所属类别	2023 年“华数杯”全国大学生数学建模竞赛	参赛编号
本科组		CM2304214

基于优化模型和模拟退火算法的不透明制品最优配色方案设计

摘要

本文针对不透明制品配色问题的**优化设计和实际生产应用**进行了研究。首先建立了三种着色剂在不同波长下 **K/S 与浓度的拟合数学模型**，并求解了其函数关系式和拟合系数。然后结合 K-M 光学模型和 CIELAB 色彩空间的总色差计算方法，在给定目标样 R 值的前提下，建立了不透明制品配色的**优化模型**，并利用**模拟退火算法**求解。在其基础上，考虑成本控制和批量配色，利用**主要目标法**对多个决策目标中的色差进行约束，并在该约束条件下求解以价格为主要目标的优化模型。最后在实际生产情况中，引入生产方和购买方，考虑到能耗影响和市场变化趋势等因素，采用**逐次线性加权和法**求解多目标优化模型，计算出五个样本的五个最优的不同配方。

针对问题一，建立了三种着色剂在一定波长范围内浓度与对应 K/S 值的多**项式拟合数学模型**。首先，选取三种着色剂在一定波长范围内浓度与对应 K/S 值的数据并绘制出散点图，根据其趋势选取三种可行的数学拟合模型，分别求解其拟合系数，根据其拟合系数选取拟合效果最好的多项式拟合数学模型，并进一步求解所有数据的函数关系式和拟合系数，将求解结果记录在表格中。

针对问题二，建立了不透明制品配色**色差最小的单目标多决策变量优化模型**，并用**模拟退火算法**求解模型得到了最优决策变量。首先将三种着色剂的浓度作为决策变量，将三种着色剂的浓度在 0%到 5%之间且三种着色剂的浓度最多只有一种浓度为 0%作为约束条件，配方与目标样的总色差最小为目标函数建立优化模型。再利用模拟退火算法，调整退火温度衰减率和马尔可夫链的长度，基于 **Metropolis 准则**求解出与目标样的色差最为接近的 10 个不同配方并记录在表格中。

针对问题三，建立了不透明制品配色**成本最低的单目标多决策变量优化模型**，用**主要目标法**将目标函数中的色差转换为约束条件，并借助模拟退火算法进行求解。首先将第二问最优解的色差在一定范围内的溢出作为约束条件，三种着色剂的浓度作为决策变量，不透明制品配色的成本最低作为目标函数建立优化模型。利用主要目标法将第二问中最优解对应的色差作为基础，在允许 0.1 色差溢出的约束条件下使成本达到最低，利用模拟退火算法求解出 10 个最优的不同配方并记录在表格中。

针对问题四，建立了不透明制品配色色差、成本和总重三个目标函数的**多目标多决策变量优化模型**，使用**逐次线性加权和法**处理三个目标函数并借助模拟退火算法进行求解。首先将色差函数和成本函数进行线性加权，通过对加权值的线性分割，利用模拟退火算法，求得离散加权值对应的所有最优解，从中选取总重最小的五组最优解。调整加权值线性分割的精度，选取五组最优解分布最为广泛的精度作为最优分割精度，对应的五组解作为最终的最优解，同样计算剩余 4 个样本得到 5 个样本对应的 25 个最优配方并记录在表格中。

本文模型的优点为：1.解构了问题中几个参数之间复杂的数学关系并用函数的形式来表示，提高了**计算精度和速度**。2.进行了合理的假设，在优化模型

中抓住了主要的目标函数，采用多种方式处理**多目标函数**，让模型和算法准确高效。3.充分考虑了**实际情况**，提高了模型的实用性，可以应对多种市场要求和生产情况。4.采用**模拟退火算法**等启发式算法，有利于**跳出局部最优解**，得到的优化结果具有一定的参考意义。

关键词：不透明制品配色 优化模型 多项式拟合 模拟退火算法 主要目标法 逐次线性加权法

一、问题重述

1.1 问题背景

不透明有色制品在日常生活中使用十分广泛，其缤纷的色彩是由着色剂配色染成。配色对不透明有色制品的外观美观度和市场竞争力有重要影响。通过计算机方法来实现不透明制品的配色，在一定程度上能够克服传统人工配色存在的如主观性强、效率低下等局限性，节省大量人力、物力和财力，对减少能耗具有重要意义。

不透明制品配色问题，就是基于光学模型，设计不透明制品的配色模型。K-M 光学模型解释了不透明材料吸收系数 K/散射系数 S 的比值与反射率 R 之间存在的关系。基于光学模型得到的颜色参数，运用 CIELAB 色彩空间的总色差计算方法，可计算色差，以作为量化判断配色效果好坏的标准。另外，实际生产中还要考虑到成本控制和批量配色等问题。因此，我们需要思考如何平衡这三个因素之间的关系，在有限的条件下不断改进，以设计出色差更小，成本更低，操作性更强的配色方案。

1.2 问题提出

在上述背景下，本次数学建模竞赛设计了"不透明制品最优配色方案设计"一题并提供了相关参考文献资料和参数计算方法。本文在此基础上建立数学模型解决以下问题：

（1）问题一：根据所给参考资料及附件 2 中数据，分别计算红、黄、蓝三种着色剂在不同波长下 K/S 与浓度的关系，并在表格中填写函数关系式与对应的拟合系数。

（2）问题二：根据附件 3 提供的已知目标样的 R 值，基于附件 1 中的光谱三刺激值加权表与附件 2 中的着色剂 K/S 基础数据库，建立不透明制品配色的优化模型，并运用该模型配出与目标样的色差小于 1 且最为接近的 10 个不同配方。

（3）问题三：在已建立的配色模型基础上，考虑成本控制和批量配色，加以改进。根据附件 4 的色母粒单位克重价格，对 2kg 的基底材料进行配色，求解出与附件 3 中目标样之间色差小于 1 且最为接近的 10 个不同配方。

（4）问题四：在问题三的基础上，加上使实际生产中配色所需要的着色剂尽可能少的目标，对附件 3 中前 5 个样本进行最优配色方案设计，每个样本给出 5 个色差小于 1 的不同配方。

二、模型假设与分析

本文模型的建立基于以下假设条件：

（**1**）假设 K-M 光学模型中只考虑 K/S 值与 R 值之间的关系，以便于排除其他因素对计算和拟合的影响。

（**2**）假设三种着色剂的浓度范围均在附件给出数据中的 0%~5%之间，用于对浓度作为决策变量时进行约束。

（**3**）假设价格的浮动是在色差达到最优解的基础上进行的，用于处理多目标函数。

（**4**）假设着色剂的克重与能耗成正相关，市场对不透明制品的评判依据于色差和成本。

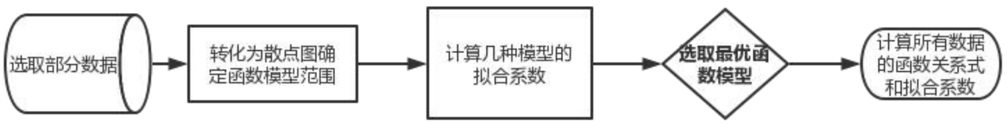
三、符号说明

四、模型的建立与求解

4.1 问题一的模型建立与求解

4.1.1 问题分析

本问给出了三种着色剂在不同波长下 K/S 与浓度的关系，要求根据数据拟合不同波长下两者的关系，并确定函数关系式和拟合系数。首先，选取部分数据转化为散点图，根据其走势选取几组较为贴近的函数模型，利用 python 程序求解其拟合系数，通过比较几种函数模型的拟合系数，选取拟合效果最好的函数模型。将附件二给出的参数带入该函数模型并进行拟合，可以求出函数关系式和拟合系数。

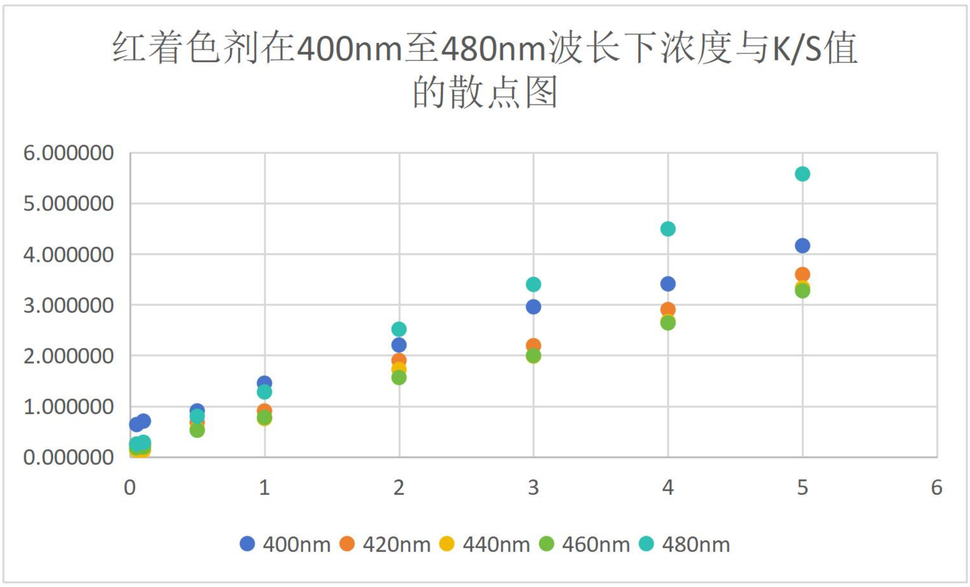

选取部分数据
转化为散点图确定函数模型范围
计算几种模型的拟合系数
选取最优函数模型
计算所有数据的函数关系式和拟合系数

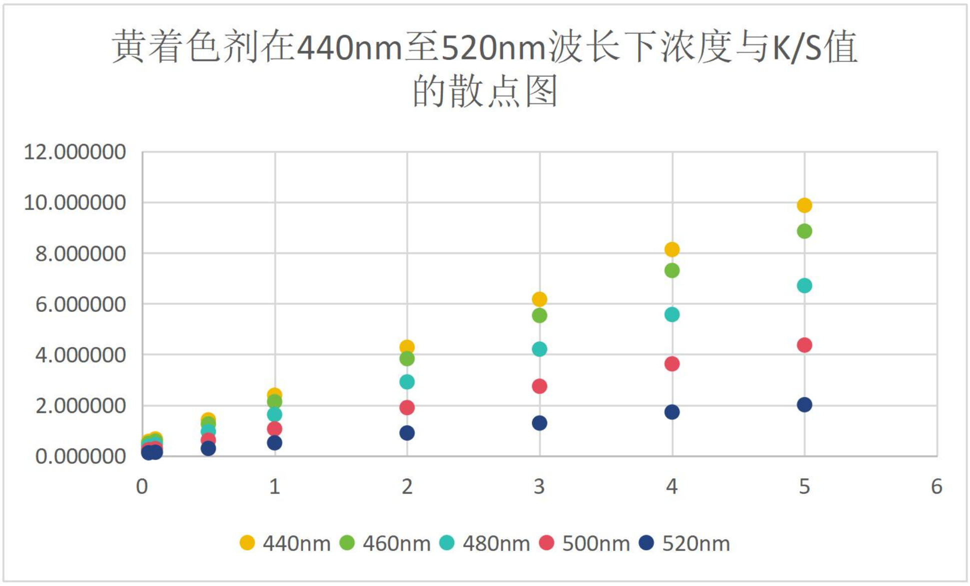
问题一流程图

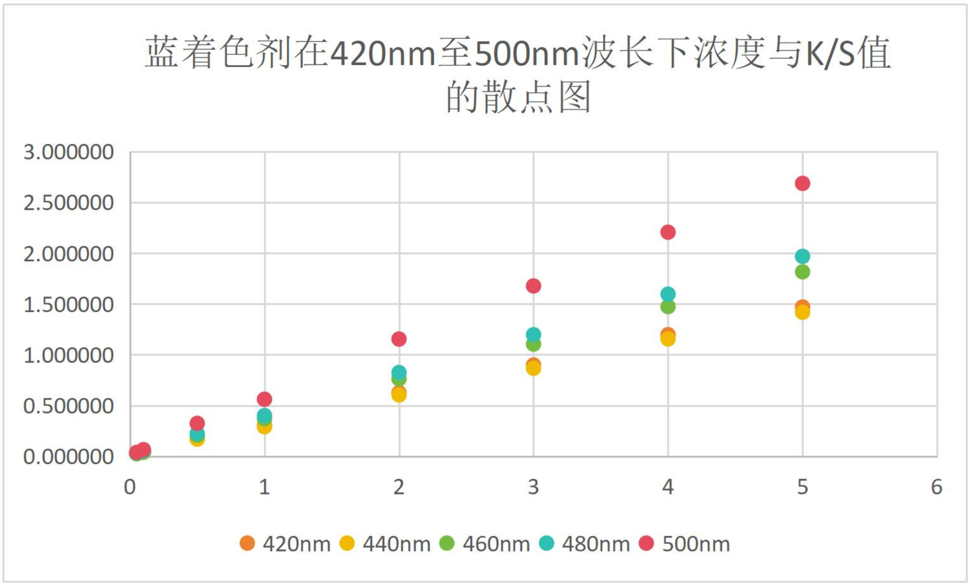
4.1.2 模型建立

函数模型范围的确定

选取三种着色剂在一定波长范围内浓度与对应 K/S 值的数据，绘制成散点图，如图所示。


红着色剂在400nm至480nm波长下浓度与K/S值的散点图
6.000000
5.000000
4.000000
3.000000
2.000000
1.000000
0.000000
0
1
2
3
4
5
6
● 400nm ● 420nm ● 440nm ● 460nm ● 480nm


黄着色剂在440nm至520nm波长下浓度与K/S值的散点图
12.000000
10.000000
8.000000
6.000000
4.000000
2.000000
0.000000
0
1
2
3
4
5
6
● 440nm ● 460nm ● 480nm ● 500nm ● 520nm


蓝着色剂在420nm至500nm波长下浓度与K/S值的散点图
3.000000
2.500000
2.000000
1.500000
1.000000
0.500000
0.000000
0
1
2
3
4
5
6
● 420nm ● 440nm ● 460nm ● 480nm ● 500nm

根据其走势及点数可得出三种可行的拟合数学模型，分别是指数、多项式（五次）、线性。编写 python 程序，对三组数据用以上三种数学模型进行拟合，并分别得出其拟合系数，如下表。

多项式拟合效果					
红色		黄色		蓝色	
波长	拟合系数	波长	拟合系数	波长	拟合系数
400nm	0.999076132	440nm	0.999999713	420nm	0.999763562
420nm	0.995717453	460nm	0.999999645	440nm	0.999746542
440nm	0.995330826	480nm	0.999999384	460nm	0.999846013
460nm	0.998755091	500nm	0.999996570	480nm	0.999868641
480nm	0.999581224	520nm	0.999984150	500nm	0.999814362

线性拟合效果					
红色		黄色		蓝色	
波长	拟合系数	波长	拟合系数	波长	拟合系数
400nm	0.993971696	440nm	0.999862864	420nm	0.999538700
420nm	0.989907616	460nm	0.999830523	440nm	0.999505492
440nm	0.988996478	480nm	0.999705822	460nm	0.999699565
460nm	0.997066209	500nm	0.999671167	480nm	0.999743713
480nm	0.999013098	520nm	0.998480473	500nm	0.999146403

指数拟合效果					
红色		黄色		蓝色	
波长	拟合系数	波长	拟合系数	波长	拟合系数
400nm	0.934644924	440nm	0.934657092	420nm	0.921696518
420nm	0.920713026	460nm	0.933791539	440nm	0.921190421
440nm	0.911892027	480nm	0.932692830	460nm	0.921835270
460nm	0.928669274	500nm	0.931823636	480nm	0.922394841
480nm	0.930468760	520nm	0.925882375	500nm	0.916104179

考虑到三种数学模型的拟合效果都较为优秀，这里给出保留九位小数的拟合数据，来更为精确地判断拟合效果的优劣。经过对三种数学模型的拟合系数的比较，最终选取精度最高的多项式进行拟合。将其余数据均带入 python 程序进行数据分析并将结果导出。三种着色剂在不同波长下 K/S 与浓度的函数关系式与拟合系数，如下表。其中浓度为百分比数值

波长	红色	
	函数关系式	拟合系数
400nm	$Y = 0.002051X^5 - 0.010625X^4 - 0.03136X^3 + 0.181794X^2 + 0.631014X + 0.612139$	0.999076132
420nm	$Y =- 0.012674X^5 + 0.161986X^4 - 0.705964X^3 + 1.160046X^2 + 0.191246X + 0.249396$	0.995717453
440nm	$Y =- 0.012674X^5 + 0.161986X^4 - 0.705964X^3 + 1.160046X^2 + 0.162746X + 0.130041$	0.995330826
460nm	$Y =- 0.006337X^5 + 0.080993X^4 - 0.352982X^3 + 0.580023X^2 + 0.382851X + 0.16443$	0.998755091

480nm	$Y =- 0.006337X^5 + 0.080993X^4 - 0.352982X^3 + 0.580023X^2 + 0.833352X + 0.214509$	0.999581224
500nm	$Y =- 0.006337X^5 + 0.080993X^4 - 0.352982X^3 + 0.580023X^2 + 1.429895X + 0.277175$	0.999827136
520nm	$Y =- 0.006337X^5 + 0.080993X^4 - 0.352982X^3 + 0.580023X^2 + 2.229895X + 0.343731$	0.999921032
540nm	$Y =- 0.006337X^5 + 0.080993X^4 - 0.352982X^3 + 0.580023X^2 + 2.429895X + 0.319349$	0.999932427
560nm	$Y =- 0.006337X^5 + 0.080993X^4 - 0.352982X^3 + 0.580023X^2 + 2.329895X + 0.363999$	0.999927062
580nm	$Y =- 0.005484X^5 + 0.069501X^4 - 0.300351X^3 + 0.488842X^2 + 0.878256X + 0.031301$	0.999631976
600nm	$Y =- 0.00561X^5 + 0.068259X^4 - 0.282375X^3 + 0.447774X^2 + 0.027184X + 0.017194$	0.999044898
620nm	$Y =- 0.000132X^5 + 0.001688X^4 - 0.007425X^3 + 0.012295X^2 + 0.045176X + 0.008161$	0.999778342
640nm	$Y = 2.0e - 5X^5 - 0.000213X^4 + 0.000686X^3 - 0.00058X^2 + 0.009633X + 0.006701$	0.999028053
660nm	$Y =- 2.4e - 5X^5 + 0.000292X^4 - 0.001249X^3 + 0.002107X^2 + 0.003789X + 0.00523$	0.99498286
680nm	$Y =- 1.1e - 5X^5 + 0.000147X^4 - 0.00071X^3 + 0.001342X^2 + 0.001148X + 0.006499$	0.998261544
700nm	$Y =- 1e - 5X^5 + 0.000147X^4 - 0.000710X^3 + 0.001341X^2 + 0.000148X + 0.006200$	0.990015435

黄色	
函数关系式	**拟合系数**
$Y =- 0.003817X^5 + 0.041305X^4 - 0.151455X^3 + 0.217372X^2 + 1.49948X + 0.483956$	0.999999594
$Y =- 0.003817X^5 + 0.041305X^4 - 0.151455X^3 + 0.217372X^2 + 1.59948X + 0.494269$	0.999999641
$Y =- 0.003817X^5 + 0.041305X^4 - 0.151455X^3 + 0.217372X^2 + 1.79948X + 0.484631$	0.999999713
$Y =- 0.003817X^5 + 0.041305X^4 - 0.151455X^3 + 0.217372X^2 + 1.60948X + 0.4185$	0.999999645
$Y =- 0.003817X^5 + 0.041305X^4 - 0.151455X^3 + 0.217372X^2 + 1.19948X + 0.322496$	0.999999384
$Y =- 0.002641X^5 + 0.028986X^4 - 0.108313X^3 + 0.158117X^2 + 0.776735X + 0.207159$	0.99999657
$Y =- 0.002641X^5 + 0.028986X^4 - 0.108313X^3 + 0.158117X^2 + 0.326735X + 0.107696$	0.99998415
$Y =- 0.000272X^5 + 0.002981X^4 - 0.01109X^3 + 0.016092X^2 + 0.142663X + 0.047312$	0.999991237
$Y =- 0.000231X^5 + 0.002448X^4 - 0.008708X^3 + 0.011759X^2 + 0.05533X + 0.023342$	0.99995724
$Y =- 0.000864X^5 + 0.009848X^4 - 0.038713X^3 + 0.062082X^2 - 0.011645X + 0.013055$	0.997255859
Y = -1.8e-5X^5+0.000177X^4-0.000594X^3+0.000748X^2+0.006688X+0.00463	0.99669633
y = -1E-05x5 + 0.0001x4 - 0.0004x3 + 0.0005x2 + 0.002x + 0.0024	0.999993488
Y = -3.0e-6X^5+3.3e-5X^4-0.000114X^3+0.000139X^2+0.000959X+0.002011	0.995144929
y = -3E-06x5 + 3E-05x4 - 0.0001x3 + 0.0001x2 + 0.0006x + 0.0018	0.999955768
y = -3E-06x5 + 3E-05x4 - 0.0001x3 + 0.0001x2 + 0.0004x + 0.0017	0.999899550
y = -3E-06x5 + 3E-05x4 - 9E-05x3 + 9E-05x2 + 0.0003x + 0.0014	0.999903013

蓝色	
函数关系式	**拟合系数**
Y = 0.009116X^5-0.107796X^4+0.428374X^3-0.661366X^2+0.739095X-0.007768	0.999449649
Y = -0.00052X^5+0.006226X^4-0.025767X^3+0.039744X^2+0.283189X+0.014166	0.999763562
Y = -0.00052X^5+0.006226X^4-0.025767X^3+0.039744X^2+0.273189X+0.012276	0.999746542
Y = -0.00052X^5+0.006226X^4-0.025767X^3+0.039744X^2+0.353189X+0.010555	0.999846013
Y = -0.00052X^5+0.006226X^4-0.025767X^3+0.039744X^2+0.383189X+0.012625	0.999868641
Y = 0.000424X^5-0.005624X^4+0.025163X^3-0.052617X^2+0.607543X+0.00962	0.999814362
Y = 0.000424X^5-0.005624X^4+0.025163X^3-0.052617X^2+0.747543X+0.014251	0.999882996
Y = 0.001218X^5-0.010554X^4+0.01593X^3+0.045462X^2+0.805218X+0.036563	0.999521571
Y = 0.001218X^5-0.010554X^4+0.01593X^3+0.045462X^2+1.205218X+0.038923	0.999773703
Y = -0.001731X^5+0.031472X^4-0.191258X^3+0.442689X^2+1.381631X+0.061326	0.999679078

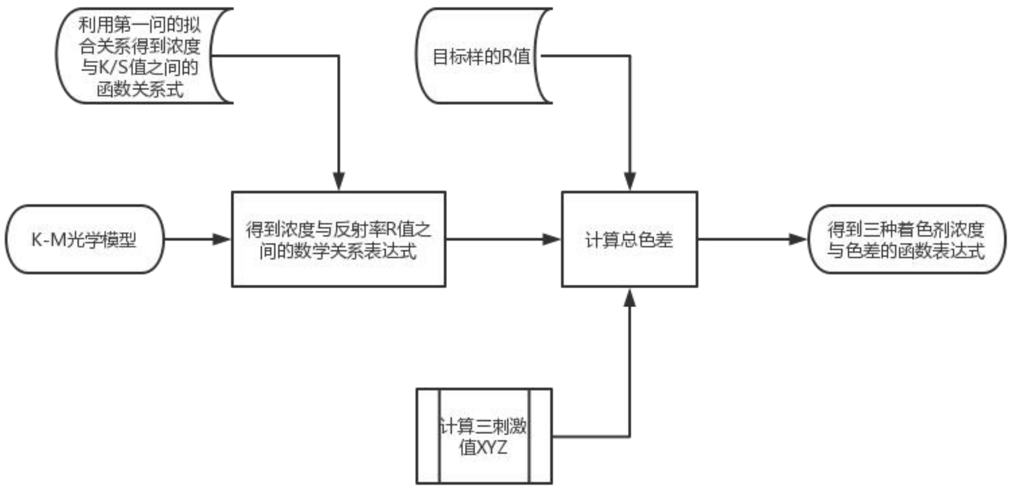
Y = -0.001731X^5+0.031472X^4-0.191258X^3+0.442689X^2+1.581631X+0.068624	0.999743847
Y = -1.2e-5X^5+0.013236X^4-0.126905X^3+0.353883X^2+1.722591X+0.068569	0.999781595
Y = -0.001731X^5+0.031472X^4-0.191258X^3+0.442689X^2+1.601631X+0.064267	0.999749221
Y = 0.001218X^5-0.010554X^4+0.01593X^3+0.045462X^2+1.205218X+0.032173	0.999773703
Y = -0.00052X^5+0.006226X^4-0.025767X^3+0.039744X^2+0.483189X+0.012261	0.999916552
Y = -0.00052X^5+0.006226X^4-0.025767X^3+0.039744X^2+0.333189X-0.000798	0.999827516

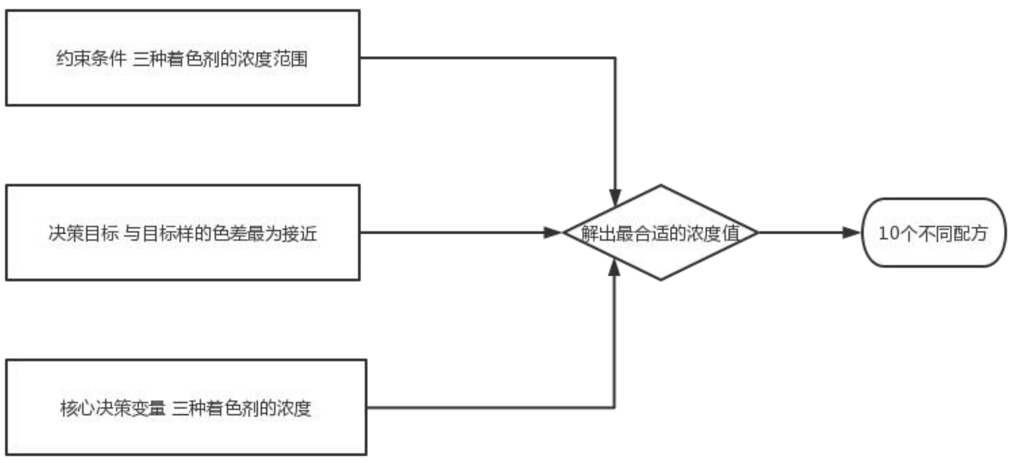
为形式统一，表格中多项式系数保留六位，对拟合系数有千分位上的影响。

4.2 问题二的模型建立与求解

4.2.1 问题分析

本题给出了目标样的 R 值、光谱三刺激值加权表和着色剂 K/S 的基础数据，要求建立不透明制品配色的优化模型。首先由 K-M 光学模型得到不透明制品中吸收系数 K 与散射系数 S 的比值和反射率 R 之间的数学关系表达式，从而得到不同浓度配方对应的反射率 R 值，结合目标样的 R 值，根据 CIELAB 色彩空间的总色差计算方法，计算出总色差ΔE，色差要求小于 1，并且色差越小说明配方越接近目标样。因此建立以三种颜色的浓度为决策变量，以总色差最小为目标函数优化目标的优化模型，用模拟退火算法求解模型得到 10 个不同配方的最优解。


利用第一问的拟合关系得到浓度与K/S值之间的函数关系式
目标样的R值
K-M光学模型
得到浓度与反射率R值之间的数学关系表达式
计算总色差
得到三种着色剂浓度与色差的函数表达式
计算三刺激值XYZ


约束条件 三种着色剂的浓度范围
决策目标 与目标样的色差最为接近
核心决策变量 三种着色剂的浓度
解出最合适的浓度值
10个不同配方

问题二流程图

4.2.2 模型建立

由 K-M 光学模型得到在 λ 波长处，基材的 K/S 值、着色剂的 K/S 值与反射率 R 之间的关系，其中每个着色剂的 K/S 值具有加和性，其数学关系表达式如下：

$$\left(\frac{k}{s}\right)_{\lambda}^{0}+\sum_{i=1}c_i\left(\frac{k}{s}\right)_{\lambda}^{i}=\frac{(1-R)^2}{2R}$$

其中每种着色剂的浓度所对应的 K/S 值使用第一问的拟合函数关系式进行计算，此处用 python 构造函数进行计算：

defKS(c，RYB，n)

由三种着色剂浓度计算对应 R 值，构造函数解一元二次方程：

defcalcurR(c1, c2, c3)

在 CIELAB 色彩空间中，总色差计算方法的颜色参数 L、红绿色度 a、黄蓝色度 b 中出现的三刺激值 XYZ 的计算方法如下：

$$X=k\int_{400}^{700}S(\lambda)\,\bar{x}(\lambda)R(\lambda)d(\lambda)$$
$$Y=k\int_{400}^{700}S(\lambda)\,\bar{y}(\lambda)R(\lambda)d(\lambda)$$
$$Z=k\int_{400}^{700}S(\lambda)\,\bar{z}(\lambda)R(\lambda)d(\lambda)$$

构造函数由模型样与目标样在可见光全光谱上 R 值计算所对应的三刺激值 XYZ：

defcalcuXYZ(R)

由三刺激值 XYZ 计算得到颜色参数明度、红绿色度和黄蓝色度，其计算方法如下：

若$\frac{x}{x_0}$、$\frac{y}{y_0}$、$\frac{z}{z_0}$均大于 0.008856，则按以下公式计算:

$$L^* = 116\left(\frac{X}{Y_0}\right)^{\frac{1}{3}} - 16$$

$$a^* = 500\left[\left(\frac{X}{X_0}\right)^{\frac{1}{3}} - \left(\frac{Y}{Y_0}\right)^{\frac{1}{3}}\right]$$

$$b^* = 200\left[\left(\frac{Y}{Y_0}\right)^{\frac{1}{3}} - \left(\frac{Z}{Z_0}\right)^{\frac{1}{3}}\right]$$

否则，按以下公式计算：

$$L^* = 903.3\left(\frac{Y}{Y_0}\right)$$

$$a^* = 3893.5\left(\frac{X}{X_0} - \frac{Y}{Y_0}\right)$$

$$b^* = 1557.4\left(\frac{Y}{Y_0} - \frac{Z}{Z_0}\right)$$

由模型样和目标样分别求得对应的L^*、a^*、b^*，并做差得到ΔL^*、Δa^*、Δb^*，并进一步计算得到总色差，其计算方法如下：

$$\Delta E^* = \left(\Delta L^{*2} + \Delta a^{*2} + \Delta b^{*2}\right)^{\frac{1}{2}}$$

综合上述计算结果，定义函数 F，计算总色差，即为目标函数，如下：

$$\mathrm{F(c1, c2, c3, R_target)}$$

建立以下模型：

目标函数： $\mathrm{F(c1, c2, c3, R_target)}$

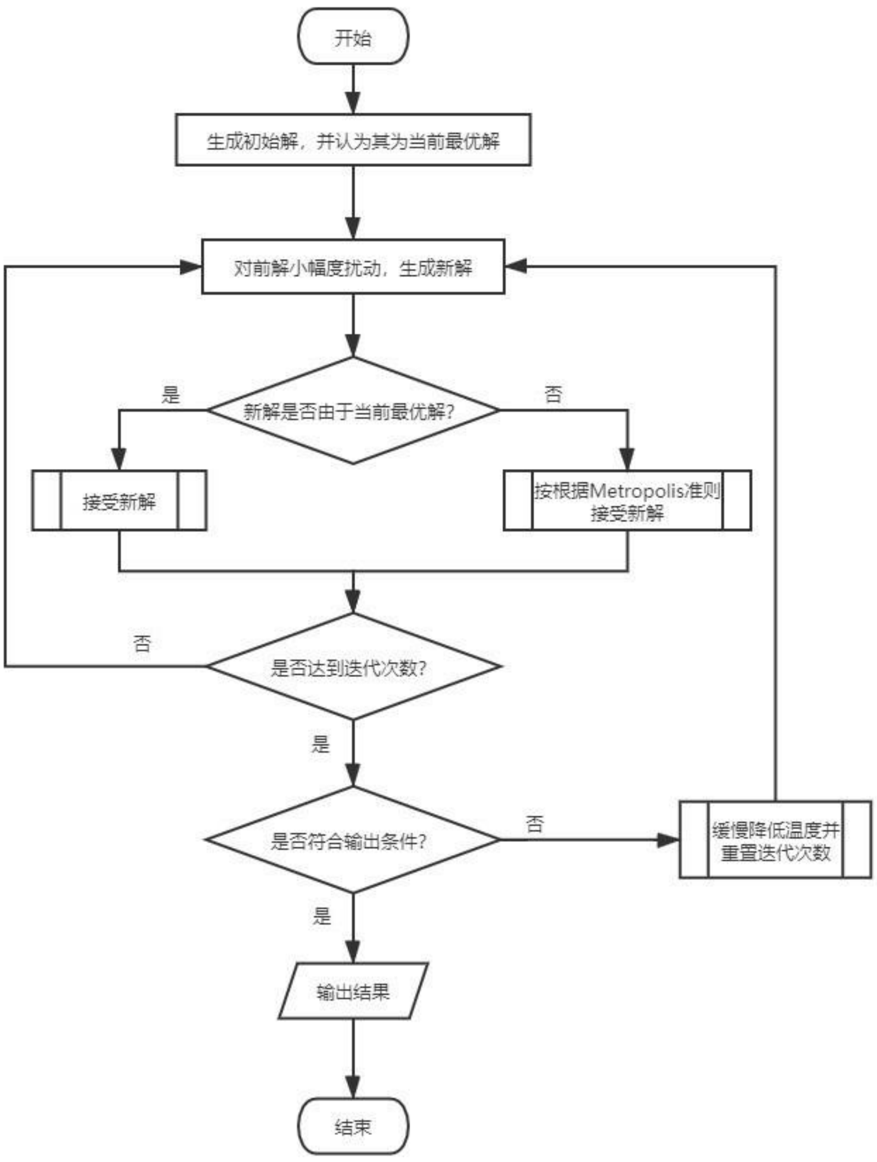
决策变量：三种着色剂的浓度c_1，c_2，c_3

约束条件：$0 \le c_1, c_2, c_3 \le 5(\%)$且三种着色剂的浓度最多只有一种浓度为 0

$$\begin{cases} Fmin \\ \mathrm{KS\left(c，RYB，n\right)} \\ \mathrm{calcurR(c1, c2, c3)} \\ \mathrm{calcuXYZ(R)} \\ \mathrm{F(c1, c2, c3, R_target)} \end{cases}$$

4.2.2 模型求解

模拟退火算法(Simulated Annealing，简称 SA)的思想最早是由 Metropolis 等提出的。其出发点是基于物理中固体物质的退火过程与一般的组合优化问题之间的相似性。模拟退火法是一种通用的优化算法,它通过模拟金属在高温下退火冷却的过程，逐步搜索问题的解空间，以找到全局最优解或者近似最优解，其流程图如下：


开始
生成初始解，并认为其为当前最优解
对前解小幅度扰动，生成新解
是
新解是否由于当前最优解?
否
接受新解
按根据Metropolis准则接受新解
否
是否达到迭代次数?
是
是否符合输出条件?
否
缓慢降低温度并重置迭代次数
是
输出结果
结束

4.2.2 结果展示

通过上述求解过程，得到 10 个配方的三种着色剂的浓度以及对应的色差，如下表：

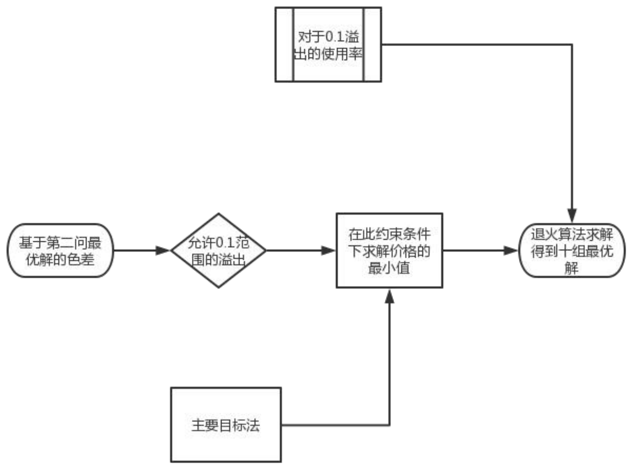
样本编号	着色剂浓度百分比			
	红色	黄色	蓝色	色差
1	0.405868753388968	0.0159161185404972	0.0443280051055219	0.0188764541197549

2	0.1079889286433858	3.028029619014244e-05	0.15581905846612099	0.683038928860927
3	0.19605127018610666	2.2357602251466338e-05	0.18098796521718252	0.716074725904492
4	0.00014309268924249743	0.03962821828028015	0.4302292924618222	0.551665824951313
5	6.87883483275753e-05	0.0220150074677026	0.3484044894071973	0.539150614371575
6	0.428444319092513	2.1995866696073113e-05	0.061805755191506134	0.721389835912762
7	0.3074447633566366	0.3573555296269151	0.07862051384656951	0.0327795861003594
8	0.10492920025242758	0.07981259554212641	0.14868613976092473	0.0260248943922577
9	0.031031329787946996	0.035943901695614344	0.3520823877367484	0.0186130549584457
10	0.06863792643266634	0.23496849152490215	0.4335639853741024	0.00996035647229648

4.3 问题三的模型建立与求解

4.3.1 问题分析

本题给出了基底材料的重量与色母粒单位千克重价格表，要求考虑成本控制与批量配色，改进配色模型。首先，利用浓度＝着色剂克重/基材重量公式，可实现浓度与着色剂克重之间的转换，每种着色剂的克重乘以对应色母粒单位克重并求和，即可得到每种配方的总价格。所以，将每种配方的总价格和总色差最小作为决策目标，利用**主要目标法**，也即只让一个最主要的决策目标作为优化的目标，将其他目标适当兼顾，把非主要决策目标降为约束条件。这里在第二问的基础上，以第二问所得最优解的色差为基准，允许 0.1 范围的溢出作为约束条件，求解价格最小值，得到目标函数。利用**模拟退火算法**求解最优的决策变量，并将着色剂克重转换为着色剂浓度，得到 10 个不同配方的最优解。



4.3.2 模型建立

引入色差变化比例来衡量对于 0. 1 色差溢出的使用率：

$$\left(delt_E - delt_{E_1}\right) * 10$$

引入价格差来衡量优化效果：

$$\Delta\,\mathrm{Calcuprice}(c1, c2, c3)$$

建立以下模型：

目标函数E_best
约束条件 0.1 的色差偏离容许度
决策变量 三种着色剂的浓度

$$\begin{cases} E\_best \\ \left(delt_E - delt_{E_1}\right) * 10 \\ \Delta\,\mathrm{Calcuprice}(c1, c2, c3) \end{cases}$$

运用模拟退火算法求解模型得到十个最优解与对应的色差变化比例与价格差，如下表

4.3.2 模型求解

此处数据见附件

4.4 问题四的模型建立与求解

4.4.1 问题分析

在实际生产过程中，对于不透明制品的生产方而言，其选取方案的主要依据是盈利和能耗问题，而对于不透明制品的购买方而言，考虑的主要是商品的质量和价格问题。本问题中，商品的成本决定了其盈利能力及价格，制品的着色剂用量决定了生产能耗，商品的色差决定了其质量。面对市场中消费者对价格与质量重视程度的改变趋势和日益严重的能源问题，对于每个样品给出的五个方案，应当满足以下条件：可以通过挑选并更改方案平衡制品成本及色差因素来应对市场趋势的改变，同时尽可能的减少配色所需的着色剂来减少能耗。

4.4.2 模型建立

引入函数来衡量配色所需的配色剂：

$$\mathrm{G}(c1, c2, c3)$$

引入函数来衡量配色所需的成本：

$$\mathrm{M}(c1, c2, c3)$$

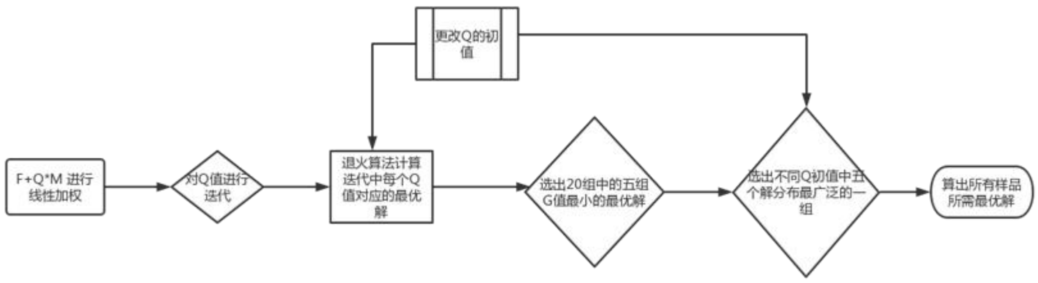
引入函数来衡量配色的色差：

$$\mathrm{F}(\mathrm{c1},\mathrm{c2},\mathrm{c3},\mathrm{R_target})$$

引入质量和价格的线性加权来衡量市场对价格与质量的重视程度：

$$\mathrm{F}+\mathrm{Q}*\mathrm{M}$$

其中 Q 为加权系数，在模拟退火算法中设定每次运行模拟退火算法程序进行 20 次迭代，每次迭代对 Q 进行 1.5 倍的指数迭代增长，以实现线性加权中加权数值的变化，来拟合市场的变化趋势。对 Q 的初值进行改变，分别为 0.1、0.2、0.3，来实现加权精度的改变。对于每组加权精度计算出来的所有最优解，选取 $\mathrm{G}(\mathrm{c1},\mathrm{c2},\mathrm{c3})$值最小的五组，对于三组加权精度，选取五个最优解分布最广泛的一组作为最终最优解。按照同样的方法计算出剩余四组样品的最优解。


更改Q的初值
F+Q*M 进行线性加权
对Q值进行迭代
退火算法计算迭代中每个Q值对应的最优解
选出20组中的五组G值最小的最优解
选出不同Q初值中五个解分布最广泛的一组
算出所有样品所需最优解

建立模型如下：

目标函数 $\mathrm{G}(\mathrm{c1},\mathrm{c2},\mathrm{c3})$　$\mathrm{M}(\mathrm{c1},\mathrm{c2},\mathrm{c3})$　$F(c1,c2,c3,R_{target})$

约束条件 $\mathrm{G}(\mathrm{c1},\mathrm{c2},\mathrm{c3})$最小的五组解分布的广泛程度
决策变量 三种着色剂的浓度

$$\begin{cases}\mathrm{G}(\mathrm{c1},\mathrm{c2},\mathrm{c3})\\ M(\mathrm{c1},\mathrm{c2},\mathrm{c3})\\ F(\mathrm{c1},\mathrm{c2},\mathrm{c3},R_{target})\\ \mathrm{F}+\mathrm{Q}*\mathrm{M}\end{cases}$$

模型求解与结果展示

k=1,q=0.1（依次为模型一 红 黄 蓝）

```
[0.3724062378546868,            5.849307217306589e-05,            2.7175657242476314e-05],
[0.2596603039628318,            0.00014168299990753412,           1.6016110581956126e-05],
[0.03876503108647762,           1.2656734527053881e-05,           1.9620683101406744e-05],
[0.0022778775991299247,         1.9473890329160543e-05,           4.195244906998138e-05],
[0.0010411482580754549, 2.7125180900193014e-05, 0.00022334579157878582]
```

```
k=1,q=0.2
[0.09343352527350265, 6.5256057462307455e-06, 3.327174357244609e-05],
[0.0020557189303957056, 0.00015900711066962832, 1.6961336287731314e-05],
[0.0005145919731025569, 4.963887740343058e-05, 0.0004757165251640528],
[0.0006368633833065583, 0.00030161827234401455, 4.004135870278513e-05],
[0.0005230918826340378, 0.00028222743161623843, 8.754193591393645e-05]

k=1,q=0.3
[0.0020708225876983136, 2.274534290666434e-05, 4.986593862809361e-05],
[0.00041049570859526263, 0.00031407667299366334, 1.5441473553348163e-05],
[0.00039037975758304355, 0.0002592929076892109, 0.00027131619620407904],
[0.00039901115549635365, 0.00022176194961736343, 4.2752240830936424e-05],
[0.0006458069763778929, 0.00011548418282324838, 6.528481964048433e-05]

k=2,q=0.1
[0.11031645007429526, 0.0001569040629891055, 5.912923910072077e-05],
[0.06684289348938109, 2.226538471465471e-06, 5.203974977663285e-05],
[0.009091474705007341, 5.1376052263592004e-05, 3.819699463268448e-05],
[0.001111231468860132, 6.233621886299594e-05, 0.00014310887919623217],
[0.0006319942214791856, 8.357584216964927e-06, 0.0004447126690938038]

k=2,q=0.2
[0.02677736155729507, 3.802128950826187e-05, 4.570084111976203e-05],
[0.002768560168313604, 5.987586490078354e-05, 0.00013932312627416369],
[0.0002895718259723083, 0.000292657659419083, 4.2706976792967035e-05],
[5.632482555299745e-06, 0.0005806340441861836, 0.00022437315131221085],
[0.00033756103128789903, 0.00018228761023645568, 0.00013119041304203698]
k=2,q=0.3
[0.0015175652009658059, 5.333973852774396e-05, 0.0003729225498502426],
[0.00028672825239414994, 0.00023364498418312028, 6.944866497908845e-05],
[0.00040126163417433087, 0.00024221657444883603, 4.0666848642075416e-05],
[0.0003211632641601012, 6.791491460310652e-05, 0.0002335786108377772],
[0.0004642554719830579, 1.5370086009737429e-06, 0.0003250291802130142]
k=3,q=0.1
[0.00014202545129853708, 3.586651603400075e-05, 0.2935024533599907],
[2.02401492450828e-05, 0.0003573872136316394, 0.0003613712045596749],
[0.0006690965384122347, 0.00014787174324057287, 9.91806728583072e-05],
[8.256213167051932e-05, 0.0005353575439121698, 0.00022014376366119495],
[2.0071411930498885e-05, 0.00028948015479531963, 0.0005719822106881838]
k=3,q=0.2
[0.00028182506651996237, 5.485405567258445e-05, 0.0004380214349352065],
[0.00018716859386790438, 3.9812047846576e-05, 0.0006941164778258667],
[7.359857453936446e-05, 0.0001618121270630715, 0.0007531218858001483],
[0.0004325514333620501, 0.0001880776770987218‍5, 0.0004462059509407618‍7],
[0.0001152735340663349‍3, 0.0003819581043253493‍6, 0.0003336885862902116‍5]
k=3,q=0.3
[0.0001356769327586753‍6, 0.0002277409280008221‍7, 0.0001923207857466033],
[0.0001444365911587386‍7, 3.70326798861361e-05,
0.0006032903721947076],[0.0003012833093170352‍6, 8.257843527582865e-05,
0.0005457007388058005], [0.0006640662849512529, 0.0001086883459540328‍5,
8.709239510859122e-05], [0.0003164029796393202, 8.042985304943418e-05,
0.0002191742032665201]
k=4,q=0.3
[8.702332766999735e-05, 0.0001361014848451200‍7, 0.2676300810106467],
[0.0005445535253711‍04, 5.74274177511323e-05, 0.0002497628126528722‍4],
[0.0004824217105187690‍6, 1.0726556337943777e-05, 0.0005233933907426332],
[0.0001706524041687482‍5, 0.0002012970209089580‍7, 0.0002515173766660527],
[0.0003546932261808957‍4, 0.0003888159487462289‍7, 0.0001816645380753414‍5]
k=4,q=0.3
[0.0004885760354880688, 0.00014504765290640‍7,
0.0003276424961393451],[0.0008326379600130001, 0.0001804960196771794,
5.992483494630658e-05], [0.0004119090144309180‍7, 0.00020301672920674,
5.3180139770037885e-05], [0.0003747065886171181, 0.0002259838749090487‍7,
0.0002555704282839546], [0.0005092776394508643, 2.45098738028632e-05,
0.0001821645317133074]
k=4,q=0.3
[0.0004742913207061882, 0.0001108568997210130‍8, 0.0001238573964583117],
[0.0001397018332220148‍5, 0.0004442296399584364, 9.01416408223047e-05],
[0.0004393422147663437‍7, 0.0002874108834784997‍6, 0.0001343250245311186],
[0.0001417197448506623, 0.0002109897976190225‍6, 0.000546428839498378]
[0.0001971781430171509‍7, 0.0001470837697533115‍5, 0.0002933373033653752]
```

五、模型的评价与扩展

5.1 模型优点

1.解构了问题中几个参数之间复杂的数学关系并用函数的形式来表示，提高了**计算精度和速度**。

2.进行了合理的假设，在优化模型中抓住了主要的目标函数，采用多种方式处理**多目标函数**，让模型和算法准确高效。

3.充分考虑了**实际情况**，提高了模型的实用性，可以应对多种市场要求和生产情况。

4.采用**模拟退火算法**等启发式算法，有利于**跳出局部最优解**，得到的优化结果具有一定的参考意义。

5.1 模型缺点及改进

模型中处理多目标函数问题时，进行决策时缺少必要依据，以及对于加权

系数的参数值确定，也缺乏必要依据。可以通过查找文献资料等确定实际情况中几种目标函数的权重划分，使优化结果更接近真实情况。

六、参考文献

[1] 王林吉，王兆青教授，《基于 CIELAB 均匀颜色空间和聚类算法的混纺测色研究》，2010-3-12

七、附录

附录 A 数据拟合-python 源程序

```python
import re
import pandas as pd
import numpy as np
import sympy as sp
import matplotlib.pyplot as plt
from scipy.stats import linregress
from scipy.optimize import curve_fit
excel_data = pd.read_excel('ks_value.xlsx')
color_start = 3
y_data_list = []
coefficients_list = []
expression_y_list = []
r_squared_list = []
x = sp.Symbol('X')
for color in range(0, 3):
    column = 400
    column_str = str(column)
    column_final = f'{column_str}nm'
    color_end = color_start + 7
    x_data = excel_data.loc[color_start:color_end, 'concentration'].astype(float)
    for num in range(2, 18):
        column_final = f'{column_str}nm'
        print(column_final)
        y_data = excel_data.loc[color_start:color_end, column_final].astype(float)
        y_data_list.append(y_data)
        coefficients = np.polyfit(x_data, y_data, deg=5)
        decimal_places = 6
        rounded_coefficients = np.round(coefficients, decimal_places)
        poly = np.poly1d(rounded_coefficients)
        coefficients_list.append(rounded_coefficients)
        column += 20
        column_str = str(column)
        print(rounded_coefficients)
        reversed_coefficients = rounded_coefficients[::-1]
        expression = sum(reversed_coefficients[i] * x ** i for i in range(5, -1, -1))
        expression_str = str(expression)
        expression_final = re.sub(r'\*\*', '^', expression_str)
        expression_final = re.sub(r'\*', '', expression_final)
        expression_final = re.sub(r' ', '', expression_final)
        expression_y = f'Y = {expression_final}'
        expression_y_list.append(expression_y)
        print(expression_y)
        x_fit = np.linspace(min(x_data), max(x_data), 100)
        y_fit = poly(x_fit)
        y_fit_resampled = np.interp(x_data, x_fit, y_fit)
        y_mean = np.mean(y_data)
        ss_total = np.sum((y_data - y_mean) ** 2)
        ss_residual = np.sum((y_data - y_fit_resampled) ** 2)
        r_squared = 1 - (ss_residual / ss_total)
        r_squared_list.append(r_squared)
        plt.scatter(x_data, y_data, label='Original Data')
        plt.plot(x_fit, y_fit, color='red', label='Fitted Polynomial')
        plt.xlabel('X')
        plt.ylabel(expression_y)
        plt.legend()
        plt.show()
    color_start = color_end + 1
data = {
    '函数关系式': expression_y_list,
    '拟合系数': r_squared_list
}
df = pd.DataFrame(data)
```

```
writer = pd.ExcelWriter('Q1_polynomial.xlsx', engine='xlsxwriter')
df.to_excel(writer, sheet_name='polynomial', startrow=2, startcol=1, index=False)
writer.save()
```

附录 B 问题二中出现的所有函数定义及模拟退火算法-python 源程序

```
def KS(c,RYB,n):#n 与波长线性相关，函数根据颜料浓度获得 K/S 值
    CC = [[[ 0.002051,-0.010625,-0.031360,0.181794,0.631014,0.612139],
[-0.012674,0.161986,-0.705964,1.160046,0.191246,0.249396],
[-0.012674,0.161986,-0.705964,1.160046,0.162746,0.130041],
[-0.006337,0.080993,-0.352982,0.580023,0.382851,0.164430],
[-0.006337,0.080993,-0.352982,0.580023,0.833352,0.214509],
[-0.006337,0.080993,-0.352982,0.580023,1.429895,0.277175],
[-0.006337,0.080993,-0.352982,0.580023,2.229895,0.343731],
[-0.006337,0.080993,-0.352982,0.580023,2.429895,0.319349],
[-0.006337,0.080993,-0.352982,0.580023,2.329895,0.363999],
[-0.005484,0.069501,-0.300351,0.488842,0.878256,0.031301],
[-0.005610,0.068259,-0.282375,0.447774,0.027184,0.017194],
[-0.000132,0.001688,-0.007425,0.012295,0.045176,0.008161],
[2.0e-5,-0.000213,0.000686,-0.00058,0.009633,0.006701],
[-2.4e-5,0.000292,-0.001249,0.002107,0.003789,0.00523],
[-1.1e-5,0.000147,-0.00071,0.001342,0.001148,0.006499],
[-1e-5,0.000147, - 0.000710,0.001341,0.000148,0.006200]],
[[-0.003817,0.041305,-0.151455,0.217372,1.49948,0.483956],
[-0.003817,0.041305,-0.151455,0.217372,1.59948,0.494269],
[-0.003817,0.041305,-0.151455,0.217372,1.79948,0.484631],
[-0.003817,0.041305,-0.151455,0.217372,1.60948,0.4185],
[-0.003817,0.041305,-0.151455,0.217372,1.19948,0.322496],
[-0.002641,0.028986,-0.108313,0.158117,0.776735,0.207159],
[-0.002641,0.028986,-0.108313,0.158117,0.326735,0.107696],
[-0.000272,0.002981,-0.01109,0.016092,0.142663,0.047312],
[-0.000231,0.002448,-0.008708,0.011759,0.05533,0.023342],
[-0.000864,0.009848,-0.038713,0.062082,-0.011645,0.013055],
[-1.8e-5,0.000177,-0.000594,0.000748,0.006688,0.00463],
[-1E-05,0.0001,- 0.0004,0.0005,0.002,0.0024],
[-3.0e-6,3.3e-5,-0.000114,0.000139,0.000959,0.002011],
[-3E-06,3E-05,- 0.0001,0.0001,0.0006,0.0018],
[-3E-06,3E-05,- 0.0001,0.0001,0.0004,0.0017],
[-3E-06,3E-05,- 9E-05,9E-05,0.0003,0.0014]],
[[0.009116,-0.107796,0.428374,-0.661366,0.739095,-0.007768],
[-0.00052,0.006226,-0.025767,0.039744,0.283189,0.014166],
[-0.00052,0.006226,-0.025767,0.039744,0.273189,0.012276],
[-0.00052,0.006226,-0.025767,0.039744,0.353189,0.010555],
[-0.00052,0.006226,-0.025767,0.039744,0.383189,0.012625],
[0.000424,-0.005624,0.025163,-0.052617,0.607543,0.00962],
[0.000424,-0.005624,0.025163,-0.052617,0.747543,0.014251],
[0.001218,-0.010554,0.01593,0.045462,0.805218,0.036563],
[0.001218,-0.010554,0.01593,0.045462,1.205218,0.038923],
[-0.001731,0.031472,-0.191258,0.442689,1.381631,0.061326],
[-0.001731,0.031472,-0.191258,0.442689,1.581631,0.068624],
[-1.2e-5,0.013236,-0.126905,0.353883,1.722591,0.068569],
[-0.001731,0.031472,-0.191258,0.442689,1.601631,0.064267],
[0.001218,-0.010554,0.01593,0.045462,1.205218,0.032173],
[-0.00052,0.006226,-0.025767,0.039744,0.483189,0.012261],
[-0.00052,0.006226,-0.025767,0.039744,0.333189,-0.000798]]]#K/S 在不同颜色，不同波长下关于颜料浓度的多项式系数
    return CC[RYB][n][0]*c**5 + CC[RYB][n][1]*c**4 + CC[RYB][n][2]*c**3 + CC[RYB][n][3]*c**2 + CC[RYB][n][4]*c + CC[RYB][n][5]

def calcuR(c1,c2,c3):#由三种染色剂浓度求 R 值
    KS0 = [0.039492,0.025906,0.017964,0.015092,0.011439,0.009515,0.007961,0.006947,0.006284,0.005889,0.005238,0.004948,0.004626,0.004247,0.004100,0.003617]
    KS1 = []
    KS2 = []
    KS3 = []
    for i in range(16):
        KS1.append(KS(c1,0,i))
    for i in range(16):
        KS2.append(KS(c2,1,i))
    for i in range(16):
        KS3.append(KS(c3,2,i))

    a = []
    for i in range(16):
        a.append(KS0[i] + c1*KS1[i] + c2*KS2[i] + c3*KS3[i])

    R = []
    for i in range(16):
        R.append(a[i]+1 - (a[i]**2 + 2*a[i])**0.5)
    return R#获得全波段

def calcuXYZ(R):#由 R 获得 XYZ 值
    lis_3 = [[0.136,0.014,0.613],
    [1.644,0.172,7.820],
    [3.463,0.560,17.755],
    [3.065,1.300,17.697],
    [0.803,2.530,7.703],
    [0.036,4.337,2.056],
    [1.062,6.870,0.548],
    [3.385,8.644,0.123],
    [6.069,8.583,0.000],
    [8.361,7.163,0.000],
    [8.707,5.100,0.000],
    [6.463,3.004,0.000],
    [3.109,1.295,0.000],
    [1.053,0.416,0.000],
    [0.275,0.107,0.000],
    [0.059,0.023,0.000]]
    X = 0
    Y = 0
    Z = 0
    for i in range(16):
        if i == 0 or i == 15:
            X += lis_3[i][0]*R[i]#R 指标是波长的话要改
        else:
            X += lis_3[i][0]*2*R[i]

    for i in range(16):
        if i == 0 or i == 15:
            Y += lis_3[i][1]*R[i]#R 指标是波长的话要改
        else:
            Y += lis_3[i][1]*2*R[i]

    for i in range(16):
        if i == 0 or i == 15:
            Z += lis_3[i][2]*R[i]#R 指标是波长的话要改
        else:
            Z += lis_3[i][2]*2*R[i]

  return [X,Y,Z]

def calcuLab(XYZ):#输入 XYZ 列表
    X0 = 94.83
    Y0 = 100
    Z0 = 107.38
    L = 116*(XYZ[1]/Y0)**(1/3) - 16
    a = 500*((XYZ[0]/X0)**(1/3) - (XYZ[1]/Y0)**(1/3))
    b = 200*((XYZ[1]/Y0)**(1/3) - (XYZ[2]/Z0)**(1/3))
    lis = [L,a,b]
    return lis#返回 L*,a*,b*列表
```

```

def calcu_delta_E(Lab1,Lab2):#由两组 L*,a*,b*列表获得色差
    delta_E = 0
    for i in range(3):
        delta_E += (Lab1[i]-Lab2[i])**2
    return delta_E**0.5

def F(c1,c2,c3,R_target):#目标函数
    return calcu_delta_E(calcuLab(calcuXYZ(calcuR(c1,c2,c3))),calcuLab(calcuXYZ(R_target)))

#以下为模拟退火算法求解函数优化问题
import random
import math
R_target = [0.681044,0.724153,0.749374,0.752377,0.733554,0.699726,0.663874,0.652066,\
            0.642280,0.689994,0.720488,0.725711,0.732905,0.771153,0.838096,0.859082]
sol_new1 = 2.5#初始值
sol_new2 = 2.5
sol_new3 = 2.5
sol_current1 = sol_new1#迭代过程中记录
sol_current2 = sol_new2
sol_current3 = sol_new3
sol_best1 = sol_new1
sol_best2 = sol_new2
sol_best3 = sol_new3
E_current = 100#目标函数优化初始化
E_best = 100
T = 0.5#初始温度
T1 = 0.1#结束温度
aa = 0.99#温度衰减率

while T >= T1:
    times = 0
    E_record = []
    while times < 1000:
        times += 1
        sol_new01 = sol_new1 + (random.random()-0.5)*0.05
        sol_new02 = sol_new2 + (random.random()-0.5)*0.05
        sol_new03 = sol_new3 + (random.random()-0.5)*0.05

        if 0 <= sol_new01 <= 5 and 0 <= sol_new02 <= 5 and 0 <= sol_new03 <= 5 and\
        sol_new01*sol_new02 + sol_new01*sol_new03 + sol_new02*sol_new03 > 1e-7:
            sol_new1 = sol_new01
            sol_new2 = sol_new02
            sol_new3 = sol_new03
        else:
            times -= 1
            continue
        E_new = F(sol_new1,sol_new2,sol_new3,R_target)
        if E_new < E_current:
            E_current = E_new
            sol_current1 = sol_new1
            sol_current2 = sol_new2
            sol_current3 = sol_new3
            if E_new <E_best:
                E_best = E_new
                sol_best1 = sol_new1
                sol_best2 = sol_new2
                sol_best3 = sol_new3
        else:
            if random.random()<math.e**(-(E_new-E_current)/T):
                E_current = E_new
                sol_current1 = sol_new1
                sol_current2 = sol_new2
                sol_current3 = sol_new3
            else:
                sol_new1 = sol_current1
                sol_new2 = sol_current2
                sol_new3 = sol_current3
        E_record.append(E_current)
    T = T*aa
print('E_record ',E_record)
print('sol_best: ',sol_best1,sol_best2,sol_best3)#最优解
print('E_best ',E_best)#色差
```

附录 C 第三问中出现的函数定义及调整的模拟退火算法-python 源程序

```
def calcuprice(c1,c2,c3):
    return 0.01*(c1*2*60 + c2*2*65 + c3*2*63)

def F(c1,c2,c3,R_target):
    return calcu_delta_E(calcuLab(calcuXYZ(calcuR(c1,c2,c3))),calcuLab(calcuXYZ(R_target)))

```

```python
#以下为模拟退火算法求解函数优化问题
import random
import math
k = 0
lis_sol_best = [[0.40586875338896833,0.015916118540497272,0.04432800510552193],
[0.1079889286433858,3.028029619014244e-05,0.15581905846612099],
[0.19605127018610666,2.2357602251466338e-05,0.18098796521718252],
[0.00014309268924249743,0.03962821828028015,0.4302292924618222],
[6.87883483275753e-05,0.0220150074677026,0.3484044894071973],
[0.428444319092513,2.1995866696073113e-05,0.061805755191506134],
[0.3074447633566366,0.3573555296269151,0.07862051384656951],
[0.10492920025242758,0.07981259554212641,0.14868613976092473],
[0.031031329787946996,0.035943901695614344,0.3520823877367484],
[0.06863792643266634,0.23496849152490215,0.4335639853741024]]
R = [[0.507045,0.532145,0.546225,0.557051,0.493464,0.430796,0.370752,0.359875,0.365956,0.513965,0.716593,0.837977,0.888602,0.900323,0.906528,0.913298],
[0.681044,0.724153,0.749374,0.752377,0.733554,0.699726,0.663874,0.652066,0.642280,0.689994,0.720488,0.725711,0.732905,0.771153,0.838096,0.859082],
[0.629420,0.668642,0.690550,0.693808,0.654061,0.603730,0.553318,0.539833,0.534764,0.616401,0.680087,0.695650,0.705559,0.747784,0.823197,0.846680],
[0.598105,0.636542,0.647724,0.634758,0.638956,0.610761,0.589352,0.556888,0.500971,0.457357,0.439055,0.430547,0.437718,0.503085,0.646008,0.692619],
[0.641290,0.680826,0.695806,0.685813,0.690465,0.665507,0.646113,0.616788,0.565565,0.524583,0.507192,0.499046,0.506062,0.568136,0.697296,0.738285],
[0.507637,0.534544,0.550889,0.560809,0.488216,0.420199,0.357506,0.345772,0.351452,0.499036,0.703689,0.825218,0.874593,0.889770,0.901908,0.910100],
[0.376815,0.379640,0.371046,0.387641,0.401820,0.412266,0.412390,0.425963,0.439564,0.576515,0.736379,0.810591,0.836221,0.858021,0.886589,0.898088],
[0.611623,0.636383,0.644250,0.654583,0.661286,0.657130,0.645990,0.645381,0.639611,0.688578,0.719943,0.725500,0.732772,0.771024,0.837919,0.858908],
[0.616192,0.654238,0.667361,0.660191,0.663057,0.638999,0.618215,0.593015,0.547792,0.519088,0.506045,0.498777,0.505973,0.568058,0.697160,0.738137],
[0.457151,0.471454,0.465966,0.472114,0.495952,0.506516,0.517486,0.510463,0.471248,0.447707,0.436959,0.430048,0.437543,0.502925,0.645742,0.692336]]
R_target = R[k]
sol_new1 = lis_sol_best[k][0]
sol_new2 = lis_sol_best[k][1]
sol_new3 = lis_sol_best[k][2]
sol_current1 = sol_new1#迭代过程中记录
sol_current2 = sol_new2
sol_current3 = sol_new3
sol_best1 = sol_new1
sol_best2 = sol_new2
sol_best3 = sol_new3
E_current = 999999#目标函数优化初始化
E_best = 999999
T = 0.5#初始温度
T1 = 0.1#结束温度
aa = 0.99#温度衰减率
lis_delta_E_best = [0.018876454119754943,0.683038928860927,0.7160747259044927,0.5516658249513138,0.5391506143715753,0.7213898359127621,0.032779586100359435,0.026024894392257712,0.018613054958445764,0.009960356472296485]

while T >= T1:
    print(T)
    times = 0
    E_record = []
    while times < 1000:
        times += 1
        sol_new01 = sol_new1 + (random.random()-0.5)*0.001
        sol_new02 = sol_new2 + (random.random()-0.5)*0.001
        sol_new03 = sol_new3 + (random.random()-0.5)*0.001
        #print(times,F(sol_new01,sol_new02,sol_new03,R_target) , lis_delta_E_best[k]+0.1)
        if 0 <= sol_new01 <= 5 and 0 <= sol_new02 <= 5 and 0 <= sol_new03 <= 5 and\
        sol_new01*sol_new02 + sol_new01*sol_new03 + sol_new02*sol_new03 > 1e-7 and\
        F(sol_new01,sol_new02,sol_new03,R_target) < lis_delta_E_best[k] + 0.1:#0.1 的色差偏离容许度
            sol_new1 = sol_new01
            sol_new2 = sol_new02
            sol_new3 = sol_new03
        else:
            times -= 1
            continue
        E_new = calcuprice(sol_new1,sol_new2,sol_new3)
        if E_new < E_current:
            E_current = E_new
            sol_current1 = sol_new1
            sol_current2 = sol_new2
            sol_current3 = sol_new3
            if E_new <E_best:
                E_best = E_new
                sol_best1 = sol_new1
                sol_best2 = sol_new2
                sol_best3 = sol_new3
        else:
            if random.random()<math.e**(-(E_new-E_current)/T):
                E_current = E_new
                sol_current1 = sol_new1
                sol_current2 = sol_new2
                sol_current3 = sol_new3
            else:
                sol_new1 = sol_current1
                sol_new2 = sol_current2
                sol_new3 = sol_current3
        E_record.append(E_current)
    T = T*aa
print('sol_best: ',sol_best1,sol_best2,sol_best3)#最优解
print('E_best ',E_best)#目标函数优化值

delta_E = calcu_delta_E(calcuLab(calcuXYZ(calcuR(sol_best1,sol_best2,sol_best3))),calcuLab(calcuXYZ(R_target)))
delta_E1= calcu_delta_E(calcuLab(calcuXYZ(calcuR(lis_sol_best[k][0], lis_sol_best[k][1], lis_sol_best[k][2]))),calcuLab(calcuXYZ(R_target)))
print('色差 ',delta_E)
print('色差变化比例',(delta_E - delta_E1)*10)
print('价格差 ',calcuprice( sol_best1, sol_best2, sol_best3) - calcuprice( lis_sol_best[k][0], lis_sol_best[k][1], lis_sol_best[k][2]))
```

附录 D 第四问中出现的函数定义以及更改后的模拟退火算法-python 源程序

```python
def F(c1,c2,c3,R_target,Q):#目标函数
    return calcu_delta_E(calcuLab(calcuXYZ(calcuR(c1,c2,c3))),calcuLab(calcuXYZ(R_target))) + calcuprice(c1,c2,c3)*Q
```

```

#以下为模拟退火算法求解函数优化问题
import random
import math
k = 0
lis_sol_best = [[0.40586875338896833,0.015916118540497272,0.04432800510552193],
[0.1079889286433858,3.028029619014244e-05,0.15581905846612099],
[0.19605127018610666,2.2357602251466338e-05,0.18098796521718252],
[0.00014309268924249743,0.03962821828028015,0.4302292924618222],
[6.87883483275753e-05,0.0220150074677026,0.3484044894071973],
[0.428444319092513,2.1995866696073113e-05,0.061805755191506134],
[0.3074447633566366,0.3573555296269151,0.07862051384656951],
[0.10492920025242758,0.07981259554212641,0.14868613976092473],
[0.031031329787946996,0.035943901695614344,0.3520823877367484],
[0.06863792643266634,0.23496849152490215,0.4335639853741024]]
R = [[0.507045,0.532145,0.546225,0.557051,0.493464,0.430796,0.370752,0.359875,0.365956,0.513965,0.716593,0.837977,0.888602,0.900323,0.906528,0.913298],
[0.681044,0.724153,0.749374,0.752377,0.733554,0.699726,0.663874,0.652066,0.642280,0.689994,0.720488,0.725711,0.732905,0.771153,0.838096,0.859082],
[0.629420,0.668642,0.690550,0.693808,0.654061,0.603730,0.553318,0.539833,0.534764,0.616401,0.680087,0.695650,0.705559,0.747784,0.823197,0.846680],
[0.598105,0.636542,0.647724,0.634758,0.638956,0.610761,0.589352,0.556888,0.500971,0.457357,0.439055,0.430547,0.437718,0.503085,0.646008,0.692619],
[0.641290,0.680826,0.695806,0.685813,0.690465,0.665507,0.646113,0.616788,0.565565,0.524583,0.507192,0.499046,0.506062,0.568136,0.697296,0.738285],
[0.507637,0.534544,0.550889,0.560809,0.488216,0.420199,0.357506,0.345772,0.351452,0.499036,0.703689,0.825218,0.874593,0.889770,0.901908,0.910100],
[0.376815,0.379640,0.371046,0.387641,0.401820,0.412266,0.412390,0.425963,0.439564,0.576515,0.736379,0.810591,0.836221,0.858021,0.886589,0.898088],
[0.611623,0.636383,0.644250,0.654583,0.661286,0.657130,0.645990,0.645381,0.639611,0.688578,0.719943,0.725500,0.732772,0.771024,0.837919,0.858908],
[0.616192,0.654238,0.667361,0.660191,0.663057,0.638999,0.618215,0.593015,0.547792,0.519088,0.506045,0.498777,0.505973,0.568058,0.697160,0.738137],
```

```python
[0.457151,0.471454,0.465966,0.472114,0.495952,0.506516,0.517486,0.510463,0.471248,0.447707,0.436959,0.430048,0.437543,0.502925,0.645742,0.692336]]
lis_delta_E_best = [0.018876454119754943,0.683038928860927,0.7160747259044927,0.5516658249513138,0.5391506143715753,0.7213898359127621,0.032779586100359435,0.026024894392257712,0.018613054958445764,0.009960356472296485]
R_target = R[k]

LIS_Q_sol_best = []
LIS_Q_mess = []
LIS_Q = []
LIS_deltaE = []
Q = 0.1
for n in range(10):
    print(n)
    sol_new1 = 1#初始值
    sol_new2 = 1
    sol_new3 = 1
    sol_current1 = sol_new1#迭代过程中记录
    sol_current2 = sol_new2
    sol_current3 = sol_new3
    sol_best1 = sol_new1
    sol_best2 = sol_new2
    sol_best3 = sol_new3
    E_current = 99999#目标函数优化初始化
    E_best = 99999
    T = 0.5#初始温度
    T1 = 0.02#结束温度
    aa = 0.985#温度衰减率
    while T >= T1:
        print(T)
        times = 0
        #E_record = []
        while times < 1000:
            times += 1
            sol_new01 = sol_new1 + (random.random()-0.5)*0.05
            sol_new02 = sol_new2 + (random.random()-0.5)*0.05
            sol_new03 = sol_new3 + (random.random()-0.5)*0.05

            if 0 <= sol_new01 <= 5 and 0 <= sol_new02 <= 5 and 0 <= sol_new03 <= 5 and\
            sol_new01*sol_new02 + sol_new01*sol_new03 + sol_new02*sol_new03 > 1e-7:
                sol_new1 = sol_new01
                sol_new2 = sol_new02
                sol_new3 = sol_new03
            else:
                times -= 1
                continue
            E_new = F(sol_new1,sol_new2,sol_new3,R_target,Q)
            if E_new < E_current:
                E_current = E_new
                sol_current1 = sol_new1
                sol_current2 = sol_new2
                sol_current3 = sol_new3
                if E_new <E_best:
                    E_best = E_new
                    sol_best1 = sol_new1
                    sol_best2 = sol_new2
                    sol_best3 = sol_new3
            else:
                if random.random()<math.e**(-(E_new-E_current)/T):
                    E_current = E_new
                    sol_current1 = sol_new1
                    sol_current2 = sol_new2
                    sol_current3 = sol_new3
                else:
                    sol_new1 = sol_current1
                    sol_new2 = sol_current2
                    sol_new3 = sol_current3
            #E_record.append(E_current)
        T = T*aa
    LIS_Q_sol_best.append([sol_best1,sol_best2,sol_best3])
    LIS_deltaE.append(calcu_delta_E(calcuLab(calcuXYZ(calcuR(sol_best1,sol_best2,sol_best3))),calcuLab(calcuXYZ(R_target))))
    LIS_Q_mess.append(2*(sol_best1 + sol_best2 + sol_best3))
    LIS_Q.append(Q)
    Q *= 1.5
'''
print('sol_best: ',sol_best1,sol_best2,sol_best3)#最优解
print('E_best ',E_best)#色差
print('价格比例 ',calcuprice( sol_best1, sol_best2, sol_best3) / calcuprice( lis_sol_best[k][0], lis_sol_best[k][1], lis_sol_best[k][2]))
delta_E = calcu_delta_E(calcuLab(calcuXYZ(calcuR(sol_best1,sol_best2,sol_best3))),calcuLab(calcuXYZ(R_target)))
```

```
delta_E1= calcu_delta_E(calcuLab(calcuXYZ(calcuR(lis_sol_best[k][0], lis_sol_best[k][1], lis_sol_best[k][2]))),calcuLab(calcuXYZ(R_target)))
print('色差 ',delta_E)
print('色差变化比例',delta_E/delta_E1)#理论上大于 1
'''
print('LIS_Q_sol_best\n',LIS_Q_sol_best)
print('LIS_Q_mess\n',LIS_Q_mess)
print('LIS_Q\n',LIS_Q)
print('LIS_deltaE',LIS_deltaE)
```