋势，而后又出现趋于平缓的趋势。观察题目给出的过滤效率的值，结合实际意义可知，过滤效率应当在 40%~100% 之间。结合 Sigmoid 函数的特征，本文认为该数据适合使用阻滞增长模型。

由于阻滞增长模型对具有类似 Sigmoid 函数的增长趋势的数据有较好的拟合效果，但是传统的阻滞增长模型是单变量模型，于是本文对其进行改进，建立改进后的二元阻滞增长模型用于该问对工艺参数与过滤效率的关系进行数学刻画。以下是其建立过程：

Step1. 建立单变量指数增长模型

根据 Logistic 人口增长模型 [3] 中对相对增长率的描述，即相对增长率 = 出生率-死亡率，本文建立过滤效率的单变量增长率模型：

$$E\left(x+\Delta x\right)-E\left(x\right)=rE\left(x\right)\Delta x \tag{24}$$

其中，$E\left(x\right)$ 表示单变量为 x 时的过滤效率，r 为过滤效率的增长率。

通过观察题目给出实验数据，本文确定过滤效率初始值为 40%，从而可将上式变形为：

$$\frac{dE}{dx}=rE, E_0=40 \tag{25}$$

其中 E_0 为过滤效率的初值。

据此可化简出过滤效率的单变量指数增长模型：

$$E\left(x\right)=E_0e^{rx} \tag{26}$$

Step2. 建立过滤效率与单个工艺参数的关系式

为了刻画出当单个工艺参数达到一定值之后过滤效率增长速度变慢的情况，本文将过滤效率增长率 r 作为因变量，将过滤效率 E 作为自变量，构建过滤效率增长率关于单个工艺参数的递减函数：

$$r\left(E\right)=r_0-sE, (r,s>0) \tag{27}$$

其中，当 E 很小时，r_0 为过滤效率的固有增长率，s 为待定参数。当 E 很大时，过滤效率增长率为 0，故有：

$$r\left(E_m\right)=0 \tag{28}$$

其中 E_m 为过滤效率的最大值。

将其代入上式 (27) 有：

$$s=\frac{r_0}{E_m} \tag{29}$$

将式 (29) 代入 (27)，即可得到过滤效率增长率 r 与单个工艺参数 x 的最终关系式：

$$r\left(E\right)=r_0\left(1-\frac{E}{E_m}\right) \tag{30}$$

Step3. 建立单变量阻滞增长模型

将式 (30) 代入式 (26) 并化简即可得到所需建立的阻滞增长模型：

$$E\left(x\right)=\frac{E_m}{1+\left(\frac{E_m}{E_0}-1\right)e^{-r_0x}} \tag{31}$$

Step4. 建立二元阻滞增长模型

由于单变量阻滞增长模型不能全面的刻画过滤效率与两个工艺参数之间的关系，本文对其进行以下改进。

令 x 表示接收距离，y 表示热空气速度。先对两个变量利用式 (31) 建立单变量阻滞增长模型：

$$\begin{aligned}E\left(x\right)&=\frac{E_m}{1+\left(\frac{E_m}{E_0}-1\right)e^{-r_1x}}\\E\left(y\right)&=\frac{E_m}{1+\left(\frac{E_m}{E_0}-1\right)e^{-r_2y}}\end{aligned} \tag{32}$$

对于两个工艺参数对过滤效率的交互作用，本文用 x, y 一起作为自变量，建立以下改进的二元阻滞增长模型：

$$E(x,y)=\frac{E_m}{1+\left(\frac{E_m}{E_0}-1\right)e^{-r_3x-r_4y}} \tag{33}$$

根据文献 [2] 中过滤效率的实验值，本文认为过滤效率的最大值 E_m 应当趋近于100%。由此本文将三个阻滞增长模型结合起来，对其分子 E_m 做以下调整：

$$E(x,y)=\frac{25}{1+\left(\frac{100}{E_0}-1\right)e^{-r_1x}}+\frac{25}{1+\left(\frac{100}{E_0}-1\right)e^{-r_2y}}+\frac{50}{1+\left(\frac{100}{E_0}-1\right)e^{-r_3x-r_4y}} \tag{34}$$

再将过滤效率的初值 $E_0=40$ 代入上式，化简可得最终改进的二元阻滞增长模型：

$$E(x,y)=\frac{25}{1+1.5e^{-r_1x}}+\frac{25}{1+1.5e^{-r_2y}}+\frac{50}{1+1.5e^{-r_3x-r_4y}} \tag{35}$$

其中，r_1, r_2, r_3, r_4 为该模型的待定参数。

5.3.6 子问题三的模型参数拟合

为了利用上式 (35) 求解过滤效率最大时的工艺参数，本文先利用已有数据进行模型参数的拟合。

本文通过 Python 中的 curve_fit 函数对参数进行拟合，并计算其拟合优度 R^2，得到以下结果，见表 16。

表 16　二元阻滞增长模型参数拟合

r_1	r_2	r_3	r_4	R^2
1.59	3.2794e-3	-0.5604	0.1077e-1	0.8379

本文在此给出用拟合参数代入模型的曲面图，如下图 8所示。

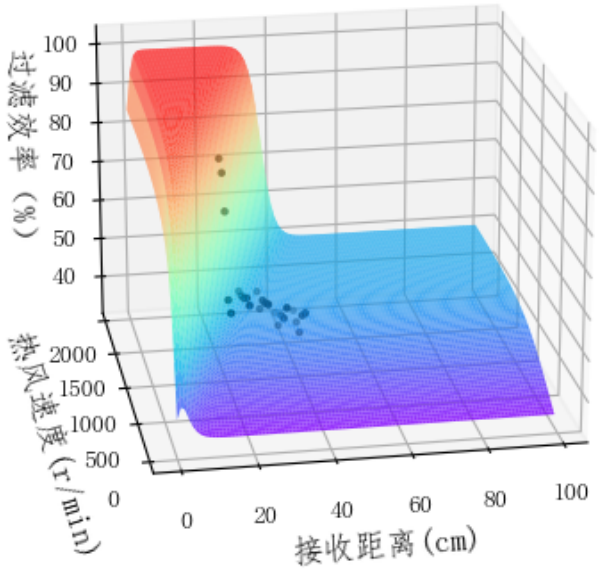


图 8　二元阻滞增长模型图

由上表 16中的 R^2 值可以看出，R^2 大于 83%，可以认为拟合得到的参数具有较好的拟合效果，同时根据图 8可以看出，该模型的拟合效果很好，可以较好的反映过滤效率与两个工艺参数的关系。

5.3.7 子问题三的非线性规划模型

a. 确定目标函数

根据上文得到的模型和参数，可以确定过滤效率与两个工艺参数的表达式。本文以过滤效率最大为目标函数，则得到以下目标函数的数学表达式：

$$\max_{x,y} E(x,y) = \frac{25}{1+1.5e^{-1.59x}} + \frac{25}{1+1.5e^{-\frac{3.2794}{1000}y}} + \frac{50}{1+1.5e^{0.5604x-0.0108y}} \tag{36}$$

b. 确定约束条件

根据实际工艺参数的限制范围，本文给出以下约束条件：

$$\begin{cases} 0 \le x \le 100 \\ 0 \le y \le 2000 \end{cases} \tag{37}$$

其中，x 为接收距离，y 为热空气速度。

c. 非线性规划求解

根据上述目标函数和约束条件，本文对以下非线性规划问题进行求解：

$$\max_{x,y} E(x,y) = \frac{25}{1+1.5e^{-1.59x}} + \frac{25}{1+1.5e^{-\frac{3.2794}{1000}y}} + \frac{50}{1+1.5e^{0.5604x-0.0108y}}$$
$$St. \begin{cases} 0 \le x \le 100 \\ 0 \le y \le 2000 \end{cases} \tag{38}$$

本文使用 python 中的 minimize 函数，根据上述目标函数和约束条件进行非线性规划的求解。

由于 minimize 函数初值的设置对其收敛性有较大影响，本文在约束条件确定的范围内以固定间隔选取共 200 组初值组合，并以相应的初值组合进行局部最优解的搜索。其搜索过程如下：

搜索算法

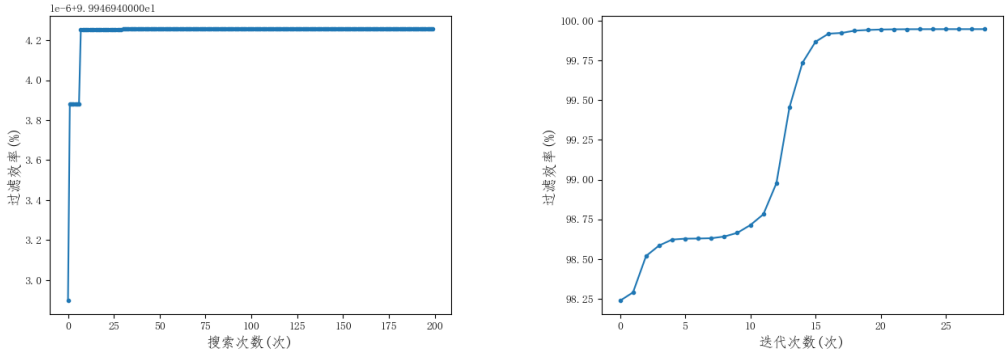
1、使用第一组初值组合进行非线性规划，得到局部最优解 max。
2、判断是否存在未被使用的初值组合，如果有跳转至步骤 3，没有跳转至步骤 6。
3、继续使用下一组初值组合进行局部最优解的求解。
4、如果步骤 3 中得到的结果比 max 大，则将 max 的值替换为步骤 3 得到的结果。
　　如果 max 更大则不变。
5、跳转至步骤 2。
6、结束。

本文将每次搜索得到 max 值进行输出，得到 max 值随搜索次数的变化，见下图 9a所示。

对于单次搜索过程，取初值为 $x = 10, y = 1000$ 作为例子，对 minimize 函数的迭代求解最优值的过程进行可视化，结果如下图 9b。

根据图 9a可知，搜索次数达到 40 次附近时，累计过滤效率达到最大。根据图 9b可知，在迭代次数达到 18 次附近时，该模型收敛到所求过滤效率的最大值。

根据对多组初始值得到的结果进行统计分析，得到以下结果：过滤效率的最大值为 99.9469%，与参考文献 [2] 中的实验数据基本吻合，此时接收距离为 10.1720cm，热空气速度为 2000r/min。



(a) 整体收敛情况　　(b) 收敛性迭代次数

图 9　非线性规划图组

5.4 问题四模型的建立与求解

5.4.1 非线性规划模型的建立与求解

a. 确定目标函数

Step1. 单个目标函数确定

根据问题分析，该问题存在两个目标函数，对于过滤效率最大这一目标函数，本文继续使用问题三中得到的二元阻滞增长模型和拟合参数作为过滤效率与两个工艺参数的转化关系式；对于过滤阻力最小这一目标函数，本文采用非线性回归分析对过滤阻力与两个工艺参数之间的关系进行回归分析。

本文采用以下非线性方程进行回归分析：

$$H(x,y) = p_{00} + p_{10}x + p_{01}y + p_{20}x^2 + p_{11}xy \tag{39}$$

其中，$H(x,y)$ 表示过滤阻力，x 表示接收距离，y 表示热空气速度，p 表示各项系数。

通过 Python 编程求解具体的系数和拟合优度 R^2，得到以下结果，见表 17。

表 17　过滤阻力回归分析结果

p_{00}	p_{10}	p_{01}	p_{20}	p_{11}	R^2
4.0595e+1	8.4663e-1	-1.9368e-2	-2.2970e-2	1.3651e-4	0.8664

根据上表可知，拟合优度 R^2 大于 86%，可以认为得到的回归方程有较好拟合效果，可以很好地反映过滤阻力与两个工艺参数之间的线性相关关系。由此本文确定过滤阻力与两个工艺参数之间的转换关系式为：

$$H(x,y) = 40.595 + 0.847x - 0.019y - 0.023x^2 + 1.3651 \times 10^{-4}xy \tag{40}$$

经过上述分析，得到两个目标函数如下：

$$\begin{aligned} &\max_{x,y} E(x,y) = \frac{25}{1+1.5e^{-1.59x}} + \frac{25}{1+1.5e^{-\frac{-3.2794}{1000}y}} + \frac{50}{1+1.5e^{0.5604x-0.0108y}} \\ &\max_{x,y} H(x,y) = 40.595 + 0.847x - 0.019y - 0.023x^2 + 1.3651 \times 10^{-4}xy \end{aligned} \tag{41}$$

Step2. 标准化处理

由于本文采取过滤效率与过滤阻力的线性组合作为最后的单目标函数，为了消除两者之间的量纲和数量级的差异，本文对过滤效率和过滤阻力的原始值进行标准化处理，其计算公式如下：

$$\begin{cases} \tilde{E} = \frac{E}{E_m} \\ \tilde{H} = \frac{H}{H_m} \end{cases} \tag{42}$$

其中，$\tilde{E}, \tilde{H}$ 为标准化后的数据，E_m, H_m 为两者的平均实验数据（表 data3[#]）中的最大值，其数值分别为 82.98% 和 36.47Pa。

Step3. 确定最终的单目标函数

题目要求过滤效率尽量高，过滤阻力尽量小，为双目标规划。由于经过标准化处理后的标准化过滤效率 $\tilde{E}$ 和标准化过滤阻力 $\tilde{H}$ 具有可比性，本文通过线性组合的方式将其转化成单目标规划。

由于题目中没有交代两者的重要程度，故在此取 $\tilde{E}$ 和 $\tilde{H}$ 为同等重要，分别取加权系数为 0.5 和 0.5。

综上，结合式 (41) 和式 (42)，得到最后的单目标函数：

$$\begin{aligned} \max_{x,y} z = & \frac{0.5}{82.98}\left(\frac{25}{1+1.5e^{-1.59x}} + \frac{25}{1+1.5e^{-\frac{-3.2794}{1000}y}} + \frac{50}{1+1.5e^{0.5604x-0.0108y}}\right) \\ & - \frac{0.5}{36.47}\left(40.595 + 0.847x - 0.019y - 0.023x^2 + 1.3651\times 10^{-4}xy\right) \end{aligned} \tag{43}$$

b. 确定约束条件

根据问题分析可知，题目已经给出了约束条件，本文利用题目给出的工艺参数的限制，并将厚度和压缩回弹性率的约束利用问题二中的回归分析模型转化成相应的工艺参数约束，得到以下约束条件：

$$\begin{cases} 0 \le x \le 100 \\ 0 \le y \le 2000 \\ 0 \le -0.8602 + 0.0542x + 0.0018y \le 3 \\ 85 \le 43.958 + 1.1539x + 0.0588y - 0.0207x^2 + 2.0664\times 10^{-5}xy - 3.1080\times 10^{-5}y^2 \le 100 \end{cases} \tag{44}$$

根据上述的约束条件，本文对其约束出的可行域进行绘制，得到以下图 10。

根据上图可以看出，该约束条件中只有部分是有效约束，本文对其进行等价化简，得到以下新的约束条件：

$$\begin{cases} 15 \le x \le 45 \\ 600 \le y \le 1300 \\ -0.8602 + 0.0542x + 0.0018y \le 3 \\ 43.958 + 1.1539x + 0.0588y - 0.0207x^2 + 2.0664\times 10^{-5}xy - 3.1080\times 10^{-5}y^2 \ge 85 \end{cases} \tag{45}$$

本文使用上述新的约束条件进行接下来的非线性规划。

c. 非线性规划求解

根据上述目标函数和约束条件，本文对以下非线性规划问题进行求解：

$$\begin{aligned} & \max_{x,y} z = \tfrac{0.5}{82.98}\left(\tfrac{25}{1+1.5e^{-1.59x}} + \tfrac{25}{1+1.5e^{-\frac{-3.2794}{1000}y}} + \tfrac{50}{1+1.5e^{0.5604x-0.0108y}}\right) \\ & \quad - \tfrac{0.5}{36.47}\left(40.595 + 0.847x - 0.019y - 0.023x^2 + 1.3651\times 10^{-4}xy\right) \\ & \text{St.} \begin{cases} 15 \le x \le 45 \\ 600 \le y \le 1300 \\ -0.8602 + 0.0542x + 0.0018y \le 3 \\ 43.958 + 1.1539x + 0.0588y - 0.0207x^2 + 2.0664\times 10^{-5}xy - 3.1080\times 10^{-5}y^2 \ge 85 \end{cases} \end{aligned} \tag{46}$$

本文使用 python 中的 minimize 函数，对上述非线性规划问题进行求解。

由于 minimize 函数初值的设置对其收敛性有较大影响，本文在约束条件确定的范围内以固定间隔选取共 700 组初值组合，并以相应的初值组合进行搜索，算法见 section 5 . 3 . 7。得到 max 值随搜索次数的变化, 见下图 11a所示。

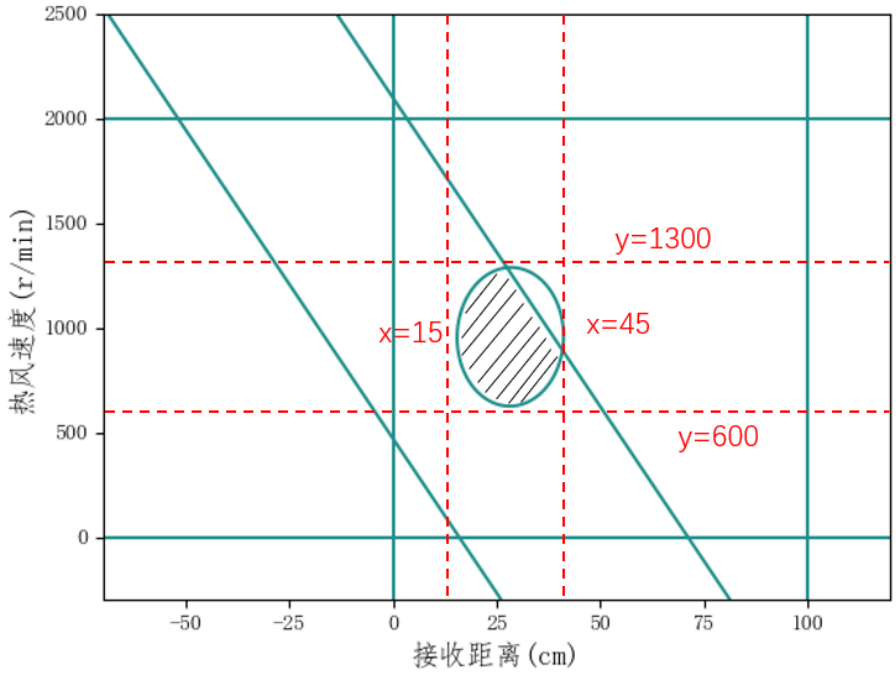
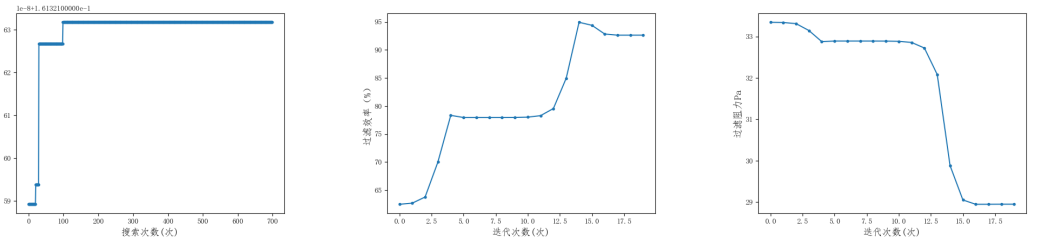


图 10　可行域



(a) 整体收敛情况　**(b) 过滤效率收敛性迭代次数**　**(c) 过滤阻力收敛性迭代次数**

图 11　非线性规划图组

对于单次搜索过程，取初值为 $x=18, y=890$ 作为例子，对 minimize 函数的迭代求解最优值的过程进行可视化，结果如下图 11b和图 11c。

根据图 11a可知，搜索次数达到 110 次附近时，累计规划目标函数值达到最大。根据图 11b可知，在迭代次数达到 14 次附近时，该模型收敛求解到过滤效率的最大值，但是根据图 11c可知，此时的过滤阻力远远没有达到最小，于是该模型继续求解能够使得过滤效率尽量大同时过滤阻力尽量小的最优解，得到最后的该组初始值下的最优解。

根据对多组初始值得到的结果进行统计分析，得到以下最终结果：当过滤效率尽量大同时过滤阻力尽量小时，过滤效率为 92.6366%，过滤阻力为 28.9473Pa，接收距离为 17.9781cm，热空气速度为 1149.6201r/min。

此外，本文通过查阅相关文献 [2]，得知：接收距离小时，纤维组成网帘时的温度高，冷却固化不充分，会使得纤维之间的粘结点增多且粘结效果变好，其结构会更加密实，使得过滤阻力增大，过滤效率提高；同时热风速度大时，对纤维的牵伸力大，从而纤维变细，纤网的致密性提高，对固体颗粒的阻拦效果增强，使得过滤阻力增大，过滤效率提高。这与本文得到的结论一致。

六、 模型的检验

问题四非线性规划模型的灵敏度分析

由于上述非线性规划模型在确定双目标规划转单目标规划问题的权值时，对过滤效率和过滤阻力的权值赋值为 0.5 和 0.5，即认为两者同等重要。此处通过改变权值，将过滤效率权重分别赋值为 $\omega = 0.2, 0.3, \cdots 0.7, 0.8$ ，相对应的，过滤阻力权重赋值为 $1-\omega = 0.8, 0.7, \cdots 0.3, 0.2$。得到目标函数为:

$$\begin{aligned}\max_{x,y} z = & \frac{\omega}{82.98}\left(\frac{25}{1+1.5e^{-1.59x}} + \frac{25}{1+1.5e^{-3.2794\times10^{-3}y}} + \frac{50}{1+1.5e^{0.5604x-1.0766\times10^{-2}y}}\right) \\ & - \frac{1-\omega}{36.47}\left(40.595 + 0.8466x - 0.01937y - 0.02297x^2 + 0.0001365xy\right)\end{aligned} \tag{47}$$

本文对 ω 以及 $1-\omega$ 的组合进行标号，便于后续分组进行检验，结果见下

表 18　组合标号

组号	ω	$1-\omega$
1	0.2	0.8
2	0.3	0.7
3	0.4	0.6
4	0.5	0.5
5	0.6	0.4
6	0.7	0.3
7	0.8	0.2

利用问题四相同的约束条件，求解出对应的目标函数 z 取得最大值时的 x, y，分别代入式 (10) (35) (40)，得到按表 18 给予两者不同权重时规划求解出的对应指标数值，如下图 12所示。

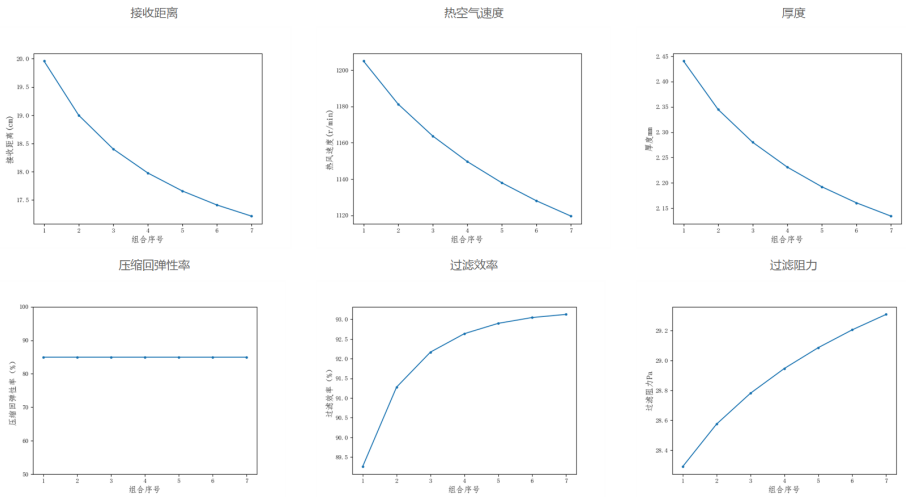


图 12　灵敏度分析组图

可以看出，在权重发生变化时，求出的不同参数和指标的值变化都非常平稳，证明本文采用模型的稳定性和可靠性。

进一步观察变量的变化趋势可以得知。

随着对过滤效率的重视程度的提升和对过滤阻力的重视程度的下降，考虑如下决策变量和约束条件中提到的指标的变化。

接收距离

接收距离会逐渐减小，从 20cm 左右逐渐减小到 17cm 左右，且减小程度逐渐放缓。

热空气速度

需要提供的热空气速度也逐渐减小，从 1200r/min 逐渐减小到 1120r/min，减小的速度相对稳定。

厚度

据图可知，得到的材料厚度逐渐变小，从 2.45mm 左右逐渐减小到 2.15mm 左右，减小的速度相对稳定。

压缩回弹性率

由图可以看出，压缩回弹性几乎不受权重变化的影响，稳定保持在 85% 左右的数值，这个数值贴近约束条件的边界。在一定程度上说明了压缩回弹性率受外界条件的影响较小。

过滤效率

随着对于过滤效率的重视程度的提升，明显可以看出过滤效率的数值越来越大，符合预期。

过滤阻力

随着对于过滤阻力的重视程度的下降，明显可以看出过滤阻力的数值越来越小，符合预期。

以上过滤效率和过滤阻力的变化趋势都验证了整个求解模型的合理性。

七、 模型的评价、改进与推广

7.1 模型的评价

7.1.1 模型的优点

1、本文在问题一中使用多因素方差分析模型，不仅对插层率对于结构变量和产品性能改变的单独作用进行定性和定量分析，还对插层率与其他工艺参数交互作用进行了定性和定量研究。比使用多元非线性回归分析模型更加全面，且解释性更强。

2、本文在问题二中使用传统的非线性回归分析模型，考虑全面，并且相较于机器学习模型，对该问题的内部机理有更好的反映。

3、本文在问题三中使用主成分分析方法进行降维，使自变量数量减少，更加有利于回归分析。且结合实际情况分段处理回归，使拟合优度可以得到很大提升。

4、本文观察工艺参数和过滤效率变化趋势，对传统的一元阻滞增长模型进行改进，结合过滤效率实际上不能大于 100% 的现实意义，得到二元的阻滞增长模型，并拟合参数，得到的拟合效果很好，可以准确地描述整个范围内的过滤效率随工艺参数的变化规律。

5、本文在问题四将过滤效率和过滤阻力的值进行标准化处理，使得组合出来的目标函数有意义。

6、本文根据绘制出可行域范围的方式缩小了实际的约束范围，体现出了优化的过程并且可以减轻求解的负担。

7.1.2 模型的缺点

1、由于题目的数据量有限，通过模型得到的规律与实际情况可能存在一定偏差。

7.2 模型的改进

可以通过多做实验，获得更多的实验数据，使模型得到的规律更加符合实际情况。

7.3 模型的推广

本文建立的多因素方差分析模型对于研究多个因素对另一个因素的影响问题具有一定的参考价值，可推广至商品采购、农作物的生长状况等相关领域的研究。

本文使用的二元阻滞增长模型，实用性强，可以推广至两个自变量影响一个因变量的情况，如：化学实验、种群数量预测等问题中。

参考文献

[1] 邹志伟, “插层熔喷气流场模拟及生产工艺参数的优化,” Master’s thesis, 天津工业大学, 2020.

[2] 武辉, “插层熔喷气流场模拟及其过滤材料性能的研究,” Master’s thesis, 天津工业大学, 2018.

[3] 司守奎，孙玺菁, Python 数学实验与建模. SCIENCE PRESS, 2020.

附录 A　支撑材料文件结构

```
submit
├─灵敏性检验
│    ├─46 权重检验.py
│    ├─灵敏性检验结果.txt
│    ├─绘图.py
│    └─计算具体值.py
├─第一问
│    ├─C 题数据.xlsx
│    ├─C 题数据填充.xlsx
│    ├─Wilcox 检验
│    ├─多因素方差分析
│    └─散点图
├─第三问
│    ├─主成分分析
│    ├─正态分布检验
│    ├─热图
│    ├─第三小问二维绘图
│    ├─结构变量与产品性能关系
│    └─阻滞增长
├─第二问
│    ├─C 题数据填充.xlsx
│    ├─data3_ 接受距离.csv
│    ├─data3_ 热风速度.csv
│    ├─data3 求平均.csv
│    ├─拟合
│    ├─数据求平均.py
│    └─绘图
└─第四问
     ├─拟合工艺参数和过滤阻力
     ├─搜索最值
     └─隐函数图像.py
```

附录 B　多因素方差分析结果

对厚度的多因素方差分析结果

	df	sum_sq	mean_sq	F	PR(>F)
插层率	1	11.49967	11.49967	92.87981	2.62E-12
接收距离	1	6.039504	6.039504	48.7795	1.36E-08
热风速度	1	1.852558	1.852558	14.96263	3.67E-04
插层率: 接收距离	1	0.580422	0.580422	4.68792	3.60E-02
插层率: 热风速度	1	0.149492	0.149492	1.207409	2.78E-01
接收距离: 热风速度	1	0.113371	0.113371	0.915666	3.44E-01
Residual	43	5.323931	0.123812	NaN	NaN

对孔隙率的多因素方差分析结果

	df	sum_sq	mean_sq	F	PR(>F)
插层率	1	110.3245	110.3245	54.62621	3.51E-09
接收距离	1	60.16266	60.16266	29.78901	2.23E-06
热风速度	1	16.54115	16.54115	8.190203	6.48E-03
插层率: 接收距离	1	2.169778	2.169778	1.074346	3.06E-01
插层率: 热风速度	1	3.258312	3.258312	1.613324	2.11E-01
接收距离: 热风速度	1	0.024292	0.024292	0.012028	9.13E-01
Residual	43	86.84393	2.019626	NaN	NaN

对压缩回弹性率的多因素方差分析结果

	df	sum_sq	mean_sq	F	PR(>F)
插层率	1	122.2899	122.2899	3.405476	0.071874
接收距离	1	10.20145	10.20145	0.284085	0.59678
热风速度	1	0.367287	0.367287	0.010228	0.919914
插层率: 接收距离	1	9.245381	9.245381	0.257461	0.614462
插层率: 热风速度	1	34.42961	34.42961	0.958781	0.332971
接收距离: 热风速度	1	7.344746	7.344746	0.204533	0.65336
Residual	43	1544.121	35.90979	NaN	NaN

对过滤阻力的多因素方差分析结果

	df	sum_sq	mean_sq	F	PR(>F)
插层率	1	41.09165	41.09165	2.627437	1.12E-01
接收距离	1	7182.503	7182.503	459.2557	1.42E-24
热风速度	1	3510.407	3510.407	224.4586	1.13E-18
插层率: 接收距离	1	20.5303	20.5303	1.312726	2.58E-01
插层率: 热风速度	1	6.344145	6.344145	0.40565	5.28E-01
接收距离: 热风速度	1	711.9631	711.9631	45.52356	2.99E-08
Residual	43	672.4959	15.63944	NaN	NaN

对过滤效率的多因素方差分析结果

	df	sum_sq	mean_sq	F	PR(>F)
插层率	1	2886.152	2886.152	78.57175	2.94E-11
接收距离	1	11078.94	11078.94	301.6099	4.76E-21
热风速度	1	9649.713	9649.713	262.7009	6.31E-20
插层率: 接收距离	1	817.0037	817.0037	22.24187	2.55E-05
插层率: 热风速度	1	21.88929	21.88929	0.595907	4.44E-01
接收距离: 热风速度	1	263.7324	263.7324	7.179773	1.04E-02
Residual	43	1579.506	36.7327	NaN	NaN

对透气性的多因素方差分析结果

	df	sum_sq	mean_sq	F	PR(>F)
插层率	1	4.02E+04	4.02E+04	9.085666	4.31E-03
接收距离	1	1.17E+06	1.17E+06	263.3661	6.02E-20
热风速度	1	7.31E+05	7.31E+05	164.98	2.59E-16
插层率: 接收距离	1	1.83E+04	1.83E+04	4.121214	4.86E-02
插层率: 热风速度	1	1.23E+04	1.23E+04	2.784809	1.02E-01
接收距离: 热风速度	1	1.19E+05	1.19E+05	26.93348	5.43E-06
Residual	43	1.90E+05	4.43E+03	NaN	NaN

附录 C　部分代码

对厚度进行 Wilcox 检验

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import scipy.stats as ss

data = pd.read_excel("../C题数据填充.xlsx", sheet_name=0)
data = data.values
plt.rcParams['font.sans-serif'] = ['FangSong']

# print(data)


pre = np.zeros([25])
lat = np.zeros([25])

for i in range(25):
    pre[i] = data[2 * i, 2]
    lat[i] = data[2 * i + 1, 2]

# print(prehoudu)
# print(lathoudu)

# myx=np.arange(25)
#
# plt.title("厚度",fontsize=20)
#
# ax=plt.plot(myx, pre, c='red', marker='.')
# bx=plt.plot(myx, lat, c='blue', marker='.')
# plt.show()

stat, p = ss.ranksums(pre, lat)
print(ss.ranksums(pre, lat))

# RanksumsResult(statistic=-5.539513675829845, pvalue=3.0331276062597446e-08)
```

孔隙率多因素方差分析

```
import numpy as np
from statsmodels.stats.anova import anova_lm
from statsmodels.formula.api import ols
import pandas as pd
from statsmodels.stats.multicomp import pairwise_tukeyhsd

data=pd.read_excel("C题数据填充.xlsx")
data=data.values

for i in range(50):
    if data[i, 8] != 0:
        data[i, 8] = int(data[i, 8] / 10 + 1)
    data[i,9]=(int(data[i,9]-20))/5
    data[i, 10] = (int(data[i, 10]) - 800) / 100
    if(data[i,8]==6):
        data[i,8]=5


predata=np.zeros([50,4])

predata[:,0]=data[:,3]
```

```
predata[:,1]=data[:,8]
predata[:,2]=data[:,9]
predata[:,3]=data[:,10]

predata=pd.DataFrame(predata,columns=['孔隙率','插层率','接收距离','热风速度'])

model = ols("孔隙率
    ~插层率+接收距离+热风速度+插层率:接收距离+插层率*热风速度+接收距离*热风速度+插层率*接收距离*热风速度",
    data=predata)
data = model.fit()
print(anova_lm(data))

# hsd=pairwise_tukeyhsd(predata['孔隙率'],predata['插层率'])
# print(hsd)
```

透气性事后分析

```
import numpy as np
from statsmodels.stats.anova import anova_lm
from statsmodels.formula.api import ols
import pandas as pd
from statsmodels.stats.multicomp import pairwise_tukeyhsd

data=pd.read_excel("C题数据填充.xlsx")
data=data.values

for i in range(50):
    if data[i, 8] != 0:
        data[i, 8] = int(data[i, 8] / 10 + 1)
    data[i,9]=(int(data[i,9]-20))/5
    data[i, 10] = (int(data[i, 10]) - 800) / 100
    if(data[i,8]==6):
        data[i,8]=5



predata=np.zeros([50,4])

predata[:,0]=data[:,7]


predata[:,1]=data[:,8]
predata[:,2]=data[:,9]
predata[:,3]=data[:,10]

predata=pd.DataFrame(predata,columns=['透气性','插层率','接收距离','热风速度'])

model = ols("透气性
    ~插层率+接收距离+热风速度+插层率:接收距离+插层率*热风速度+接收距离*热风速度+插层率*接收距离*热风速度",
    data=predata)
data = model.fit()
print(anova_lm(data))

# hsd=pairwise_tukeyhsd(predata['透气性'],predata['插层率'])
# print(hsd)

hsd = pairwise_tukeyhsd(groups=predata['插层率'], endog=predata['透气性']).summary()

print(hsd)
```

透气性散点图绘制

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd

data=pd.read_excel("../C题数据填充.xlsx",sheet_name=0)
data=data.values
plt.rcParams['font.sans-serif']=['FangSong']

# print(data)


pre=np.zeros([25])
lat=np.zeros([25])

for i in range(25):
    pre[i]=data[2 * i, 7]
    lat[i]=data[2 * i + 1, 7]

# print(prehoudu)
# print(lathoudu)

myx=np.arange(25)

plt.title("透气性",fontsize=20)

ax=plt.plot(myx, pre, c='red', marker='.')
bx=plt.plot(myx, lat, c='blue', marker='.')
plt.show()

res=np.zeros([2,25])
res[0,:]=pre[:]
res[1,:]=lat[:]
print(res)
np.savetxt("透气性数据.csv",res,delimiter=',')
```

第二问厚度

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from scipy.optimize import curve_fit
plt.rcParams['font.sans-serif'] = ['FangSong']

x = pd.read_csv("data3_接受距离.csv", header=None)
x = x.values

y = pd.read_csv("data3_热风速度.csv", header=None)
y = y.values

myx=np.zeros([2,25])
myx[0]=x.flatten()
myx[1]=y.flatten()

z = pd.read_csv("data3求平均.csv", header=None)
z = z.values
myz=z[:,0]
myz=myz.flatten()

# plt.plot(myx[0],myz,c='red')


def fun(x,k0,k1,k2):
    return k0*x[0,:]+k1*x[1,:]+k2
```

```
popt, pcov=curve_fit(fun, myx, myz)


y2=np.zeros([25])
for i in range(25):
    y2[i]=popt[0]*myx[0,i]+popt[1]*myx[1,i]+popt[2]

# plt.plot(myx[0],y2,c='blue')
# plt.show()
print(popt)


def __sst(y_no_fitting):
    y_mean = sum(y_no_fitting) / len(y_no_fitting)
    s_list =[(y - y_mean)**2 for y in y_no_fitting]
    sst = sum(s_list)
    return sst


def __ssr(y_fitting, y_no_fitting):
    y_mean = sum(y_no_fitting) / len(y_no_fitting)
    s_list =[(y - y_mean)**2 for y in y_fitting]
    ssr = sum(s_list)
    return ssr


def __sse(y_fitting, y_no_fitting):
    s_list = [(y_fitting[i] - y_no_fitting[i])**2 for i in range(len(y_fitting))]
    sse = sum(s_list)
    return sse


def goodness_of_fit(y_fitting, y_no_fitting):
    SSR = __ssr(y_fitting, y_no_fitting)
    SST = __sst(y_no_fitting)
    rr = SSR /SST
    return rr

print(goodness_of_fit(y2,myz))
# print(pcov)

p00=popt[0]
p10=popt[1]
p01=popt[2]
fig = plt.figure()
ax = plt.axes(projection="3d")

plotx=np.arange(start=20,stop=40,step=0.5)
ploty=np.arange(start=800,stop=1200,step=10)
plotx=plotx.reshape([-1,1])
# ploty=ploty.reshape([-1,1])
Z=popt[0]*plotx+popt[1]*ploty+popt[2]

ax.plot_surface(plotx,ploty,Z,alpha=0.8, cstride=1, rstride = 1, cmap='copper')

ax.scatter(myx[0],myx[1],myz,marker='.')



pre=pd.read_excel("pre.xlsx",header=None)
pre=pre.values

pre_results=np.zeros([8])
for i in range(8):
```

```
    x=pre[i,0]
    y=pre[i,1]
    pre_results[i]=popt[0]*x+popt[1]*y+popt[2]

ax.scatter(pre[:,0],pre[:,1],pre_results,marker='.',c='darkred')
ax.set_zlabel('厚度（mm）',fontsize=13)
# ax.view_init(0, 10)
plt.xlabel('接收距离(cm)',fontsize=13)
plt.ylabel('热风速度(r/min)',fontsize=13)
plt.show()


np.savetxt("厚度预测结果.csv",pre_results,delimiter=',')
```

第三问过滤效率-透气性绘图

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

data = pd.read_csv("data3求平均.csv", header=None)
data = data.values

x=data[:,4]
x=x.flatten()
y=data[:,5]
y=y.flatten()
def fun(X,p1,p2):
    return p1*X+p2


popt, pcov=curve_fit(fun, x, y)

p1=popt[0]
p2=popt[1]



y2=np.zeros([25])

for i in range(25):
    y2[i]=p1*x[i]+p2
print(popt)

# ##################################拟合优度R^2的计算#######################################
def __sst(y_no_fitting):
    """
    计算SST(total sum of squares) 总平方和
    :param y_no_predicted: List[int] or array[int] 待拟合的y
    :return: 总平方和SST
    """
    y_mean = sum(y_no_fitting) / len(y_no_fitting)
    s_list =[(y - y_mean)**2 for y in y_no_fitting]
    sst = sum(s_list)
    return sst


def __ssr(y_fitting, y_no_fitting):
    """
    计算SSR(regression sum of squares) 回归平方和
    :param y_fitting: List[int] or array[int] 拟合好的y值
    :param y_no_fitting: List[int] or array[int] 待拟合y值
    :return: 回归平方和SSR
```

```
    """
    y_mean = sum(y_no_fitting) / len(y_no_fitting)
    s_list =[(y - y_mean)**2 for y in y_fitting]
    ssr = sum(s_list)
    return ssr


def __sse(y_fitting, y_no_fitting):
    """
    计算SSE(error sum of squares) 残差平方和
    :param y_fitting: List[int] or array[int] 拟合好的y值
    :param y_no_fitting: List[int] or array[int] 待拟合y值
    :return: 残差平方和SSE
    """
    s_list = [(y_fitting[i] - y_no_fitting[i])**2 for i in range(len(y_fitting))]
    sse = sum(s_list)
    return sse


def goodness_of_fit(y_fitting, y_no_fitting):
    """
    计算拟合优度R^2
    :param y_fitting: List[int] or array[int] 拟合好的y值
    :param y_no_fitting: List[int] or array[int] 待拟合y值
    :return: 拟合优度R^2
    """
    SSR = __ssr(y_fitting, y_no_fitting)
    SST = __sst(y_no_fitting)
    rr = SSR /SST
    return rr

print(goodness_of_fit(y2,y))



plotx=np.arange(start=40,stop=85,step=1)

ploty=p1*plotx+p2

plt.plot(plotx,ploty)
plt.rcParams['font.sans-serif'] = ['FangSong']
plt.rcParams['axes.unicode_minus'] = False
plt.scatter(x,y,marker='.')
plt.xlabel('过滤效率 (%) ',fontsize=13)
plt.ylabel('透气性 mm/s',fontsize=13)
plt.show()
```

第三问结构变量与过滤效率关系

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

data = pd.read_excel("主成分分析.xlsx", header=None)

data = data.values

myx1 = data[:, 0]

myx2 = data[:, 1]

myX = np.zeros([2, 25])
myX[0, :] = myx1
```

```
myX[1, :] = myx2

target = pd.read_csv("data3求平均.csv", header=None)
target = target.values

my_y = target[:, 4]

my_y = my_y[:22]
myX = myX[:, :22]


def fun(X, p00, p10, p01, p20, p11):
    x = X[0, :]
    y = X[1, :]

    return p00 + p10 * x + p01 * y + p20 * x ** 2 + p11 * x * y


popt, pcov = curve_fit(fun, myX, my_y)

p00 = popt[0]
p10 = popt[1]
p01 = popt[2]
p20 = popt[3]
p11 = popt[4]
y2 = np.zeros([22])

for i in range(22):
    x = myX[0, i]
    y = myX[1, i]
    y2[i] = p00 + p10 * x + p01 * y + p20 * x ** 2 + p11 * x * y

print(popt)


# ##################################拟合优度R^2的计算#########################################
def __sst(y_no_fitting):
    """
    计算SST(total sum of squares) 总平方和
    :param y_no_predicted: List[int] or array[int] 待拟合的y
    :return: 总平方和SST
    """
    y_mean = sum(y_no_fitting) / len(y_no_fitting)
    s_list = [(y - y_mean) ** 2 for y in y_no_fitting]
    sst = sum(s_list)
    return sst


def __ssr(y_fitting, y_no_fitting):
    """
    计算SSR(regression sum of squares) 回归平方和
    :param y_fitting: List[int] or array[int] 拟合好的y值
    :param y_no_fitting: List[int] or array[int] 待拟合y值
    :return: 回归平方和SSR
    """
    y_mean = sum(y_no_fitting) / len(y_no_fitting)
    s_list = [(y - y_mean) ** 2 for y in y_fitting]
    ssr = sum(s_list)
    return ssr


def __sse(y_fitting, y_no_fitting):
    """
    计算SSE(error sum of squares) 残差平方和
```

```
    :param y_fitting: List[int] or array[int] 拟合好的y值
    :param y_no_fitting: List[int] or array[int] 待拟合y值
    :return: 残差平方和SSE
    """
    s_list = [(y_fitting[i] - y_no_fitting[i]) ** 2 for i in range(len(y_fitting))]
    sse = sum(s_list)
    return sse


def goodness_of_fit(y_fitting, y_no_fitting):
    """
    计算拟合优度R^2
    :param y_fitting: List[int] or array[int] 拟合好的y值
    :param y_no_fitting: List[int] or array[int] 待拟合y值
    :return: 拟合优度R^2
    """
    SSR = __ssr(y_fitting, y_no_fitting)
    SST = __sst(y_no_fitting)
    rr = SSR / SST
    return rr


print(goodness_of_fit(y2, my_y))

fig = plt.figure()
ax = plt.axes(projection="3d")

plotx = np.arange(start=-2.5, stop=1.5, step=0.1)
ploty = np.arange(start=-2, stop=2, step=0.1)
plotx = plotx.reshape([-1, 1])
Z = p00 + p10 * plotx + p01 * ploty + p20 * plotx ** 2 + p11 * plotx * ploty
ax.plot_surface(plotx, ploty, Z, alpha=0.8, cstride=1, rstride=1, cmap='copper')
ax.scatter(myX[0], myX[1], my_y, marker='.')
ax.view_init(30, 30)

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

ax.set_zlabel('过滤效率 (%) ', fontsize=13)
# ax.view_init(0, 10)
plt.xlabel('主成分1', fontsize=13)
plt.ylabel('主成分2', fontsize=13)
plt.show()
```

第三问产品性能热图

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from scipy.stats import spearmanr, pearsonr

plt.rcParams['font.sans-serif'] = ['FangSong']
plt.rcParams['axes.unicode_minus'] = False
z = pd.read_csv("data3求平均.csv", header=None)
z = z.values

data = z[:, 3:6]

res = pd.DataFrame(data, columns=['过滤阻力',
                                  '过滤效率',
                                  '透气性'])

corr = res.corr(method='pearson')
```

```
sns.heatmap(corr, annot=True, vmax=1, vmin=-1, xticklabels=True, yticklabels=True,
            square=True, cmap="YlGnBu")

plt.show()


print(pearsonr(res['过滤阻力'], res['过滤效率']))
print(pearsonr(res['过滤阻力'], res['透气性']))
print(pearsonr(res['过滤效率'], res['透气性']))
```

主成分分析

```
import numpy as np
import pandas as pd
from numpy.linalg import eig
from sklearn.datasets import load_iris
from sklearn.decomposition import PCA

data = pd.read_csv("data3求平均.csv", header=None)
data = data.values
x = data[:, 0:3]

for i in range(25):
    x[i, 0] = (x[i, 0] - 2.6076) / 0.4792901
    x[i, 1] = (x[i, 1] - 95.88) / 0.798759455
    x[i, 2] = (x[i, 2] - 86.6072) / 1.218528894

x_pca = np.zeros([25, 2])

for i in range(25):
    x_pca[i, 0] = -0.650110206 * x[i, 0] - 0.612391423 * x[i, 1] + 0.449814923 * x[i, 2]
    x_pca[i, 1] = 0.190957569 * x[i, 0] + 0.441310513 * x[i, 1] + 0.87680114 * x[i, 2]

print(x_pca)
```

逻辑增长模型

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

x = pd.read_csv("data3_接受距离.csv", header=None)
x = x.values

y = pd.read_csv("data3_热风速度.csv", header=None)
y = y.values

myx = np.zeros([2, 25])
myx[0] = x.flatten() / 10
myx[1] = y.flatten() / 1000

z = pd.read_csv("data3求平均.csv", header=None)
z = z.values
myz = z[:, 4]
myz = myz.flatten()


def fun(X, k1, k2, k3, k4):
    x = X[0, :]
    y = X[1, :]
    return 25 / (1 + 1.5 * np.exp(-k1 * x)) + 25 / (1 + 1.5 * np.exp(-k2 * y)) + 50 / (
```

```
          1 + 1.5 * np.exp(-k3 * x - k4 * y))


init = [15.9, 3.281, -5.605, 10.77]
# init = [[15.9, 2.29426109, -3.9162163, 6.95940933]]
popt, pcov = curve_fit(fun, myx, myz, p0=init)

print(popt)

y2 = np.zeros([25])
k1 = popt[0]
k2 = popt[1]
k3 = popt[2]
k4 = popt[3]

for i in range(25):
    x = myx[0, i]
    y = myx[1, i]
    y2[i] = 25 / (1 + 1.5 * np.exp(-k1 * x)) + 25 / (1 + 1.5 * np.exp(-k2 * y)) + 50 / (
          1 + 1.5 * np.exp(-k3 * x - k4 * y))


# ##################################拟合优度R^2的计算#######################################
def __sst(y_no_fitting):
    """
    计算SST(total sum of squares) 总平方和
    :param y_no_predicted: List[int] or array[int] 待拟合的y
    :return: 总平方和SST
    """
    y_mean = sum(y_no_fitting) / len(y_no_fitting)
    s_list = [(y - y_mean) ** 2 for y in y_no_fitting]
    sst = sum(s_list)
    return sst


def __ssr(y_fitting, y_no_fitting):
    """
    计算SSR(regression sum of squares) 回归平方和
    :param y_fitting: List[int] or array[int] 拟合好的y值
    :param y_no_fitting: List[int] or array[int] 待拟合y值
    :return: 回归平方和SSR
    """
    y_mean = sum(y_no_fitting) / len(y_no_fitting)
    s_list = [(y - y_mean) ** 2 for y in y_fitting]
    ssr = sum(s_list)
    return ssr


def __sse(y_fitting, y_no_fitting):
    """
    计算SSE(error sum of squares) 残差平方和
    :param y_fitting: List[int] or array[int] 拟合好的y值
    :param y_no_fitting: List[int] or array[int] 待拟合y值
    :return: 残差平方和SSE
    """
    s_list = [(y_fitting[i] - y_no_fitting[i]) ** 2 for i in range(len(y_fitting))]
    sse = sum(s_list)
    return sse


def goodness_of_fit(y_fitting, y_no_fitting):
    """
    计算拟合优度R^2
    :param y_fitting: List[int] or array[int] 拟合好的y值
```

```
    :param y_no_fitting: List[int] or array[int] 待拟合y值
    :return: 拟合优度R^2
    """
    SSR = __ssr(y_fitting, y_no_fitting)
    SST = __sst(y_no_fitting)
    rr = SSR / SST
    return rr


print(goodness_of_fit(y2, myz))
plt.rcParams['font.sans-serif'] = ['FangSong']
plt.rcParams['axes.unicode_minus'] = False
fig = plt.figure()
ax = plt.axes(projection="3d")

plotx = np.arange(start=0, stop=100, step=1)
ploty = np.arange(start=0, stop=2000, step=20)
plotx = plotx.reshape([-1, 1])

Z = 25 / (1 + 1.5 * np.exp(-k1 * plotx/10)) + 25 / (1 + 1.5 * np.exp(-k2 * ploty/1000)) + 50 /
    (
       1 + 1.5 * np.exp(-k3 * plotx/10 - k4 * ploty/1000))

ax.plot_surface(plotx,ploty,Z,alpha=0.8, cstride=1, rstride = 1, cmap='rainbow')
ax.scatter(myx[0]*10,myx[1]*1000,myz,marker='.',c='black')

ax.set_zlabel('过滤效率（%）',fontsize=13)
ax.view_init(20, -100)
plt.xlabel('接收距离(cm)',fontsize=13)
plt.ylabel('热风速度(r/min)',fontsize=13)
plt.show()
```

第三问收敛性绘图

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd


def obj(X):
    x = X[0]
    y = X[1]
    k1 = 15.9
    k2 = 3.27944218
    k3 = -5.60411769
    k4 = 10.76586634
    return -(25 / (1 + 1.5 * np.exp(-k1 * x / 10)) + 25 / (1 + 1.5 * np.exp(-k2 * y / 1000)) +
        50 / (
           1 + 1.5 * np.exp(-k3 * x / 10 - k4 * y / 1000)))
X=np.zeros([29,2])
X[0,0]=10
X[0,1]=1000
for i in range(28):
    data=pd.read_csv("中间结果"+str(i)+".csv",header=None)
    X[i+1,:]=data.values.flatten()

Y=np.zeros([29,1])

for i in range(29):
    Y[i,0]=obj(X[i,:])
plt.rcParams['font.sans-serif'] = ['FangSong']
plt.rcParams['axes.unicode_minus'] = False
idx=np.arange(29)
plt.plot(idx,-Y,marker='.')
```

```
plt.xlabel('迭代次数(次)',fontsize=13)
plt.ylabel('过滤效率(%)',fontsize=13)
plt.show()
```

第四问工艺参数和过滤阻力关系

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

x = pd.read_csv("data3_接受距离.csv", header=None)
x = x.values

y = pd.read_csv("data3_热风速度.csv", header=None)
y = y.values

myx = np.zeros([2, 25])
myx[0] = x.flatten()
myx[1] = y.flatten()

z = pd.read_csv("data3求平均.csv", header=None)
z = z.values
myz = z[:, 3]
myz = myz.flatten()


def fun(X, p00, p10, p01, p20, p11):
    x = X[0, :]
    y = X[1, :]

    return p00 + p10 * x + p01 * y + p20 * x ** 2 + p11 * x * y


popt, pcov = curve_fit(fun, myx, myz)

p00 = popt[0]
p10 = popt[1]
p01 = popt[2]
p20 = popt[3]
p11 = popt[4]
y2 = np.zeros([25])

for i in range(25):
    x = myx[0, i]
    y = myx[1, i]
    y2[i] = p00 + p10 * x + p01 * y + p20 * x ** 2 + p11 * x * y

print(popt)


# ##################################拟合优度R^2的计算#########################################
def __sst(y_no_fitting):
    """
    计算SST(total sum of squares) 总平方和
    :param y_no_predicted: List[int] or array[int] 待拟合的y
    :return: 总平方和SST
    """
    y_mean = sum(y_no_fitting) / len(y_no_fitting)
    s_list = [(y - y_mean) ** 2 for y in y_no_fitting]
    sst = sum(s_list)
    return sst
```

```
def __ssr(y_fitting, y_no_fitting):
    """
    计算SSR(regression sum of squares) 回归平方和
    :param y_fitting: List[int] or array[int] 拟合好的y值
    :param y_no_fitting: List[int] or array[int] 待拟合y值
    :return: 回归平方和SSR
    """
    y_mean = sum(y_no_fitting) / len(y_no_fitting)
    s_list = [(y - y_mean) ** 2 for y in y_fitting]
    ssr = sum(s_list)
    return ssr


def __sse(y_fitting, y_no_fitting):
    """
    计算SSE(error sum of squares) 残差平方和
    :param y_fitting: List[int] or array[int] 拟合好的y值
    :param y_no_fitting: List[int] or array[int] 待拟合y值
    :return: 残差平方和SSE
    """
    s_list = [(y_fitting[i] - y_no_fitting[i]) ** 2 for i in range(len(y_fitting))]
    sse = sum(s_list)
    return sse


def goodness_of_fit(y_fitting, y_no_fitting):
    """
    计算拟合优度R^2
    :param y_fitting: List[int] or array[int] 拟合好的y值
    :param y_no_fitting: List[int] or array[int] 待拟合y值
    :return: 拟合优度R^2
    """
    SSR = __ssr(y_fitting, y_no_fitting)
    SST = __sst(y_no_fitting)
    rr = SSR / SST
    return rr


print(goodness_of_fit(y2, myz))

fig = plt.figure()
ax = plt.axes(projection="3d")

plotx = np.linspace(start=20, stop=40, num=50)
ploty = np.linspace(start=800, stop=1400, num=50)
plotx = plotx.reshape([-1, 1])

Z = p00 + p10 * plotx + p01 * ploty + p20 * plotx ** 2 + p11 * plotx * ploty

ax.plot_surface(plotx, ploty, Z, alpha=0.8, cstride=1, rstride=1, cmap='copper')
ax.scatter(myx[0], myx[1], myz, marker='.')

plt.rcParams['font.sans-serif'] = ['FangSong']
plt.rcParams['axes.unicode_minus'] = False
ax.set_zlabel('过滤阻力Pa', fontsize=13)
# ax.view_init(0, 10)
plt.xlabel('接收距离(cm)', fontsize=13)
plt.ylabel('热风速度(r/min)', fontsize=13)
plt.show()
```

第四问结果

```
import matplotlib.pyplot as plt
import numpy as np
```

```
from scipy.optimize import minimize


def obj(X):
    x = X[0]
    y = X[1]
    zuli = 4.05950528e+01 + 8.46631161e-01 * x - \
          1.93679792e-02 * y - 2.29699455e-02 * x ** 2 \
          + 1.36511669e-04 * x * y
    zuli /= 36.47

    k1 = 15.9
    k2 = 3.27944218
    k3 = -5.60411769
    k4 = 10.76586634
    xiaolv = 25 / (1 + 1.5 * np.exp(-k1 * x / 10)) \
            + 25 / (1 + 1.5 * np.exp(-k2 * y / 1000)) \
            + 50 / (1 + 1.5 * np.exp(-k3 * x / 10 - k4 * y / 1000))

    xiaolv /= 82.98

    return -0.5 * xiaolv + 0.5 * zuli


myobj = lambda x: obj(x)


# def con1(X):
#     x = X[0]
#     y = X[1]
#     return -0.8648 + 0.05428 * x + 0.001844 * y

def con2(X):
    x = X[0]
    y = X[1]
    return -(-0.8648 + 0.05428 * x + 0.001844 * y - 3)


def con3(X):
    x = X[0]
    y = X[1]
    return 43.96 + 1.149 * x + 0.05897 * y - 0.02063 * x ** 2 + 2e-5 * x * y - 85 - 3.114e-5 *
        y ** 2


mycons = (
    {'type': 'ineq', 'fun': lambda x: con2(x)},
    {'type': 'ineq', 'fun': lambda x: con3(x)}
)
bound = []

bound.append((15, 45))
bound.append((600, 1300))
bound = tuple(bound)
xcall = []

# x0 = np.arange(start=15, stop=45, step=3)
# x1 = np.arange(start=600, stop=1300, step=10)
#
# pre = 900
# preinit = []
# idx = np.arange(700)
# maxidx = np.zeros([700])
#
```

```
# p = 0
# for i in range(len(x0)):
#     for j in range(len(x1)):
#         init = [x0[i], x1[j]]
#         # print(init)
#         res = minimize(myobj, init, bounds=bound, constraints=mycons, options={'disp': True,
    'maxiter': 200},
#                        callback=xcall.append)
#         if pre >= res.fun:
#             preinit = init
#             pre = res.fun
#         maxidx[p] = -pre
#         p += 1
#         # print(res.fun, '\n', res.success, '\n', res.x, res.message) #
    输出最优值、求解状态、最优解
# print(pre)
# print(preinit)
# plt.rcParams['font.sans-serif'] = ['FangSong']
# plt.rcParams['axes.unicode_minus'] = False
#
# plt.xlabel('搜索次数(次)', fontsize=13)
# # plt.ylabel('过滤效率(%)',fontsize=13)
#
# plt.plot(idx, maxidx, marker='.')
#
# plt.show()
#
# res = minimize(myobj, [18, 890], bounds=bound, constraints=mycons, options={'disp': True,
    'maxiter': 200},
#                callback=xcall.append)
# print(res.fun, '\n', res.success, '\n', res.x, res.message) # 输出最优值、求解状态、最优解

x0 = [18, 890]

res = minimize(myobj, x0, bounds=bound, constraints=mycons, options={'disp': True, 'maxiter':
    200},
               callback=xcall.append)

for i in range(len(xcall)):
    np.savetxt("中间结果" + str(i) + ".csv", xcall[i])

print(res.fun, '\n', res.success, '\n', res.x, res.message) # 输出最优值、求解状态、最优解
```

第四问最优解收敛绘图

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
def obj(X):
    x = X[0]
    y = X[1]
    zuli = 4.05950528e+01 + 8.46631161e-01 * x - \
           1.93679792e-02 * y - 2.29699455e-02 * x ** 2 \
           + 1.36511669e-04 * x * y
    zuli /= 36.47

    k1 = 15.9
    k2 = 3.27944218
    k3 = -5.60411769
    k4 = 10.76586634
    xiaolv = 25 / (1 + 1.5 * np.exp(-k1 * x / 10)) \
             + 25 / (1 + 1.5 * np.exp(-k2 * y / 1000)) \
             + 50 / (1 + 1.5 * np.exp(-k3 * x / 10 - k4 * y / 1000))
```

```
    xiaolv /= 82.98

    return zuli,xiaolv

X=np.zeros([20,2])
X[0,0]=18
X[0,1]=890

for i in range(19):
    data=pd.read_csv("中间结果"+str(i)+".csv",header=None)
    X[i+1,:]=data.values.flatten()


Zuli=np.zeros([20,1])
Xiaolv=np.zeros([20,1])
for i in range(20):
    Zuli[i,:],Xiaolv[i,:]=obj(X[i,:])


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

idx=np.arange(20)

plt.plot(idx,Zuli*36.47,marker='.')

plt.xlabel('迭代次数(次)',fontsize=13)
plt.ylabel('过滤阻力Pa',fontsize=13)
plt.show()

plt.plot(idx,Xiaolv*82.98,marker='.')
plt.xlabel('迭代次数(次)',fontsize=13)
plt.ylabel('过滤效率 (%) ',fontsize=13)
plt.show()
```

可行域图像绘制

```
import matplotlib.pyplot as plt
import numpy as np

# 作点
x = np.linspace(-70, 120, 1000)
y = np.linspace(-300, 2500, 1000)

# 构造网格
x, y = np.meshgrid(x, y)
z = x
# 上式等号右边即为要绘制的方程，方程的右边等于0
# 绘制等高线,8表示等高线数量加1
plt.contour(x, y, z, 0)

z = x - 100
plt.contour(x, y, z, 0)

z = y - 2000
plt.contour(x, y, z, 0)

z = y
plt.contour(x, y, z, 0)

z = -0.8648 + 0.05428 * x + 0.001844 * y
plt.contour(x, y, z, 0)

z = -0.8648 + 0.05428 * x + 0.001844 * y - 3
```

```
plt.contour(x, y, z, 0)

z = 43.96 + 1.149 * x + 0.05897 * y - 0.02063 * x ** 2 + 2e-5 * x * y - 85 - 3.114e-5 * y ** 2
plt.contour(x, y, z, 0)
plt.rcParams['font.sans-serif'] = ['FangSong']
plt.rcParams['axes.unicode_minus'] = False
plt.xlabel('接收距离(cm)',fontsize=13)
plt.ylabel('热风速度(r/min)',fontsize=13)
# z = 43.96 + 1.149 * x + 0.05897 * y - 0.02063 * x ** 2 + 2e-5 * x * y -100- 3.114e-5 * y ** 2
# plt.contour(x, y, z, 0)

plt.show()
```

灵敏性检验计算

```
import matplotlib.pyplot as plt
import numpy as np
from scipy.optimize import minimize


#### 压缩回弹性
def yasuohuitan(X):
    popt = [4.39577299e+01, 1.15386219e+00, 5.88017537e-02, - 2.07223498e-02, 2.06636726e-05, -
        3.10795521e-05]
    p00 = popt[0]
    p10 = popt[1]
    p01 = popt[2]
    p20 = popt[3]
    p11 = popt[4]
    p02 = popt[5]
    x = X[0]
    y = X[1]
    return p00 + p10 * x + p01 * y + p20 * x ** 2 + p11 * x * y + p02 * y ** 2


#### 厚度
def houdu(X):
    popt = [0.05419093, 0.00184155, -0.86020508]
    return popt[0] * X[0] + popt[1] * X[1] + popt[2]


def obj(X):
    x = X[0]
    y = X[1]
    zuli = 4.05950528e+01 + 8.46631161e-01 * x - \
          1.93679792e-02 * y - 2.29699455e-02 * x ** 2 \
          + 1.36511669e-04 * x * y
    zuli /= 36.47

    k1 = 15.9
    k2 = 3.27944218
    k3 = -5.60411769
    k4 = 10.76586634
    xiaolv = 25 / (1 + 1.5 * np.exp(-k1 * x / 10)) \
            + 25 / (1 + 1.5 * np.exp(-k2 * y / 1000)) \
            + 50 / (1 + 1.5 * np.exp(-k3 * x / 10 - k4 * y / 1000))

    xiaolv /= 82.98

    return xiaolv * 82.98, zuli * 36.47


x = [ 17.97813515 ,1149.62005709]
print("压缩回弹，厚度，过滤效率，过滤阻力")
```

```
print(yasuohuitan(x))
print(houdu(x))
print(obj(x))
```