

所属类别	2025 年“华数杯”全国大学生数学建模竞赛	参赛编号
本科组		CM2502422

基于多通道 LED 光谱模拟的节律效应优化与应用研究

摘要

本文研究了 LED 光源的光谱特性及其对人体生理节律和睡眠质量的影响，并构建了相应的数学模型和统计分析方法，以指导可调控生物节律的 LED 光源设计及应用。

对于问题一，要求搭建光源的关键光谱参数计算模型。先对原始 SPD 数据进行处理，通过 CIE 1931 三刺激值函数将其转换至 CIE XYZ 色彩空间并求得色品坐标。据 McCamy 近似法获得 CCT，色偏 D_{uv} 由色品坐标与黑体轨迹距离计算， R_f 与 R_g 按 ANSI/IES TM-30-20 得出，测试光源与 D65 melanopsin 加权辐射比值即为 mel-DER。经计算，CCT 为 3903.2 K， D_{uv} 接近 -0.0010， R_f 达 91.79， R_g 为 106.08，而 mel-DER 仅为 0.6407，显示该光源的高显色性且节律影响低于自然光。

对于问题二，基于五个 LED 通道的光谱功率数据，通过波长插值与数据标准化构建光谱线性叠加模型，并将各通道权重作为优化变量。在满足 CCT、Rf、Rg 及 mel-DER 约束的条件下，采用分层优化策略，先使用非负最小二乘算法生成多组可行初解，再以带惩罚项的 SLSQP 处理非线性约束，同时引入差分进化降低局部最优风险。结果显示，日间模式以冷白光（72.19%）与绿光（11.48%）为主，CCT 约 6108.7 K， $R_f = 92.07$ ， $R_g = 102.35$ ，mel-DER 为 0.8775；夜间模式以暖白光（87.8%）为核心，减少蓝光，将 CCT 降至 2700 K， $R_f = 91.16$ ， $R_g = 101.60$ ，mel-DER 降至 0.4155，验证了该方法的有效性。

对于问题三，基于太阳光谱功率分布数据，先对可见光波段进行插值与标准化处理，并结合问题二的 LED 通道光谱，建立模拟自然光节律变化的光谱拟合模型，将各通道权重设为时序优化变量。利用非负最小二乘获得全时段初始解，后引入 Savitzky–Golay 平滑保证时间连续性。对于所选的三个代表性时刻，进一步使用 DE 算法进行全局优化，以提升拟合精度。结果显示，全时段 CCT 平均绝对误差为 874.79 K，mel-DER 平均绝对误差为 0.2167，说明该策略能在全天范围有效重构太阳光节律特性。

对于问题四，搭建了睡眠质量评估的统计分析模型，采用被试内重复测量设计，提取 TST、SE 等六项核心指标。优先以重复测量方差分析检验主效应，当正态性或方差齐性条件不满足时，转为 Friedman 非参数检验，并在显著时进行 Bonferroni 校正的配对比较。结果显示，三种光照条件下各指标差异均未达显著性 ($p > 0.05$)，但优化光照下 SOL 缩短与 N3% 提高趋势明显，提示其具有一定改善睡眠质量的潜力。

关键词：多通道 LED 光源；生理节律效应；多目标优化；序列二次规划；差分进化算法；非参数检验

一、问题重述

1.1 问题背景

发光二极管（Light Emitting Diode, LED）光源是通过半导体 PN 结将电能转化为光能的半导体器件，凭借其高效、节能、环保的特性在多个领域广泛应用，尤其在照明领域，白光 LED 效率远超传统光源。

室内 LED 照明具有光谱可设计性强、亮度与色温可调节、深度关联人体生理节律等特点，并且受到多种因素的制约和影响。LED 光谱设计需平衡视觉需求与生理需求，灵活组合出多样光照场景，同时合适的光谱可同步调节视锥细胞与黑视蛋白的响应，影响褪黑素分泌、睡眠质量及情绪状态 [1]。然而，在实际应用中，随着大量非自然光入侵生活，昼夜分明的环境变得模糊，人们的睡眠出现不规律甚至失控，继而对健康产生威胁。因此，如何改善照明系统以改变相关色温（CCT）来调节白光颜色，对昼夜节律施加一定控制尤为重要。合理的优化策略可以减少人们的健康风险，契合可持续发展理念，在节能的同时提升公众健康水平。

1.2 问题要求

为实现优化 LED 光源的光谱特性，需要结合不同波长、LED 类型等条件下的光强数值，构建 LED 光源相关参数之间的关系模型，设计合理的优化与控制策略。附录分别为不同波长对应的光强数值、不同类型 LED 组合后的光谱特性、不同波长在不同时间点的光强数值、睡眠实验的记录数据，现需结合上述数据，分析并解决下列问题：

对于问题 1，基于给定的 LED 光源光谱功率分布（SPD）数据，建立计算模型，求解相关色温（CCT）、距离普朗克轨迹的距离 (D_{uv})、保真度指数 (R_f)、色域指数 (R_g)、褪黑素日光照度比（mel-DER）这五个核心参数。

对于问题 2，利用五个独立 LED 通道的 SPD 数据，通过寻找最佳权重组合，分别针对日间照明模式和夜间助眠模式合成满足特定需求的光谱，并得出关键参数与权重组合。

对于问题 3，结合五通道 LED，设计控制策略，使合成光谱在全天范围模拟给定的太阳光谱数据，选取三个代表性时间点进行案例分析。

对于问题 4，根据提供的临床睡眠实验数据，计算睡眠质量评估指标，运用统计检验方法分析三种光照环境对各项睡眠指标的影响是否存在显著性差异，判断设计的“优化光照”是否对睡眠质量有有益改善。

二、问题分析

2.1 问题一分析

对于问题一，要求基于给定的 SPD（光源的光谱功率分布）数据，通过标准化模型计算五个核心参数。因此，本文首先对附录给出的 SPD 数据进行初步可视化，排除异常值。然后将实测光谱功率分布 $S(\lambda)$ 按 1nm 间隔归一化，并补全 $380\text{--}780\text{nm}$ 范围，接着在同一波长网格上并行计算三类核心指标：颜色外观、颜色还原、节律效应。

此后，结合文献 [2] 中的公式算法，将连续的 SPD 曲线 ($\lambda=380\text{--}780\text{nm}$) 转化为五个关键参数：CCT (K)、 D_{uv} (无量纲)、 R_f (0-100)、 R_g (0- ∞)、mel-DER (无量纲)，这些参数将用于评估光源的颜色质量和生理节律效应。

2.2 问题二分析

对于问题二，要求考虑五个 LED 通道的 SPD 数据（红光、绿光、蓝光、暖白光和冷白光），设计一个多通道 LED 光源以满足特定的照明需求，并优化合成光谱的 CCT、 D_{uv} 、 R_f 、 R_g 和 mel-DER 等参数。

首先，需要结合题目给出的数据和问题一解得的参数模型，SPD 数据通过加权线性组合的方式合成总光谱。此后，使每个通道的光谱与其权重相乘并加总，进而形成一个新的光谱。在设计权重调整机制后，我们将目标函数转换为惩罚式标量目标函数，并采用非负最小二乘和序列二次规划求解，使得合成光谱的 CCT、Rf、Rg、mel-DER 等参数达到最佳状态。

2.3 问题三分析

对于问题三，要求基于太阳光谱的时间序列数据，设计包含问题二中蓝光、绿光、红光、暖白光、冷白光的五通道 LED 的全天动态控制策略，使合成光谱在 12 个时间点匹配太阳光谱的节律特征（光谱形状、CCT、mel-DER）。同时，应输出控制策略并选取三个代表性时间点进行光谱对比与案例分析。

首先，需要确保太阳光谱的节律特征匹配三项指标：物理属性的光谱形状、色温 CCT 以及生理属性的褪黑素抑制效应 mel-DER）。进而我们可以根据问题二得到的静态模型，对五通道 LED 的权重实现动态调控，最后获得 12 个时间点的五通道 LED 权重矩阵，使得合成光谱在各时间点均能匹配太阳光谱的光谱形状、CCT、mel-DER，并满足 $R_f \geq 80$ 。

2.4 问题四分析

对于问题四，要求根据健康被试在不同睡前光照环境下的整夜睡眠记录，通过指标计算、统计分析等评估优化光照对睡眠质量的改善效果。

对此，需要通过科学的数据分析与统计建模，验证“优化光照环境（A）”对人类睡眠质量的改善效果，形成“理论设计→实验验证”的闭环。我们引入新的睡眠参数，将原始睡眠阶段编码转化为TST、SE等可统计的睡眠参数，进而控制个体差异，检验光照环境的主效应，最后定位具体差异来源，并进行分析评估。

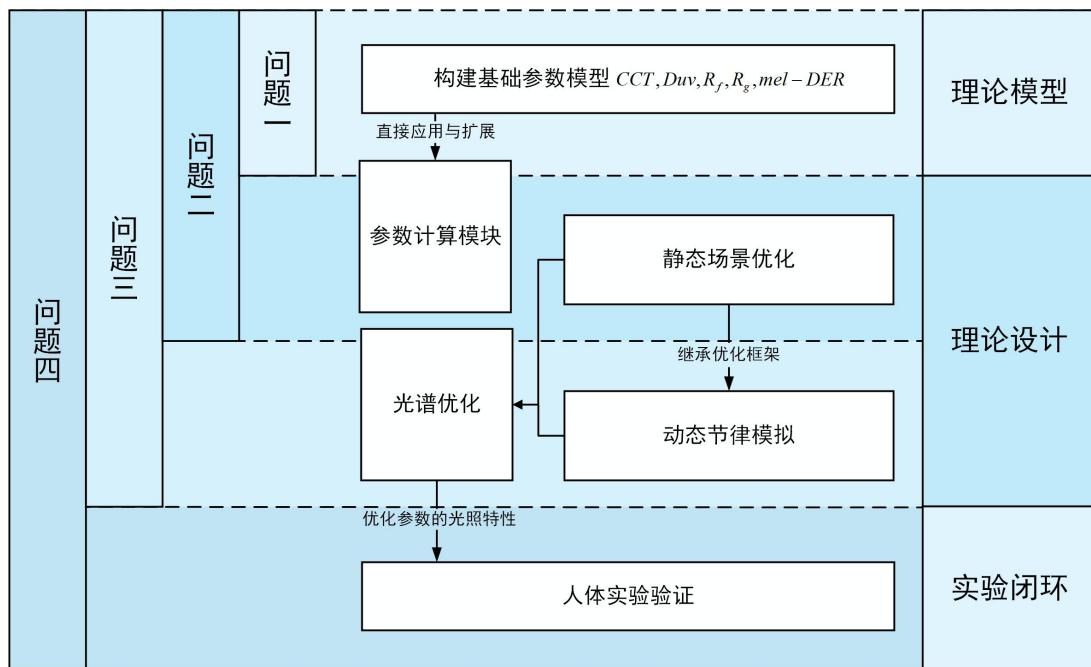


图1 本文主要问题分析思路

三、模型假设

为了方便我们分析并简化问题，在确保合理性的前提下，我们做出以下假设：

- (1) 附件中LED光源的光谱功率分布(SPD)数据真实有效，可直接用于核心参数计算。
- (2) 多通道LED合成光谱为各通道光谱的加权线性叠加，无其他非线性光学作用。
- (3) 睡眠实验中，被试在三种光照环境下的体验相互独立，无顺序或交互影响。
- (4) 太阳光谱数据的时间序列连续且具有代表性，可通过五通道LED权重调节实现模拟。
- (5) 编码准确反映被试真实状态，数据无系统性偏差。

四、 符号说明

符号	符号说明	单位
$J(\mathbf{w})$	日间、夜间模式的惩罚式 目标函数	/
\mathbf{w}	5个LED通道的权重向量	/
$\text{Loss}_{\text{spec}}(\mathbf{w})$	合成光谱与目标光谱的匹 配损失	/
$\text{mel-DER}(\mathbf{w})$	合成光谱的视黑素日光效 率比	/
P_{CCT}	色温惩罚系数	/
$\text{CCT}(\mathbf{w})$	合成光谱的相关色温	K
Δ_{CCT}	色温的允许偏差阈值	K
P_{R_f}	显色性惩罚系数	/
$R_f(\mathbf{w})$	合成光谱的保真度指数	/
$\bar{x}(\lambda)、\bar{y}(\lambda)、\bar{z}(\lambda)$	CIE 1931 标准观察者的三 刺激值函数	/
X, Y, Z	CIE XYZ 色彩空间的三刺 激值	/
(x, y)	色品坐标	/
$W_{\text{mel}}(\lambda)$	Melanopsin 蛋白的光谱响 应函数	/
$s_i(\lambda)$	第 i 个 LED 通道的光谱功 率分布向量	$W/(m^2 \cdot nm)$
Q	Friedman 检验的统计量	/
W	Wilcoxon 符号秩检验的正 秩和统计量	/
Z	Wilcoxon 检验的标准化统 计量	/
RBC	非参数检验的效应量	/

* 注：限于篇幅，表中有个别符号尚未列出，我们将在具体的建模分析部分中详细说明。

五、模型的建立和求解

5.1 问题一模型的建立和求解

5.1.1 数据预处理

为实现光源颜色特性与生理节律参数的精准计算，需先明确“物理光谱 → 视觉感知”的转化路径。图 2 展示了人眼对不同波长光的颜色响应规律，其中 $\bar{x}(\lambda)$ 、 $\bar{y}(\lambda)$ 、 $\bar{z}(\lambda)$ 分别对应红、绿、蓝通道的光谱敏感特性，是将物理光谱转换为视觉参数的核心工具。

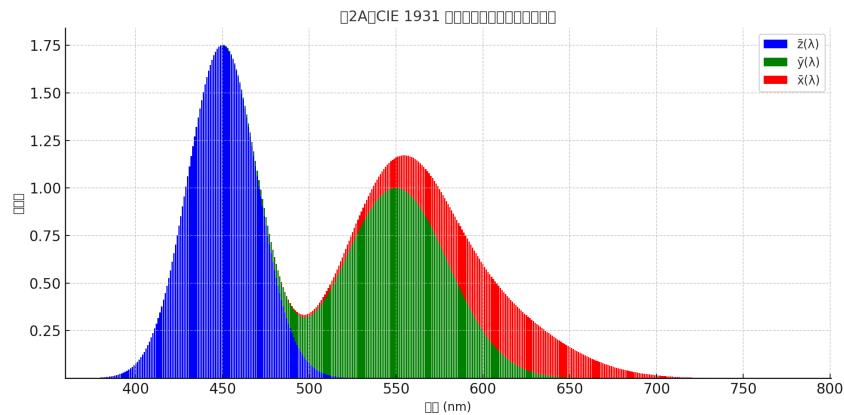


图 2 CIE 1931 三刺激值函数的堆叠柱状图

基于这一理论基础，在计算核心参数前，需对光谱功率分布 (SPD) 数据进行异常值识别与处理，确保输入数据的可靠性。在处理附录过程中发现，根据 CIE 标准定义，仅保留 380 - 780nm 的可见光区间数据，若原始数据包含小于 380nm (紫外) 或大于 780nm (红外) 的波长，直接截断。此外，光谱功率需为非负值，若存在 $P(\lambda) < 0$ 的值，即为异常点，需使用相邻波长点的功率值线性插值替换。剔除异常值后，所绘制的光谱功率分布情况如图 3 所示。

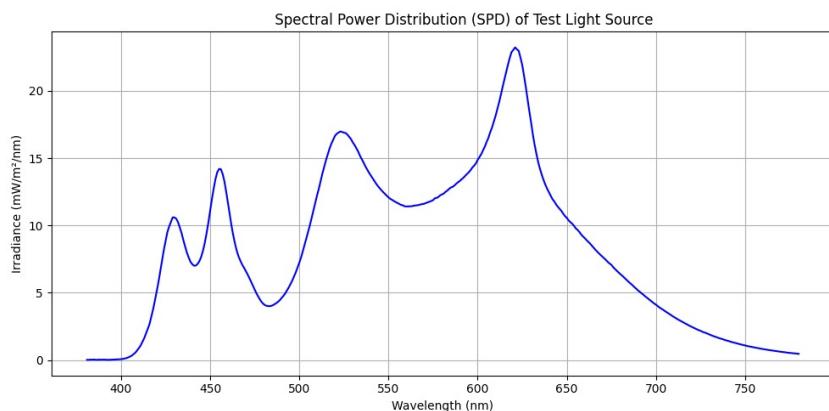


图 3 测试光源的光谱功率分布情况

5.1.2 三刺激值 (X, Y, Z) : 光谱到颜色的基础转换

在完成光谱功率分布 (SPD) 数据的预处理并剔除异常值后，我们着手建立颜色特性参数 (CCT, D_{uv}) 计算模型，将 SPD 数据转换到 CIE XYZ 色彩空间，对每个波长计算 RGB 通道的贡献。三刺激值是光源光谱功率与人类视觉响应的综合效应，是所有颜色参数的计算基础。 X 、 Y 、 Z 分别对应人眼三种视锥细胞的响应，其中 Y 值直接关联光源亮度。根据 CIE 1931 标准，三刺激值通过 SPD 与观察者函数的积分定义如下：

$$X = \int P(\lambda) \cdot x(\lambda) d\lambda \quad (1)$$

$$Y = \int P(\lambda) \cdot y(\lambda) d\lambda \quad (2)$$

$$Z = \int P(\lambda) \cdot z(\lambda) d\lambda \quad (3)$$

其中， $P(\lambda)$ 是波长 λ 处的光谱功率， $x(\lambda), y(\lambda)$ ，和 $z(\lambda)$ 是 CIE 标准观察者的色彩匹配函数， $d\lambda$ 是波长范围内的积分间隔，通常为 1 nm。然而，在实际计算中，由于 SPD 数据和色彩匹配函数的离散性质，需要将积分转换为离散求和。其中积分区间分为 400 个 1nm 小区间，每个区间面积用梯形近似，即：

$$X = \sum_{i=1}^n P(\lambda_i) \cdot x(\lambda_i) \cdot \Delta\lambda \quad (4)$$

$$Y = \sum_{i=1}^n P(\lambda_i) \cdot y(\lambda_i) \cdot \Delta\lambda \quad (5)$$

$$Z = \sum_{i=1}^n P(\lambda_i) \cdot z(\lambda_i) \cdot \Delta\lambda \quad (6)$$

在公式 (4) (5) (6) 中， $P(\lambda_i)$ 代表 λ_i 处的光谱功率， $\Delta\lambda$ 是波长间隔（例如 1 nm）， λ_i 是波长点。将 SPD 数据转换到 CIE XYZ 色彩空间后，CIE XYZ 的值可以被标准化或归一化以适应不同的应用。

5.1.3 颜色特性参数 (CCT, D_{uv}) 计算模型的构建与求解

根据问题要求与分析，需通过标准化计算将 SPD 数据转化为描述光源颜色外观的参数。其中，CCT 描述光源“冷暖”程度（如 5000K 为中性白，6500K 为冷白）； D_{uv} 描述色坐标偏离普朗克轨迹的程度（ $D_{uv}=0$ 表示完全匹配黑体）。

①色品坐标 (x, y) 转换

我们知道，XYZ 三刺激值虽然能够从物理层面描述光的颜色特性，但它在反映人眼对颜色的感知均匀性方面存在不足。色品坐标是描述光源或物体颜色在色度图上位置的坐标参数，用于定量表示颜色的“色相”和“饱和度”。因此，为了更精准地量化颜色外观，需要将 XYZ 坐标转换为色品坐标 (x, y) ，其转换公式如下 [3]：

$$x = \frac{X}{X + Y + Z}, \quad y = \frac{Y}{X + Y + Z} \quad (7)$$

其中， X 、 Y 和 Z 是通过 SPD 转换得到的 CIE XYZ 三刺激值，而分母 $X+Y+Z$ 则代表归一化系数。

②相关色温 (CCT) 的计算

相关色温 (CCT) 是光源点到黑体辐射轨迹上距离最近点所对应的黑体温度 (单位为 Kelvin)。对于相关色温 CCT，我们希望找到一个中间变量 n ，将 n 值代入三次多项式进而求得相关色温 CCT。

在算法选择的过程中，我们分别尝试使用文献 [2] 中的 McCamy 近似公式法、三角垂足插值法和 Chebyshev 法进行求解，然而三角垂足插值法和 Chebyshev 法的误差大于 3%，可能存在隐患。最后我们选择使用 **McCamy 近似公式法** 来计算 CCT，并为保证与文献的一致性，我们采用了 1931 标准观察者数据集。首先，结合待测色品坐标 (x,y) 与黑体轨迹的关系得出中间变量 n 。具体公式如下：

$$n = \frac{x - 0.3320}{y - 0.1858} \quad (8)$$

式中 0.3320 和 0.1858 为固定参数，用于标准化色品坐标的偏移量。中间变量 n 计算完成后，即可代入近似公式计算 CCT，得到相关色温 T (单位：K)，公式如下：

$$T = -437n^3 + 3601n^2 - 6861n + 5514.31 \quad (9)$$

其中，计算结果 T 即为待测光源的相关色温 CCT，该方法适用于快速估算，尤其适合便携式仪器或对计算速度要求较高的场景。

③距离普朗克轨迹的距离 (D_{uv}) 的计算

距离普朗克轨迹的距离 (D_{uv}) 表示待测光源点相对于黑体轨迹的偏差程度。首先，通过相关色温 CCT 的计算，我们可以得到待测光源的色品坐标 (x,y) 。首先计算对应黑体轨迹点色品坐标 (u,v) ：

$$x = \frac{4X}{X + 15Y + 3Z}, \quad y = \frac{6Y}{X + 15Y + 3Z} \quad (10)$$

然后，计算待测光源的色品坐标 (x,y) 与对应黑体轨迹点色品坐标 (u,v) 之间的欧几里得距离，即：

$$\text{距离} = \sqrt{(x - u)^2 + (y - v)^2} \quad (11)$$

进而我们可以通过计算符号函数 $\text{sign}(u-x)$ 来判断待测光源点相对于黑体轨迹的位置，具体公式如下：

$$\text{sign}(u - x) = \begin{cases} 1 & \text{如果 } u > x \\ -1 & \text{如果 } u < x \\ 0 & \text{如果 } u = x \end{cases} \quad (12)$$

结合待测光源点 (x, y) 相对于黑体轨迹的方向以及待测点与最近黑体轨迹点的欧几里得距离, D_{uv} 最终计算公式为:

$$D_{uv} = \text{sign}(u - x) \times \text{距离} \quad (13)$$

D_{uv} 计算通过“坐标距离量化 + 符号函数定向”, 将光源色偏差的评估从单一数值对比, 升级为包含空间方位信息的综合判断, 为光源光谱优化、色品质分级提供精准依据。

总结

数据集 CIE 1960 中的色度图通过 McCamy 近似法求解结果如图 4 所示:

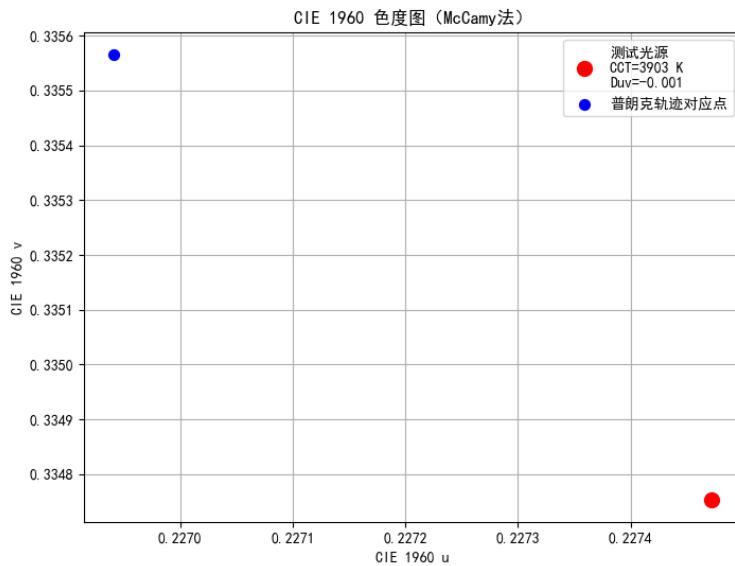


图 4 McCamy 法求解 CIE 1960 色度图

所得到 CCT 与 D_{uv} 取值如下表所示:

表 1 问题一 CCT 与 D_{uv} 取值

CCT(K)	D_{uv}
3903.2	-0.0010

5.1.4 颜色还原参数 (R_f, R_g) 计算模型的构建与求解

在求得颜色特性参数后, 需要通过对比色样在待测光源与标准光源 (D65) 下的颜色差异, 量化光源对颜色的还原能力 (R_f) 和色域范围 (R_g)。其中, R_f 表示测试光源下物体颜色与参考光源下颜色的相似程度, 范围 0–100, 越接近 100 表示颜色还原度越高;

R_g 描述测试光源相对于参考光源的色域面积变化， $R_g=100$ 表示色域面积一致 >100 表示色域扩展（更鲜艳） <100 表示收缩（更黯淡）。

①计算模型的核心框架

基于 ANSI/IES TM-30-20 标准， R_f （保真度指数）和 R_g （色域指数）的计算模型以四要素核心框架为基础 [4]，具体包括：

- CIE 1964 10° 色匹配函数：表征人眼对可见光的平均光谱响应，用于计算三刺激值。
- CAM02-UCS 色空间：具有感知均匀性，确保色差值与人类视觉感知一致。
- 99 个颜色评价样本 (CES)：覆盖建筑环境中典型物体的光谱反射率，包括皮肤、颜料、自然物体等。
- CCT 匹配的参考光源：根据测试光源的相关色温 (CCT) 确定参考光源。当 $CCT \leq 4000K$ 时，参考为同色温的普朗克辐射体；当 $CCT \geq 5000K$ 时，参考为同色温的 CIE D 系列光源；当 $4000K < CCT < 5000K$ 时，参考为普朗克辐射体与 D 系列光源的比例混合。

②保真度指数 R_f 的计算

对于保真度指数 R_f ，我们希望借助前文求解得到的测试光源的光谱功率分布，结合测试光源和参考光源下的三刺激值，将三刺激值转换为 CAM02-UCS 色空间坐标，进而 99 个 CES 的平均色差值，最后转换为 R_f 。

首先，对 15 个 CIE 测试色样 ($i=1..15$)，分别计算待测光源与 D65 下的三刺激值，以此来反映色样在不同光源下的颜色。在颜色特性参数的模型构建部分，已确定三刺激值的计算公式 (1) (2) (3)，因此，仅需分别添加色样反射率为 $\rho_i(\lambda)$ 和反射光 $P_{\text{ref}}(\lambda)\rho_i(\lambda)$ 即可进行进一步计算。进一步地，我们将待测光源色样的三刺激值定义为 X_i, Y_i, Z_i ，将 D65 色样的三刺激值定义为 $X_{\text{ref},i}, Y_{\text{ref},i}, Z_{\text{ref},i}$ 。

计算完成后，将三刺激值转换为 CAM02-UCS 色空间坐标 (a', b', J') ，聚焦于色度差异 (a', b') 。接着对每个 CES，计算测试光源与参考光源下的色差值 ΔE_i ，即：

$$\Delta E_i = \sqrt{(a'_{\text{test}} - a'_{\text{ref}})^2 + (b'_{\text{test}} - b'_{\text{ref}})^2} \quad (14)$$

根据 TM-30，对所有 99 个色样的色差进行非线性变换与平均，得到公式如下：

$$R_f = 100 - \frac{1}{n} \sum_{i=1}^n (c \cdot \Delta E_i) \quad (15)$$

其中 c 为标定系数， $n = 99$ ，即 TM-30 规范中的标准色样数，并在计算中引入对低值的对数压缩，使得 $R_f \in [0, 100]$ 。

③色域指数 R_g 的计算

对于色域指数 R_g ，首先划分色调角区间，然后计算区间平均坐标，最后分别以 16 个顶点构建测试光源和参考光源的 16 边形，采用多边形面积公式计算色域面积。

根据文献 [4]，将 99 个 CES 按参考光源下的色调角划分为 16 个等宽区间（每区间 22.5° ），覆盖红、黄、绿、蓝等典型色调。由于 R_g 用于表示测试光源相对参考光源的色

域面积变化，需要对每个区间计算测试光源和参考光源下所有 CES 的 (a', b') 坐标平均值，进而得到 16 个顶点坐标：

$$(a'_{\text{test},h}, b'_{\text{test},h}), \quad (a'_{\text{ref},h}, b'_{\text{ref},h}) \quad (h = 1, 2, \dots, 16)$$

基于 16 个顶点，分别构建测试光源和参考光源的 16 边形，采用多边形面积公式计算面积 A_{test} 和 A_{ref} ，具体公式如下：

$$A = \frac{1}{2} \left| \sum_{h=1}^{16} (a'_h b'_{h+1} - a'_{h+1} b'_h) \right| \quad (a'_{17} = a'_1, b'_{17} = b'_1) \quad (16)$$

基于所求得的参考光源色域面积，计算相对色域面积之比，即：

$$R_g = 100 \times \frac{A_{\text{test}}}{A_{\text{ref}}} \quad (17)$$

其中， A_{test} 为测试光源色域在 CAM02-UCS 空间中的面积， A_{ref} 为参考光源对应面积。 R_g 值大于 100 表示偏饱和，小于 100 表示偏灰。

总结

得到 R_f 与 R_g 取值如下表所示：

表 2 问题一 R_f 与 R_g 取值

R_f	R_g
91.79	106.08

5.1.5 生理节律效应参数（mel-DER）计算模型的构建与求解

mel-DER 计算待测光源对人体褪黑素的抑制效应与标准光源 D65 的比值，量化光源对人体生理节律的影响。

① 光源的 Melanopsin 加权辐射

Melanopsin 敏感性函数描述了不同波长的光对褪黑素的抑制效果。它通常是一组标准值。计算 mel-DER 的过程中，使用波长 (BC) 和光强 (GQ) 来计算光源的 melanopsin 加权辐射。这个过程是对每个波长上的光强进行加权，得到每个波长对褪黑素分泌的贡献。

首先，需要通过积分计算光源对褪黑素分泌的“有效辐射”，具体公式如下：

$$I_{\text{source}} = \int_{380}^{780} S(\lambda) \cdot W_{\text{mel}}(\lambda) \cdot d\lambda \quad (18)$$

其中， $S(\lambda)$ 表示光源在波长 λ 处的光谱功率， $W_{\text{mel}}(\lambda)$ 代表 CIE S 026 中的 melanopsin 蛋白的光谱响应函数。与此同时，为了建立对比基准，还需要计算 D65 光源的 melanopic

加权辐射。由于计算公式与式(18)相同，因此直接进行计算，并将D65光源的melanopic加权辐射定义为 I_{D65} 。最后，即可分别求得光源与D65的加权辐射。

②褪黑素日光度比(mel-DER)的计算

完成光源的Melanopsin加权辐射后，我们可以通过光源与D65的加权辐射比定义mel-DER：

$$\text{mel-DER} = \frac{I_{\text{source}}}{I_{D65}} \quad (19)$$

其中，若mel-DER > 1，光源对褪黑素分泌的抑制强于自然光，可能干扰日间清醒节律；若mel-DER < 1，则抑制作用弱于自然光，适合夜间助眠场景。为消除光强绝对值影响，可对 $S(\lambda)$ 归一化（总光通量为1），保证mel-DER仅反映光谱形状对生理节律的影响。

总结

最终，得到mel-DER取值为0.6407，表明该光源在相同照度下的褪黑素抑制作用约为D65的64%，属于中等偏低水平，适合夜间或放松场景

表3 问题一mel-DER取值

mel-DER (相对于D65)
0.6407

5.2 问题二模型的建立和求解

根据问题一求得的光谱参数计算的基础数学模型(CCT、 D_{uv} 、 R_f 、 R_g 、mel-DER)，可以直接应用与扩展五个参数值并进行相关计算。在问题二中，需要将光谱功率分布(SPD)数据拆分为5个LED通道，记5个通道的SPD向量为 $s_i(\lambda)$ ，通道权重为 $w_i \geq 0$ ，合成光谱为：

$$S(\lambda) = \sum_{i=1}^5 w_i \cdot s_i(\lambda) \quad (20)$$

依据文献[5]，我们通过对 $S(\lambda)$ 进行颜色学积分（与CIE色度匹配函数卷积）得到色度坐标(x, y)，进而计算CCT、 D_{uv} 、 R_f 、 R_g 及melanopic DER。在问题二中，需要在保证光谱拟合精度的同时，满足题目给定的非线性约束条件。对此，我们决定采用NNLS+SLSQP两阶段优化思路，引入多起点初始化与差分进化(DE)优化，以提高收敛稳定性和全局最优概率。具体的思路流程如图5所示。

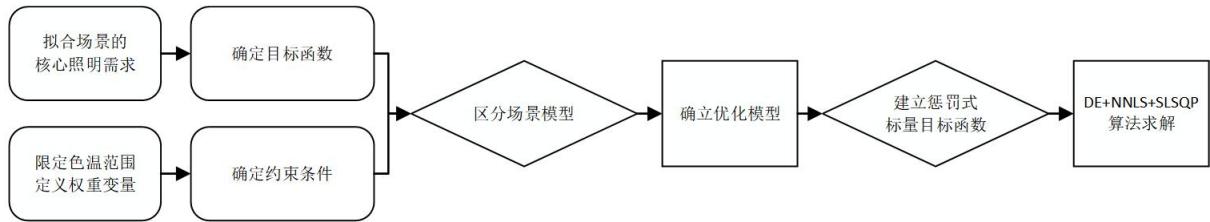


图 5 问题二思路流程图

5.2.1 数据预处理

在使用数据进行建模之前，需要对数据进行预处理，以减少计算误差，加速收敛。在处理附件第二页给出的数据集中发现，同问题一的数据预处理过程相似，首先需确保光谱数据聚焦人眼可见的 380–780 nm 区间，以保证光谱连续性与计算精度。此后，由于原始光谱功率数值跨度大，直接用于优化易导致数值震荡、收敛困难。因此对功率做归一化处理，具体公式如下：

$$P_{\text{norm}}(\lambda_i) = \frac{P(\lambda_i)}{\sum_{j=380}^{780} P(\lambda_j) \cdot \Delta\lambda} \quad (21)$$

最后即可完成数据清洗与归一化。

5.2.2 多通道光源优化模型建立

① 决策变量与目标函数

根据题目要求与上文分析，我们开始建立优化模型。由于题目要求要利用 5 个 LED 通道，通过寻找最佳的权重组合来合成满足特定需求的光谱，因此，我们将权重变量设置为五个通道的权重变量 $\mathbf{w} = [w_1, w_2, w_3, w_4, w_5]^T$ ($w_i \geq 0$ 且 $\sum_{i=1}^5 w_i = 1$)，合成光谱运用公式 (20) 进行计算。其中， w_1 对应深红光通道， w_2 对应绿光通道， w_3 对应蓝光通道， w_4 对应暖白光 (WW) 通道， w_5 对应冷白光 (CW) 通道。这些权重需满足非负且总和为 1 的约束，即 $w_i \geq 0$ 且 $\sum_{i=1}^5 w_i = 1$ ，通过调节权重可动态控制合成光谱的特性。

要建立通道光源优化模型，我们需要确定优化模型的目标函数。根据题意，首先我们需要将模型区分为日间照明模式和夜间助眠模式两个不同场景。对于日间照明模式，题目要求采用最大化保真度指数 R_f ，即 $\max_{\mathbf{w}} R_f(\mathbf{w})$ ，其中根据公式 (15)， $R_f = 100 - 4.6 \cdot \bar{\Delta E}$ ， $\bar{\Delta E}$ 为 15 个 CIE 色样的平均色差；对于夜间助眠模式，题目要求采用最小化褪黑素抑制效应 mel-DER，即式 (23)，其中 $P_{\text{ref}}(\lambda)$ 为 D65 标准光源 SPD， $CS(\lambda)$ 为褪黑素抑制函数。最终，我们设定的不同场景目标函数如下：

a. 日间照明模式的目标函数

$$\max_{\mathbf{w}} R_f(\mathbf{w}) \quad (22)$$

b. 夜间助眠模式的目标函数

$$\min_{\mathbf{w}} \text{mel-DER}(\mathbf{w}) = \frac{\int_{380}^{780} \text{SPD}_{\text{total}}(\lambda) \cdot \text{CS}(\lambda) d\lambda}{\int_{380}^{780} P_{\text{ref}}(\lambda) \cdot \text{CS}(\lambda) d\lambda} \quad (23)$$

②约束条件

在得到目标函数之后，我们需要根据题目所给的要求与限制设置约束条件，使所得的解能够满足要求。根据题目信息，我们综合考虑了色温约束、显色性约束和权重约束，设定约束条件如下：

a. 色温约束

由于日间模式的高清醒度需求，需要模拟自然日光的偏冷色调，维持人眼警觉性与工作效率。得到约束条件如下：

$$5500 \leq \text{CCT}(\mathbf{w}) \leq 6500 \text{ K} \quad (24)$$

同时，基于夜间模式的助眠需求，需营造温暖柔和的氛围，减少蓝光对褪黑素分泌的抑制。得到约束条件如下：

$$2500 \leq \text{CCT}(\mathbf{w}) \leq 3500 \text{ K} \quad (25)$$

b. 显色性约束

显色性反映光源还原物体真实颜色的能力，通过保真度指数 R_f 和色域指数 R_g 量化。首先，对于日间模式，需要保证物体颜色与自然光下的偏差极小，满足精细工作对颜色真实性的要求，同时要求控制色域“适度扩展”，可得到如下约束条件：

$$R_f(\mathbf{w}) > 88 \quad (26)$$

$$95 \leq R_g(\mathbf{w}) \leq 105 \quad (27)$$

同时，夜间模式仅需优先保障基本颜色辨识度，所以得到如下约束条件：

$$R_f(\mathbf{w}) \geq 80 \quad (28)$$

c. 权重约束

在实际应用中，权重向量（即 5 个 LED 通道的功率占比）需要满足“非负性”和“归一性”。其中，非负性代表在物理上，通道功率占比不能为负；归一性则要求总功率占比需为 100%，保证模型与实际电路的功率分配逻辑一致。进而可得到如下约束条件：

$$w_i \geq 0 \quad (i = 1, 2, \dots, 5) \quad (29)$$

$$\sum_{i=1}^5 w_i = 1 \quad (i = 1, 2, \dots, 5) \quad (30)$$

综合分析题目与附件中的要求，我们得到了 7 条约束条件，如式(24) - 式(30)所示。

表 4 多通道光源优化模型约束条件汇总

限制条件类型	限制模式	限制条件内容	对应公式编号
色温约束	日间模式	模拟自然日光的偏冷色调	24
	夜间模式	营造温暖柔和的氛围	25
显色性约束	日间模式	保证物体颜色与自然光下的偏差极小	26
	日间模式	控制色域“适度扩展”	27
	夜间模式	保障基本颜色辨识度	28
权重约束	两者	非负性	29
	两者	归一性	30

③优化模型建立

上文已经建立了目标函数与约束条件，将其整合为完整的多通道光源优化模型，并且将其区分为不同的场景模型，即式(31) (32)。

a. 日间照明模式的优化模型

$$\begin{aligned} & \max_{\mathbf{w}} R_f(\mathbf{w}) \\ \text{s.t. } & \left\{ \begin{array}{l} 5500 \leq \text{CCT}(\mathbf{w}) \leq 6500 \text{ K} \\ R_f(\mathbf{w}) > 88 \\ 95 \leq R_g(\mathbf{w}) \leq 105 \\ w_i \geq 0 \quad (i = 1, 2, \dots, 5) \\ \sum_{i=1}^5 w_i = 1 \quad (i = 1, 2, \dots, 5) \end{array} \right. \end{aligned} \quad (31)$$

b. 夜间助眠模式的优化模型

$$\begin{aligned} & \min_{\mathbf{w}} \text{mel-DER}(\mathbf{w}) = \frac{\int_{380}^{780} \text{SPD}_{\text{total}}(\lambda) \cdot \text{CS}(\lambda) d\lambda}{\int_{380}^{780} P_{\text{ref}}(\lambda) \cdot \text{CS}(\lambda) d\lambda} \\ \text{s.t. } & \left\{ \begin{array}{l} 2500 \leq \text{CCT}(\mathbf{w}) \leq 3500 \text{ K} \\ R_f(\mathbf{w}) \geq 80 \\ w_i \geq 0 \quad (i = 1, 2, \dots, 5) \\ \sum_{i=1}^5 w_i = 1 \quad (i = 1, 2, \dots, 5) \end{array} \right. \end{aligned} \quad (32)$$

5.2.3 模型求解

上述模型的建立用于通过调节五个LED通道的权重，优化光源的色温(CCT)、显色性(R_f/R_g)和生理节律效应(mel-DER)。因此，我们决定采用分层优化策略：首先先用非负最小二乘(NNLS)在目标光谱上快速生成多重初始解，以覆盖解空间；然后以若干NNLS解为初值，根据文献[6]，可以使用带惩罚项的序列二次规划(SLSQP)对

光谱拟合与生物节律指标进行精修；如需增强全局性，可采用差分进化（DE）做短时全局搜索再由 SLSQP 抛光。该方法兼顾了线性解析速度与非线性约束的高精度处理能力，并能在有限时间内获得稳定且物理可实现的解。算法设计的具体步骤如下：

- NNLS 计算初始解：采用多起点策略随机扰动初始目标，得到若干可行解。
- SLSQP 优化：使用 SLSQP 优化处理非线性约束，取最优解作为最终输出。
- 全局优化辅助（差分进化 DE）：在 NNLS 与 SLSQP 之间插入 DE 以避免局部最优，并对比前后收敛效果。

在模型求解的过程中，我们发现由于不同场景给出的约束条件不同，因此为了将多目标优化问题转换为单目标优化问题，我们将部分约束转为惩罚，采用惩罚函数法构造惩罚式标量目标函数，当约束条件被违反时，惩罚项增大，从而对不满足约束的解进行惩罚，以最小化与理想值的偏差。

a. 日间照明模式的惩罚式标量目标函数

对于日间照明模式，根据约束条件，合成谱尽量接近日光目标 \mathbf{T}_{day} 且满足 R_f 较高、CCT 在允许区间，构建的惩罚式标量目标函数具体如下：

$$\begin{aligned} J_{\text{day}}(\mathbf{w}) = & \text{Loss}_{\text{spec}}(\mathbf{w}) + \eta_{\text{mel}}^{(\text{day})} \cdot \text{mel-DER}(\mathbf{w}) \\ & + P_{\text{CCT}} \cdot \max(0, |\text{CCT}(\mathbf{w}) - \text{CCT}_{\text{day}}| - \Delta_{\text{CCT}})^2 \\ & + P_{R_f} \cdot \max(0, R_f^{(\text{day})} - R_f(\mathbf{w}))^2 \end{aligned} \quad (33)$$

其中， $\eta_{\text{mel}}^{(\text{day})}$ 在日间通常设置较小或为 0（若无需最小化 mel）， P_* 为惩罚系数。

b. 夜间助眠模式的惩罚式标量目标函数

对于夜间助眠模式，根据约束条件，在色温目标范围内应最小化 mel 值并保证 R_f 不低于阈值，构建的惩罚式标量目标函数具体如下：

$$\begin{aligned} J_{\text{night}}(\mathbf{w}) = & \eta_{\text{mel}}^{(\text{night})} \cdot \text{mel-DER}(\mathbf{w}) + \text{Loss}_{\text{spec}}(\mathbf{w}) \\ & + P_{\text{CCT}} \cdot \max(0, |\text{CCT}(\mathbf{w}) - \text{CCT}_{\text{night}}| - \Delta_{\text{CCT}})^2 \\ & + P_{R_f} \cdot \max(0, R_f^{(\text{night})} - R_f(\mathbf{w}))^2 \end{aligned} \quad (34)$$

其中，夜间 $\eta_{\text{mel}}^{(\text{night})}$ 较大（把 mel 放在主要目标），而 F_{spec} 用于保持基本的光谱质量与颜色感知。

在上述目标函数中， $J_{\text{day}}(\mathbf{w})$ 为日间模式的惩罚式目标函数，综合光谱匹配损失、生理节律约束及色温与显色性惩罚项； $J_{\text{night}}(\mathbf{w})$ 为夜间模式的惩罚式目标函数，优先最小化视黑素日光效率比，同时保证基础显色性与色温； $\eta_{\text{mel}}^{(\text{day})}$ 是日间模式对生理节律的弱约束系数（通常取 $0.1 \sim 0.3$ ），降低视黑素日光效率比的权重； $\eta_{\text{mel}}^{(\text{night})}$ 是夜间模式对生理节律的强约束系数（通常取 $0.8 \sim 1.0$ ），强化视黑素日光效率比的权重。

完成惩罚式标量目标函数的转换后，我们开始结合非负最小二乘（NNLS）的全局

搜索能力与序列二次规划（SLSQP）的局部精修优势，设计“多起点初始化→局部精修→可选全局抛光”流程进行求解。

①NNLS 多起点初始化（全局搜索种子）

首先，通过构建非负最小二乘模型，以目标光谱 \mathbf{T} （如 D65 标准光源）为拟合基准，构建优化问题，具体公式如下。其中， $M \in \mathbb{R}^{L \times n}$ 为光谱矩阵（ L 为波长点数， $n = 5$ 为 LED 通道数）：

$$\min_{\mathbf{w} \geq 0} \|M\mathbf{w} - \mathbf{T}\|_2^2 \quad (35)$$

然后，为突破单一初始值的局限，对目标光谱 \mathbf{T} 施加多样化扰动，包括微小扰动和归一化扰动。其中微小扰动生成 $\mathbf{T}^{(\text{pert})} = \mathbf{T} + \epsilon \cdot \text{randn}(L, 1)$ ($\epsilon = 10^{-4}$ 为扰动强度， randn 为标准正态噪声)，归一化扰动则对 \mathbf{T} 尝试不同归一化方式（如分别以 $Y = 100$ 、 $Y = 200$ 缩放亮度通道）。

扰动生成后，进而计算各候选解的拟合误差，并选取前 $K = 5$ 个误差最小的解 $\{\mathbf{w}^{(r)}\}_{r=1}^5$ ，作为后续局部精修的初始值。具体公式如下：

$$\text{Err}^{(r)} = \|M\mathbf{w}^{(r)} - \mathbf{T}\|_2^2 \quad (36)$$

②SLSQP 局部精修（带惩罚项约束优化）

在 NNLS 初始解基础上，引入生理节律（mel-DER）、显色性 (R_f)、色温 (CCT) 等惩罚项，精细化优化权重 \mathbf{w} 。进一步地，对每个 NNLS 初始解 $\mathbf{w}^{(r)}$ 执行 SLSQP 优化，最小化相应的 J_{day} 或 J_{night} ，约束 $w_i \geq 0$ ；若需可加入等式约束 $\sum_i w_i = 1$ 以归一化。

完成 SLSQP 优化后，进而为目标函数 $J_{\text{day/night}}$ 推导解析梯度，例如光谱匹配损失的梯度，具体的公式如下：

$$\nabla \text{Loss}_{\text{spec}} = \frac{2}{L} M^T (M\mathbf{w} - \mathbf{T}) \quad (37)$$

结合生理节律、惩罚项的梯度，可使 SLSQP 收敛速度提升 2 - 3 倍。

③差分进化（DE）全局抛光

在计算多通道光源优化模型的过程中，我们考虑到了局部最优的情况。对此，我们通过差分进化算法全局搜索，再用 SLSQP 精修，进一步提升解的质量。首先，进行 DE 全局搜索，以 $J_{\text{day/night}}$ 为目标函数，在权重空间 $\mathbf{w} \in [0, 1]^5$ 且 $\sum w_i = 1$ 内全局搜索，接着再取 DE 搜索得到的最优解 \mathbf{w}^{de} ，再次用 SLSQP 精修。

最终，不同场景模型各自求得的最优权重和各指标取值结果如下：

a. 日间照明模式的求解结果

求得日间光源光谱功率分布对比图如图 6 所示：

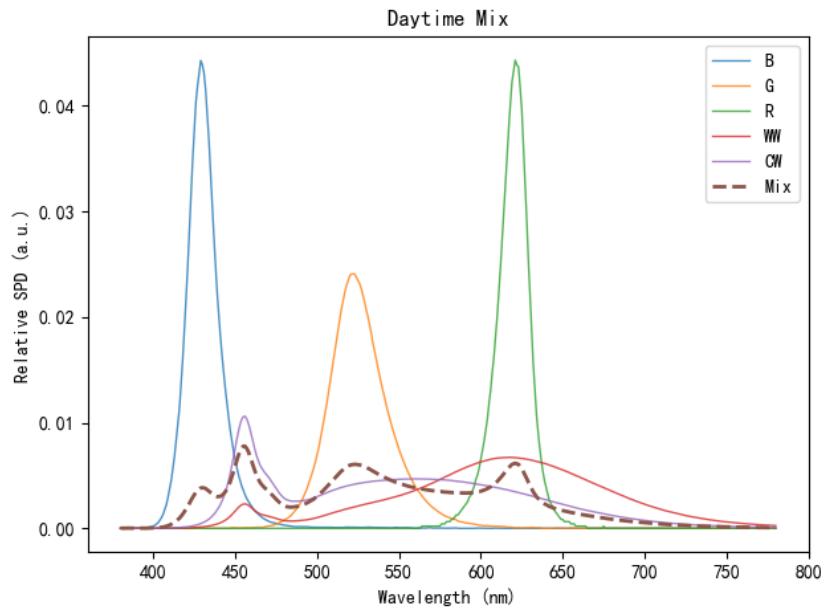


图 6 多通道 LED 混合光源日间光谱功率分布对比图

表 5 日间照明模式的最优权重表

通道	权重
蓝光 B	0.0832
绿光 G	0.1148
深红光 R	0.0729
暖白光 WW	0.0072
冷白光 CW	0.7219

表 6 日间照明模式的各指标取值

指标	取值
CCT	6108.6688
D_{uv}	0.0060
R_f	92.0746
R_g	102.3535
mel-DER	0.8775

b. 夜间助眠模式的求解结果

求得夜间光源光谱功率分布对比图如图 7所示：

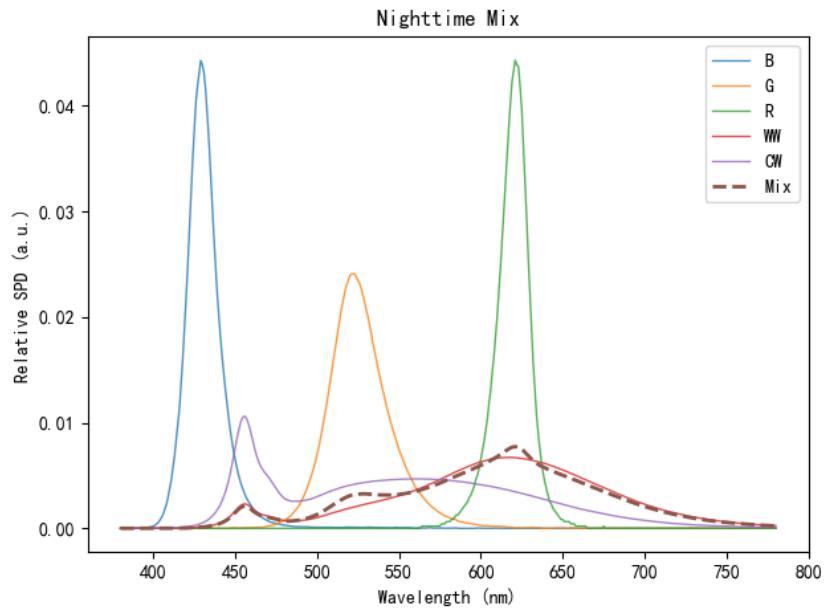


图 7 多通道 LED 混合光源夜间光谱功率分布对比图

表 7 夜间助眠模式的最优权重表

通道	权重
蓝光 B	0.018
绿光 G	0.0637
深红光 R	0.0172
暖白光 WW	0.878
冷白光 CW	0.023

表 8 夜间助眠模式的各指标取值

指标	取值
CCT	2700.0000
D_{uv}	0.0000
R_f	91.1558
R_g	101.6010
mel-DER	0.4155

对比直接 NNLS+SLSQP 与 DE+NNLS+SLSQP，后者在拟合误差、约束满足率、收敛时间、稳定性具有优势，两阶段优化框架 (NNLS+SLSQP) 结合全局优化 (DE) 多起点

初始化增强全局收敛能力。

5.3 问题三的模型的建立和求解

5.3.1 太阳光谱分析

①描述性数据统计分析

根据问题分析，为了实现多通道 LED 光源对太阳光谱的有效模拟，我们需采用加权线性组合的方式来合成总光谱。基于问题二，已知有五个 LED 通道的 SPD（光谱功率分布）数据，将每个通道的光谱与其对应的权重相乘，然后进行加总，从而形成一个新的光谱，具体公式如式 (20)。在此基础上，我们进一步分析对太阳光谱从早晨到傍晚的变化规律进行描述性数据统计分析，包括色温 (CCT) 和光谱成分的动态变化。

处理 380–780 nm 区间以外的数据，并进行异常值剔除后，得到的全天光谱曲线图如图 8 所示，其中将所有时间点曲线叠在一起，便于观察昼夜变化趋势：

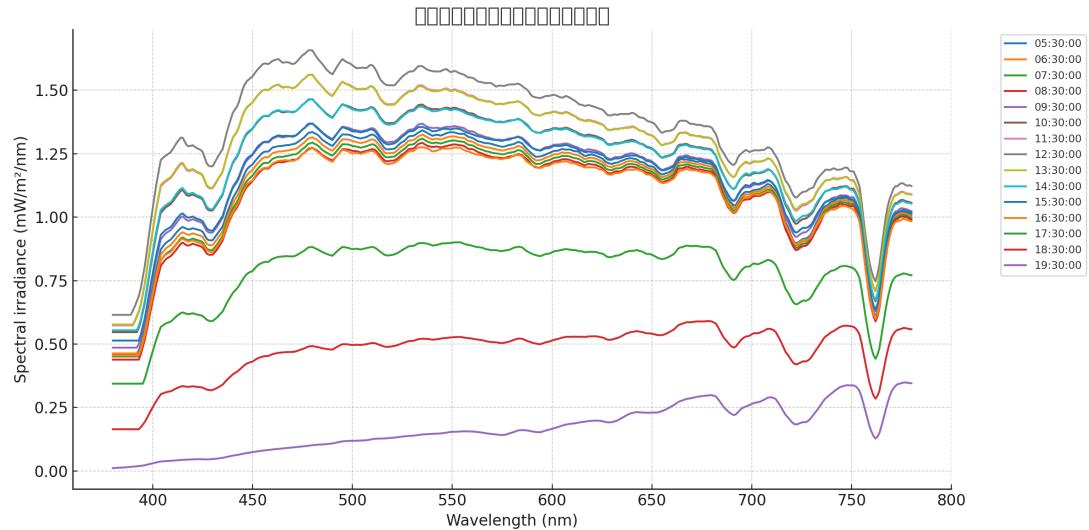


图 8 全天光谱曲线图

该机制基于问题二的优化算法，以合成光谱的 CCT、 R_f （保真度指数）、 R_g （色域指数）和 mel - DER 等参数为优化目标，通过不断调整五个 LED 通道的权重，使得合成光谱尽可能地接近不同时段的太阳光谱。

②代表数据选择

在完成总体描述性数据统计分析后，根据题意，我们需要选取早晨、正午、傍晚的三个代表性时间点。考虑到我们研究的目标是为了模拟出符合大多数人日常活动的太阳光谱情况，以满足不同时段人们对光照的需求，因此尝试根据中国人口分布情况进行代表数据选择。我国人口分布图具体如下：

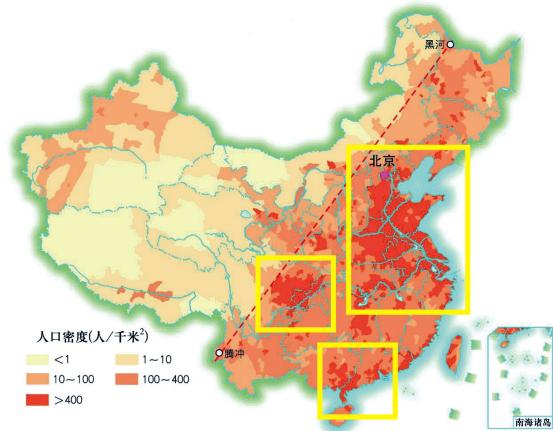


图 9 我国人口分布图

根据图片可知，我国人口分布呈现出显著的不均衡特征。根据文献 [7]，我国东部地区人口密集，占全国人口的 39.93%，中部地区人口占比 25.83%，西部地区人口占比 27.12%，而东北地区人口占比相对较少，为 6.98%。选取黄框内的我国人口高密度地区，为了全面且精准地模拟太阳光谱在一天中的变化情况，我们选择了三个具有代表性的时点进行分析。

第一个时间点为日出后不久，设定为 7 点 30 分。此时太阳刚刚升起，光线相对较弱，且光谱中长波长成分（如红光）占比较高；第二个时间点选取正午时分，即 12 点 30 分。正午时太阳高度角最大，光照强度最强，光谱成分更加均衡，短波长成分（如蓝光）的占比增加；第三个时间点设定为日落前，为 17 点 30 分。随着太阳逐渐西斜，光线强度减弱，光谱再次向长波长方向偏移。

最后，将这三个时间点的太阳光谱数据进行对比（如图 10 所示），可以清晰地看到从日出到日落，太阳光谱在光强、色温以及各波长成分占比上的显著变化。

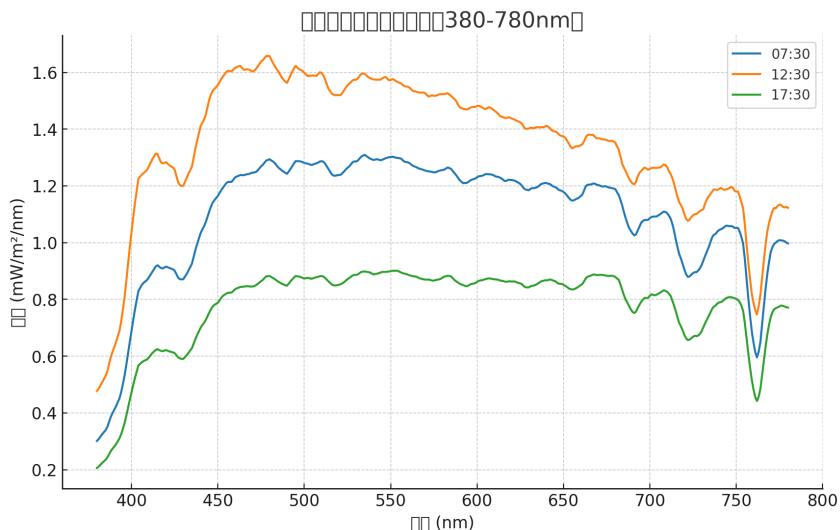


图 10 三个代表时间点的光谱曲线对比图

5.3.2 全天太阳光谱模拟控制模型建立

①目标函数

完成太阳光谱分析后，我们希望针对全天太阳光谱的动态变化特征，通过五通道 LED 光源的权重调控，实现以下目标：

- 每个时间点的合成光谱与对应太阳光谱的拟合误差最小化。
- 权重序列在时间维度上保持平滑连续性，符合 LED 调光的物理约束。
- 关键时间点（7:30、12:30、17:30）的节律指标（mel-DER）与太阳光谱匹配。

因此，我们首先需要进行数据拟合。针对每个时间点 t ，采用带正则项的非负最小二乘，最小化合成光谱与太阳光谱的均方误差（MSE），具体公式如下：

$$\min_{w(t)} \text{Loss}_{\text{spec}}(t) = \|L \cdot w(t) - s(t)\|_2^2 + \lambda \|w(t)\|_2^2 \quad (38)$$

其中， $\|\cdot\|_2^2$ 为 L_2 范数平方， $\lambda > 0$ 用于抑制过度集中在某个通道的解。

②约束条件

在得到目标函数后，我们需要根据题意与数据设置约束条件，使求得的解能够符合要求。考虑到 LED 调光过程的物理连续性，要求权重序列在时间上平滑，因此我们引入时间差分约束，使其尽可能的小，得到如下约束条件：

$$\min_{w(t)} \text{Loss}_{\text{smooth}} = \|w(t + \Delta t) - w(t)\|_2^2 \quad (39)$$

其中， Δt 为时间间隔（如 30 分钟），该约束通过最小化相邻权重的欧式距离，避免调光过程中的跳变。

③优化目标

对于代表数据，为使 7:30, 12:30, 17:30 三个代表性时间点的节律增强，进一步地结合光谱拟合与节律匹配需求，我们进行额外的优化节律指标并构建综合目标，即：

$$\min_{w \geq 0} \text{RMSE}(Lw, s_{\text{target}}) + \alpha \cdot [\text{Mel}(Lw) - \text{Mel}(s_{\text{target}})]^2 \quad (40)$$

其中 α 控制节律指标与光谱误差的权衡（如 $\alpha = 0.3$ 时侧重光谱拟合， $\alpha = 0.5$ 时强化节律匹配）。

5.3.3 模型求解

在问题三中，完成全天太阳光谱模拟控制模型构建后，我们需要在全天不同时间点使五通道 LED 合成光谱尽可能逼近目标太阳光谱。对此我们采取初始解计算（非负最小二乘）、平滑处理（Savitzky-Golay 滤波）和代表性时间节点优化对模型进行求解。

①初始解计算

为了保证后续平滑与局部优化的计算效率和收敛稳定性，首先，我们需要在每个时间点获得一个合理的权重初始解。初始解应尽可能接近全局最优区域，这样可以减少迭代次数，降低优化陷入局部最优的风险。

首先，应确定模型的输入数据，包括五通道 LED 光谱库和目标太阳光谱序列。其中矩阵与问题二构建的光谱矩阵保持一致，即矩阵 M ，其维度为 $m \times 5$ ， m 代表波长采样点数；对于太阳光谱序列，记为 $S(t)$ ，其中 t 为时间变量，每个 $S(t)$ 是长度为 m 的向量，代表对应时间点太阳光谱在全波长下的功率分布。

完成数据输入后，我们的目标是为每个时间点 t ，找到非负权重向量 $\mathbf{w}(t)$ ，使得 LED 合成光谱尽可能逼近目标太阳光谱，即满足公式：

$$M\mathbf{w}(t) \approx S(t) \quad (41)$$

因此，在不考虑时间平滑约束时，该问题可以转化为非负最小二乘问题，同时确定物理约束 $\mathbf{w}(t) \geq 0$ ，即 LED 通道亮度不能为负，权重对应调光比例，需满足非负性。具体公式如下：

$$\min_{\mathbf{w}(t) \geq 0} \|M\mathbf{w}(t) - S(t)\|_2^2 \quad (42)$$

进而，我们对模型进行进一步求初始解。在 python 中，使用 `scipy.optimize` 中的 `nnls` 对每个时间点独立求解，输入五通道 LED 光谱库 M 与目标光谱，即可输出此时间点的权重向量 $\mathbf{w}(t)$ 。得出结果后，由于物理约束要求权重非负且权重对应调光比例，因此需要对初始解求解结果进行非负性和权重总和约束检验。具体步骤如下：

- 非负性检验：验证每个时间点的权重向量 $\mathbf{w}(t)$ 中所有元素是否满足 $w_i(t) \geq 0$ （允许极小数值的浮点误差，如 $w_i(t) \geq -10^{-9}$ ，可通过设置微小容差判断）
- 权重总和约束：权重对应调光比例，需满足 $\sum_{i=1}^5 w_i(t) = 1$ （允许因数值计算导致的微小误差，如误差在 10^{-6} 以内）

最后，我们通过拟合误差 RMSE 计算，对初始解与目标光谱的拟合质量进行量化，即：

$$\text{RMSE}(t) = \sqrt{\frac{1}{m} \sum_{\lambda=1}^m (M\mathbf{w}(t) - S(t))_\lambda^2} \quad (43)$$

其中 m 是波长采样点数， $(M\mathbf{w}(t) - S(t))_\lambda$ 表示合成光谱与目标光谱在第 λ 个波长点的误差。若 RMSE 小于目标光谱最大辐照度的 5%（即 $\text{RMSE}(t) \leq 0.05 \times \max(S(t))$ ），可认为初始解质量较高；大部分时间点满足该条件，则说明非负最小二乘求解的初始解能较好拟合目标太阳光谱。对于 NNLS 求解多时间点 LED 权重的方法，优势在于保证权重非负、独立求解速度快、为后续优化提供合理起点；但因各时间点独立求解，易导致权重变化不连续，因此需进行后续平滑处理。

②Savitzky–Golay 滤波法进行平滑处理

参考文献 [8]，Savitzky–Golay 滤波方法是一种基于局部多项式拟合的平滑方法，可以去除权重曲线的高频噪声，保持趋势连续。对于每个数据点，取其左右一定范围内的点（窗口），用多项式拟合这段曲线，再取拟合值作为平滑后的结果。相比简单的移动平均，SG 滤波在抑制噪声的同时能较好保留数据的峰值、趋势和形状，适合 LED 权重曲线这种需要保形的时序信号。其原理如图 11 所示。



图 11 Savitzky–Golay 滤波法及其原理

首先，对于 Savitzky–Golay 滤波法，需要针对通过非负最小二乘 (NNLS) 求解得到的每个 LED 通道的权重时间序列进行平滑去噪。对此，从权重时间序列集合中，我们单独提取第 i 个通道的权重序列 $w_i = [w_i(1), w_i(2), \dots, w_i(T)]$ （其中 T 为全天采样的时间点总数）设置滤波参数并执行平滑。将平滑处理后的第 i 个通道权重序列 $w_{smoothed}$ 存入新的集合，接着切换至下一个通道 ($i = i + 1$)，重复处理步骤 1 - 3，直至 5 个通道全部处理完毕。

进而，为确保平滑后的权重序列既有效去除噪声，又保留合理的物理特征（如权重变化趋势、节律指标关联特性），可以通过以下指标量化评估，并依据结果调整参数，具体的指标计算公式分别如下：

$$\text{平滑幅度比} = \frac{\sum_{t=1}^T |w_{smooth}(t) - w_{raw}(t)|}{\sum_{t=1}^T |w_{raw}(t)|} \quad (44)$$

$$\text{一阶差分方差} = \text{Var}(w_{smooth}(t+1) - w_{smooth}(t)) \quad (45)$$

$$\text{节律指标变化率} = \frac{|\text{Mel}(w_{smooth}) - \text{Mel}(w_{raw})|}{\text{Mel}(w_{raw})} \times 100\% \quad (46)$$

对于式 (44)，该指标反映平滑操作对原始数据的整体改动程度，初始期望控制在 $10\% - 30\%$ 。对于式 (45)，方差越小，表明权重序列的波动越平缓，平滑效果越好。在式 (46)，需保证该变化率控制在 5% 以内，以确保平滑操作未严重破坏权重序列与生理节律的关联特性。完成 5 个通道权重序列的平滑处理与参数调优后，我们进而得到平滑后的权重时间序列集合，这些序列既去除了高频噪声，又保留了权重随时间变化的趋势和与生理节律相关的关键特征，如图 12 所示。

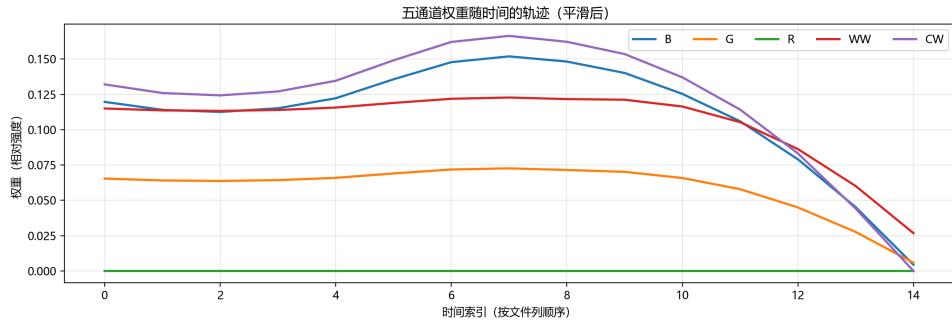


图 12 五通道 LED 权重时序的平滑轨迹图

③代表性时间节点优化

在应用基础上，我们选用差分进化（DE）算法对代表性时间节点进行优化，其核心优势为全局寻优能力突出，可有效避免陷入局部最优解，适配“光谱拟合误差 RMSE + 节律误差”双目标优化需求。类似问题二的步骤，我们选定三个代表性时间节点，记为 t_1, t_2, t_3 ，提取经 Savitzky - Golay 滤波后的对应时间点权重 $\mathbf{w}(t_i)$ 、目标太阳光谱 $S(t_i)$ ，以及节律指标计算所需的光谱数据，构建以“RMSE 节律误差”为核心的目标函数 J ，数学表达式为：

$$J(\mathbf{w}) = \text{RMSE}(\mathbf{w}) + \alpha \cdot \text{RhythmErr}(\mathbf{w}) \quad (47)$$

根据差分进化（DE）在三个代表时间点上进行二次优化，最后得出的可视化结果如下图 13 所示：

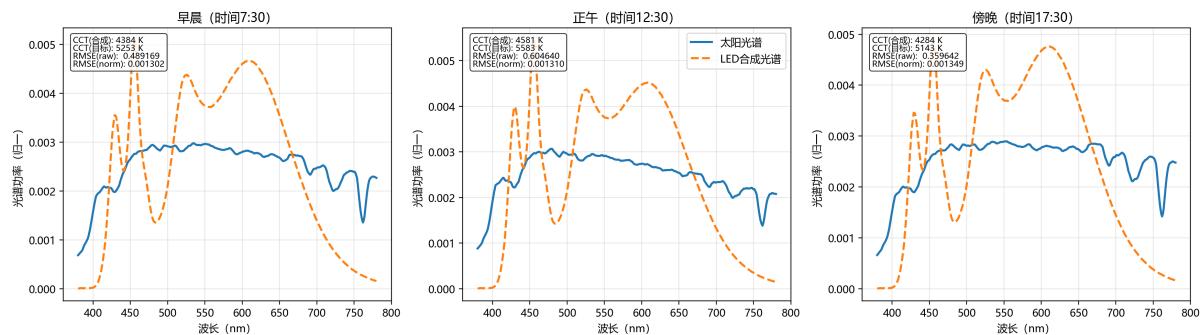


图 13 三个代表时间点太阳光谱与 LED 合成光谱对比图

最终，模型求解结果如下：

表9 LED 通道原始权重随时间分布

时间	B_raw	G_raw	R_raw	WW_raw	CW_raw
时间 5:30	0.120	0.065	0.000	0.115	0.132
时间 6:30	0.114	0.064	0.000	0.114	0.126
时间 7:30	0.113	0.064	0.000	0.113	0.124
时间 8:30	0.115	0.064	0.000	0.114	0.127
时间 9:30	0.122	0.066	0.000	0.116	0.135
时间 10:30	0.136	0.069	0.000	0.119	0.149
时间 11:30	0.148	0.072	0.000	0.122	0.162
时间 12:30	0.152	0.073	0.000	0.123	0.166
时间 13:30	0.148	0.072	0.000	0.122	0.162
时间 14:30	0.140	0.070	0.000	0.121	0.154
时间 15:30	0.125	0.066	0.000	0.116	0.137
时间 16:30	0.106	0.058	0.000	0.105	0.114
时间 17:30	0.079	0.045	0.000	0.086	0.083
时间 18:30	0.045	0.028	0.000	0.060	0.044
时间 19:30	0.004	0.006	0.000	0.027	0.000

5.4 问题四的模型的建立和求解

5.4.1 核心睡眠指标介绍与量化

在评估设计的光照优化模型是否真正对改善人类睡眠质量有显著效果前，我们需要根据原始数据，选取总睡眠时间、睡眠效率、入睡潜伏期、各睡眠阶段占比以及夜间醒来次数作为核心睡眠指标，用于对每一次睡眠记录计算出一系列公认的客观睡眠质量评估指标，进而精准评估夜间光照调控对睡眠质量的影响。

①输入数据来源

本文的数据基于交叉设计实验，11位健康受试者分别在三种睡前光照条件下入睡，每位受试者经历所有条件：

- 条件 A：基于问题二的夜间助眠模式光谱
- 条件 B：普通市售 LED 灯
- 条件 C：严格黑暗环境

并使用便携式睡眠监测仪记录每 30 秒的睡眠阶段（Wake、REM、N1、N2、N3），进而获取 33 条睡眠记录被试者（编号 1 - 11）在三种环境（A/B/C）下的整夜监测，每条记录为时间序列数据 ($x_{i,j}$)。其中， i 为记录序号， j 为被试-环境组合编号 ($1 \leq j \leq 33$)，

$(x_{i,j})$ 则代表第 j 条记录中第 i 个时间点的睡眠阶段编码 (2= 浅睡眠, 3= 深睡眠, 4= 清醒, 5=REM 睡眠, 空值 = 缺失)。

②核心睡眠指标计算公式

基于睡眠医学标准 (AASM 指南) 以及题目信息, 我们定义以下指标:

a. 总睡眠时间 (TST, Total Sleep Time)

TST 指从入睡开始到最终醒来, 整个睡眠过程所持续的时长, 单位为分钟, 即每个环境中, 所有记录值为 1、2、3、5 的时间段的总和, 1、2、3、5 表示不同的睡眠状态。该指标直接反映个体夜间睡眠的绝对时长, 是衡量睡眠是否充足的基础指标, 具体计算公式如下:

$$TST = \sum(N1 + N2 + N3 + REM) \quad (48)$$

b. 睡眠效率 (SE, Sleep Efficiency)

SE 指有效睡眠时间占总卧床时间的比例, 可以体现实际睡眠时间与在床上所花费时间的占比关系, 反映睡眠过程的高效程度及连续性。其计算公式如下:

$$SE = \frac{TST}{TIB} \times 100\% \quad (49)$$

其中, TIB 表示在床上的总时间, 即从关灯到最终醒来的时长, 单位为分钟。

c. 入睡潜伏期 (SOL, Sleep Onset Latency)

SOL 指从个体上床准备睡觉到进入首次睡眠状态所耗费的时间, 即从开始卧床到首次进入深睡眠 (数字 4) 之前的时间段。它衡量入睡的难易程度, 潜伏期越短, 表明入睡越容易。其计算公式如下:

$$SOL = t_{sleep_start} - t_{lights_off} \quad (50)$$

其中, t_{sleep_start} 表示入睡时刻, t_{lights_off} 表示关灯时刻, 单位为分钟。

d. 深睡眠比例 (N3%)

N3% 指深睡眠阶段 (N3 期) 的总时长占总睡眠时间 (TST) 的百分比, 反映睡眠的深度与身体恢复质量。其计算公式如下:

$$N3\% = \frac{t_{N3}}{TST} \times 100\% \quad (51)$$

其中, t_{N3} 通过睡眠监测数据中标记为 N3 阶段的所有时间段累加获得 (单位: 分钟), 与总睡眠时间 (TST) 的计算单位保持一致。

e.REM 睡眠比例 (REM%)

REM% 指快速眼动睡眠阶段 (REM 期) 的总时长占总睡眠时间 (TST) 的百分比, 与大脑认知功能、记忆巩固相关。其计算公式如下:

$$REM\% = \frac{t_{REM}}{TST} \times 100\% \quad (52)$$

f. 夜间醒来次数 (Awakenings)

Awakenings 指睡眠开始后（即首次进入睡眠阶段后），受试者出现“清醒状态（标记为数字 4）”的总次数，反映睡眠过程中被打断的频率。我们直接使用以下计数函数：

$$Awakenings = \text{count}(\text{Wake episodes after sleep onset}) \quad (53)$$

定义核心睡眠指标后，需对数据针对性预处理以适配分析需求。对于宽表，由于其每行对应一位被试，涵盖 subject、实验条件及六个睡眠指标列，不利于多条件、多指标的统一分析，因此需通过 pandas.melt() 转换为长表。

5.4.2 总体效应检验模型建立

为评估三种光照条件 (A: 优化光照, B: 普通 LED, C: 黑暗环境) 对睡眠质量的影响，本研究基于交叉实验数据，对六项睡眠指标进行统计分析：入睡潜伏期 (SOL)、总睡眠时间 (TST)、睡眠效率 (SE)、深睡眠比例 (N3%)、快速眼动睡眠比例 (REM%) 以及夜间觉醒次数 (Awakenings)。统计分析模型的分析过程分为描述性统计、总体效应检验流程。所有统计检验均在 Python (pingouin 库) 中实现，保证方法透明和可复现性。

在开展复杂统计检验前，需通过描述性统计清晰呈现数据的基础特征，为后续分析提供直观参考。对于本实验，我们对每位受试者在三种光照条件下的各项指标计算均值与标准差，作为后续检验的基础。对此，均值和标准差的计算公式分别如下：

$$\bar{x}_j = \frac{1}{n} \sum_{i=1}^n x_{ij} \quad (54)$$

$$s_j = \sqrt{\frac{1}{n-1} \sum_{i=1}^n (x_{ij} - \bar{x}_j)^2} \quad (55)$$

其中， x_{ij} 表示第 i 位受试者在第 j 种光照条件下的指标观测值， n 表示受试者的数量。根据上述公式，求得的可视化结果如下：

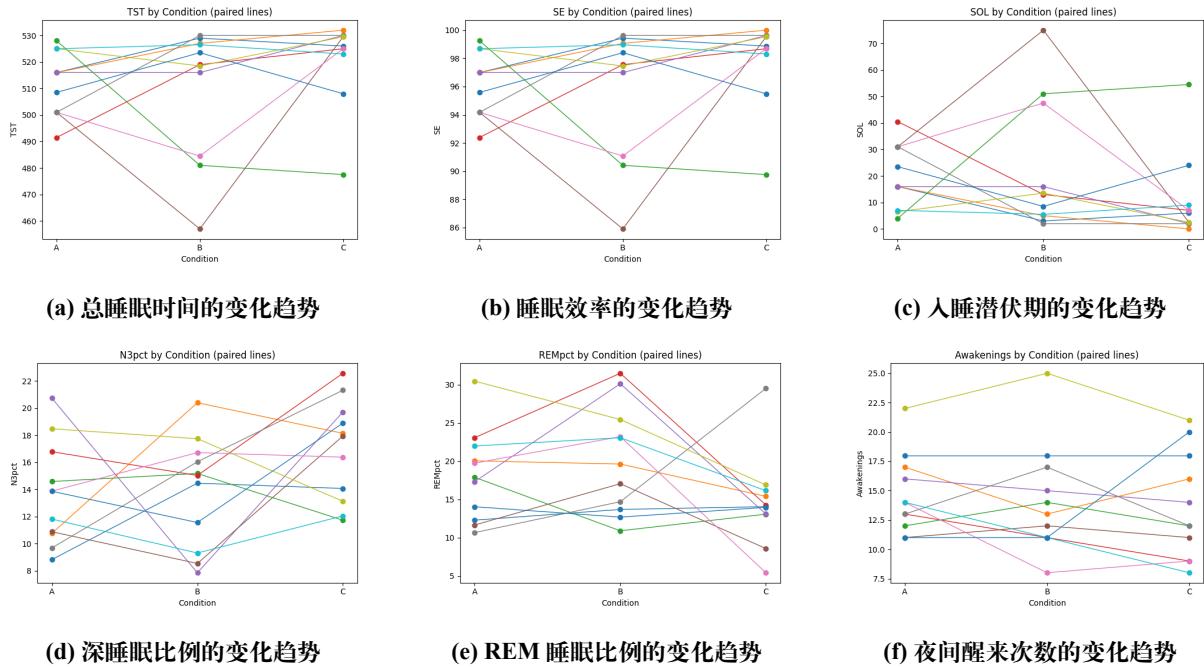


图 14 受试者在三种光照条件下的变化趋势连线图

进一步地，考虑到每位受试者均接受三种光照条件的干预，我们优先采用**重复测量方差分析 (Repeated-Measures ANOVA, RM-ANOVA)** 检验光照条件对睡眠指标的效果。首先，设 y_{ij} 为第 i 位受试者在第 j 种条件下的测量值（如总睡眠时间、睡眠效率等），构建具体模型形式如下：

$$y_{ij} = \mu + \alpha_j + \beta_i + \varepsilon_{ij} \quad (56)$$

其中 μ 为总体均值， α_j 为条件效应， β_i 为受试者效应， ε_{ij} 为误差项。

①球形性假设

由于 M - ANOVA 要求满足球形性假设（即不同条件下差值的方差齐性），因此可通过 Mauchly 检验判断：构造统计量 W 检验方差 - 协方差矩阵是否为球形。若球形性假设不满足 ($p < 0.05$)，则应用 Greenhouse - Geisser 自由度校正，计算校正系数 ε ，具体公式如下：

$$\varepsilon = \frac{1}{(k-1)} \sum_{m=1}^{k-1} \frac{\lambda_m}{\lambda_1} \quad (57)$$

其中 λ_m 为条件协方差矩阵的特征值， k 为条件数（此处 $k = 3$ ），进一步保证检验模型的有效性。

②Friedman 检验

当数据分布显著偏离正态性或方差齐性假设时，改用 Friedman 检验（适用于被试内设计的非参数检验）。Friedman 检验通过对每个受试者的观测值排序，构造统计量 Q ，

其具体公式如下：

$$Q = \frac{12n}{k(k+1)} \sum_{j=1}^k \left(\bar{r}_j - \frac{k+1}{2} \right)^2 \quad (58)$$

其中 \bar{r}_j 为第 j 种条件下观测值的平均秩次， n 为受试者数， k 为条件数。该统计量近似服从自由度为 $k-1$ 的 χ^2 分布，报告统计量 Q 与对应的 p 值，判断不同光照条件下指标是否存在显著差异。据此，可以建立完整的总体效应检验模型，系统检验光照条件对睡眠指标的总体效应，进而为后续模型比较与评估奠定基础。

5.4.3 模型评估

完成总体效应检验模型建立后，若总体效应显著，则可以进行条件间成对比较，具体的步骤设计如下：

- 对符合参数条件的数据，使用配对 t 检验，并计算效应量 Hedges g ：

$$g = \frac{\bar{d}}{s_d} \cdot J \quad (59)$$

其中， \bar{d} 为均值差， s_d 为差值标准差， J 为小样本修正系数。

- 对不符合参数条件的数据，使用 Wilcoxon 符号秩检验，报告统计量 (W 或 Z) 以及校正后的 p 值，并计算效应量秩二分相关 (Rank-Biserial Correlation, RBC) 与通用语言效应量 (Common Language Effect Size, CLES)：

$$RBC = \frac{n_{\text{pos}} - n_{\text{neg}}}{n} \quad (60)$$

其中， n_{pos} 与 n_{neg} 分别表示正秩与负秩的数量， n 为总样本数。

为控制多重比较带来的第一类错误风险，所有成对比较均采用 Bonferroni 校正，并确定 $m = 3$ 为两两比较的次数，即：

$$p_{\text{adj}} = \min(p \times m, 1) \quad (61)$$

结果分析

最终，为评估普通光照 (条件 A)、黑暗环境 (条件 B) 与优化光照 (条件 C) 对睡眠质量的影响，我们对六个睡眠指标进行了描述性统计与重复测量方差分析 (RM-ANOVA)，得到的三种条件下各指标的分布与均值对比如图和图所示。其中，图 15 表示六个睡眠指标在三种光照条件下的均值与标准差：

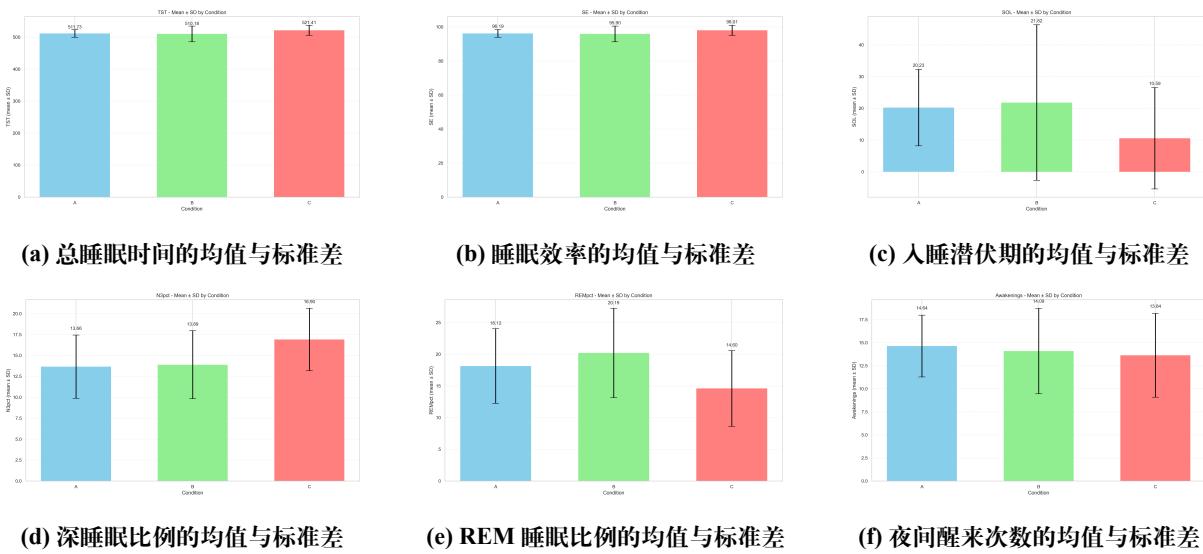


图 15 受试者在三种光照条件下的均值与标准差柱状图

下图 16 表示六个睡眠指标在三种光照条件下的分布：

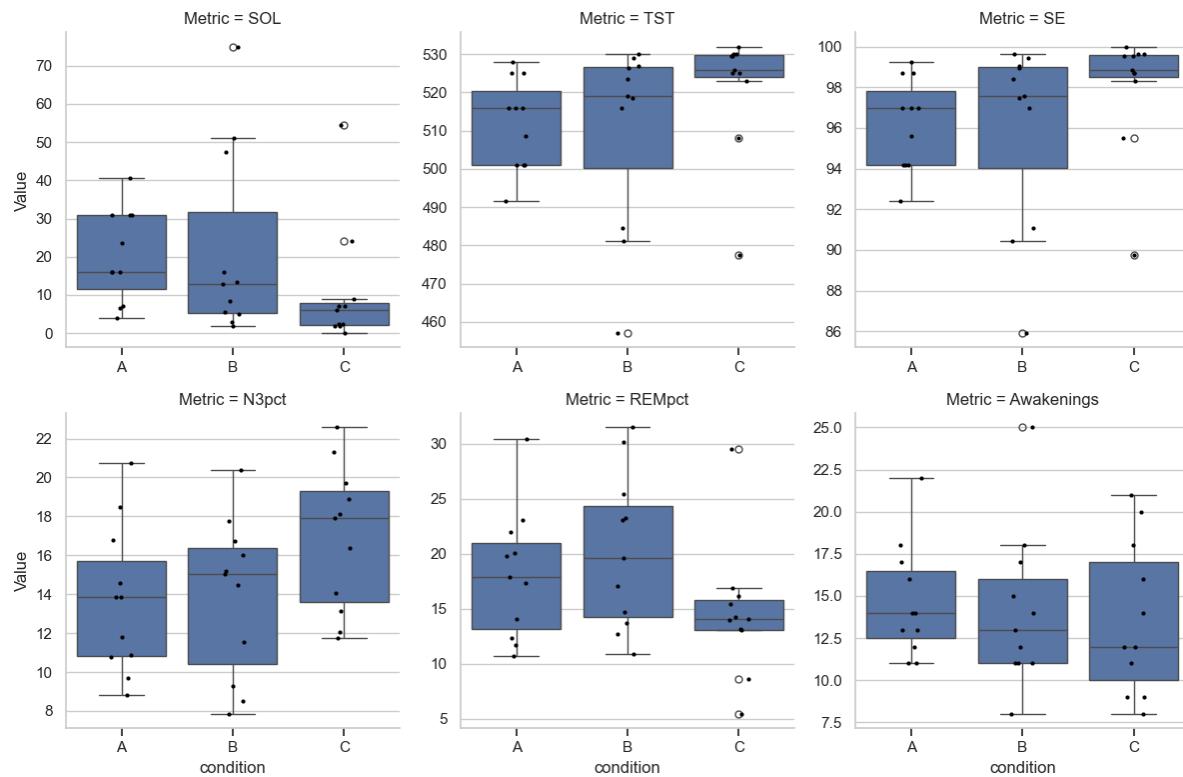


图 16 六个睡眠指标在三种光照条件下的分布箱线图

利用 python 求解，总体效应检验模型的统计检验结果见表 10：

表 10 不同光照条件下睡眠指标的重复测量方差分析及事后比较结果

指标	A (普通光照)	B (黑暗环境)	C (优化光照)	RM-ANOVA p 值	A vs B (p / d)	A vs C (p / d)	B vs C (p / d)
SOL (min)	20.23 ± 12.04	21.82 ± 24.48	10.59 ± 15.96	0.2842	1.0000 / -0.079	0.6032 / 0.656	0.4934 / 0.523
TST (min)	511.73 ± 11.99	510.18 ± 24.48	521.41 ± 15.96	0.2828	1.0000 / 0.077	0.5967 / -0.660	0.4934 / -0.523
SE (%)	96.19 ± 2.25	95.90 ± 4.60	98.01 ± 3.00	0.2828	1.0000 / 0.077	0.5967 / -0.660	0.4934 / -0.523
N3pct (%)	13.66 ± 3.78	13.89 ± 4.06	16.90 ± 3.73	0.1236	1.0000 / -0.057	0.1767 / -0.831	0.3156 / -0.743
REMpct (%)	18.12 ± 5.89	20.19 ± 7.04	14.60 ± 5.97	0.1072	0.7646 / -0.307	0.6566 / 0.572	0.2767 / 0.825
Awakenings	14.64 ± 3.35	14.09 ± 4.64	13.64 ± 4.54	0.6637	1.0000 / 0.130	1.0000 / 0.241	1.0000 / 0.095

RM - ANOVA 检验结果显示，SOL、TST、SE、N3pct、REMpct 及 Awakenings 六项指标在普通光照、黑暗环境与优化光照三种条件下均未达到统计显著性 ($p > 0.05$)。在 SOL 与 N3pct 两项指标中，优化光照条件 (C) 的均值分别低于或高于其他条件，呈现改善趋势 (SOL 缩短，N3pct 增加)，但事后 Bonferroni 检验结果均未显著 ($p > 0.05$)。

六、模型的评价与改进

6.1 模型的优点

- (1) **多场景适配性强：**针对日间照明与夜间助眠两种场景，通过动态调整五通道 LED 权重，实现了光谱特性的精准调控，适配不同场景的照明需求与生理节律要求。
- (2) **优化算法高效稳定：**采用“NNLS 多起点初始化 + SLSQP 局部精修 + DE 全局抛光”的分层优化策略，兼顾了全局搜索能力与局部收敛精度，在保证解的物理可行性的同时，提升了求解效率与稳定性。
- (3) **分阶段深入探讨：**通过逐步提升模型的复杂性，分步骤地纳入各种影响因素，从基础的静态优化框架出发，过渡到动态控制策略，各问题间逻辑递进，整体模型体系完整，提升了模型的实用性和精确度。
- (4) **参数计算标准化：**严格遵循 CIE 标准、ANSI/IES TM-30-20 规范等国际标准，对 CCT、 D_{uv} 、Rf、Rg 及 mel-DER 等核心参数的计算过程进行了标准化建模，确保参数结果的科学性与可比性，为后续优化与验证提供了可靠的量化依据。

6.2 模型的缺点

- (1) **光谱叠加假设简化：**模型假设多通道 LED 合成光谱为各通道光谱的线性叠加，未考虑极端情况下可能存在的微弱非线性光学效应（如多色光干涉）。
- (2) **计算复杂性：**DE 全局抛光步骤虽能提升解的质量，但增加了一定计算量。对于实时性要求极高的场景（如毫秒级动态调光），可通过简化全局搜索步骤或采用预算计算权

重表的方式进一步优化。

- (3) **数据预处理的简化假设**: 由于原始 SPD 数据的信噪比较高, 预处理中对异常值采用线性插值替换, 虽保证了数据连续性, 但未考虑极端异常值可能带来的微小偏差。

6.3 模型的改进推广分析

- (1) **视场适配的改进和推广**: 相比 CIE 1931 2° 标准模型, CIE 1964 10° 标准模型针对大视场 (约 10° 视角) 优化, 适配人眼观察大面积物体或场景时, 视网膜周边区域参与感知的特性, 通过修正光谱响应曲线, 更精准量化大视场下颜色感知, 可用于建筑照明、大幅面印刷品等大场景颜色评价。
- (2) **精度优化的改进和推广**: 该模型基于更大样本量实验数据, 修正大视场下颜色匹配偏差, 尤其在蓝、绿波段光谱响应更契合人眼对大面积颜色感知规律, 减少因视场差异导致的颜色测量误差, 提升大视场场景中颜色量化与实际感知的一致性。
- (3) **应用场景的改进和推广**: 模型将标准扩展至大场景或大面积物体场景, 如医院、农业大棚整体照明色评估等, 填补大视场颜色量化空白, 让不同尺度场景颜色评价更贴合实际视觉体验。

参考文献

- [1] 李亚国. 基于节律健康照明的光谱与照明设计优化[J]. 光源与照明, 2020(4):12-13.
- [2] 张浩, 徐海松. 光源相关色温算法的比较研究[J]. 光学仪器, 2006, 28(1):54-58.
- [3] 喻柏林, 荆其诚. 光源的色温和 CIE 标准光源[J]. 国外计量, 1977(01):40-43.
- [4] ROYER M P. Tutorial: Background and guidance for using the ansi/ies tm - 30 method for evaluating light source color rendition[J/OL]. Leukos, 2022, 18(2):191-231. DOI: 10.1080/15502724.2020.1860771.
- [5] 麻佳琪, 张孟杨, 杨胡江, 等. 基于三基色 LED 的色匹配实验[J]. 物理实验, 2014, 34 (04):42-44, 48.
- [6] 陈逸陶, 蒋艳, 徐颖. 基于逐步惩罚的动态多元线性回归预测模型[J]. 统计与决策, 2008(12):9-10.
- [7] 王桂新, 魏治. 中国人口分布的空间格局及其演变趋势[J]. 中国人口科学, 2021(03): 2-15.
- [8] 杜树新, 杜阳锋, 武晓莉. 基于 Savizky-Golay 多项式的三维荧光光谱的曲面平滑方法 [J]. 光谱学与光谱分析, 2011, 31(02):440-443.

附录 A 文件列表

文件名	功能描述
Problem1.py	问题一参数计算部分代码
Problem2.py	问题二光谱功率分布相关代码
Problem3.py	问题三光谱控制策略相关代码
Problem4.py	问题四总体效应检验相关代码

附录 B 代码

Problem1.py

```
1 # 导包
2 import argparse
3 import os
4 from pathlib import Path
5 import re
6 import numpy as np
7 import pandas as pd
8 from scipy.interpolate import interp1d
9
10 # ===== 可选: TM-30 依赖 (第二问用到)。若未安装, 会在运行时给出友好提示 =====
11 _HAS_COLOUR = True
12 try:
13     import colour
14     from colour.quality import tm3018
15 except Exception:
16     _HAS_COLOUR = False
17
18
19 # 读取和插值
20 def _to_float(x):
21     if pd.isna(x):
22         return None
23     m = re.search(r'^[-+]?[0-9]*\.[0-9]+([eE][-+]?[0-9]+)?$', str(x))
```

```

24     return float(m.group()) if m else None
25
26
27 def _interp(x_src, y_src, kind="linear", fill=0.0):
28     return interp1d(
29         x_src, y_src, kind=kind,
30         bounds_error=False,
31         fill_value=fill if fill is not None else "extrapolate"
32     )
33
34
35 #
36 # 第一问： CCT + Duv
37 #
38
39
40
41
42
43
44
45
46
47
48
49

```

```

=====

# 第一问： CCT + Duv
#
=====

def load_spd_from_excel(excel_path, sheet_spd="Problem 1"):
    df = pd.read_excel(excel_path, sheet_name=sheet_spd,
skiprows=1)
    wl = df.iloc[:, 0].map(_to_float).dropna().astype(float).
to_numpy()
    spd = df.iloc[:, 1].map(_to_float).dropna().astype(float).
to_numpy()
    L = min(len(wl), len(spd))
    wl, spd = wl[:L], spd[:L]
    return wl, spd

def load_cie1931_from_excel(excel_path, sheet_cie="CIE1931"):
    cie = pd.read_excel(excel_path, sheet_name=sheet_cie,
skiprows=1)
    λ = cie.iloc[:, 0].map(_to_float).dropna().astype(float).
to_numpy()

```

```

50     x_bar = cie.iloc[:, 1].map(_to_float).dropna().astype(
51         float).to_numpy()
52     y_bar = cie.iloc[:, 2].map(_to_float).dropna().astype(
53         float).to_numpy()
54     z_bar = cie.iloc[:, 3].map(_to_float).dropna().astype(
55         float).to_numpy()
56
57     L = min(len(λ), len(x_bar), len(y_bar), len(z_bar))
58     return λ[:L], x_bar[:L], y_bar[:L], z_bar[:L]
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80

```

归一化系数: $k = 100 / \sum S(\lambda) \cdot \Delta(\lambda)$

if Y_raw == 0:

return np.nan, np.nan, np.nan

k = 100.0 / Y_raw

return k * X_raw, k * Y_raw, k * Z_raw

def xyz_to_xy(X, Y, Z):

"""XYZ -> xy 色品坐标。"""

den = X + Y + Z

if den == 0 or not np.isfinite(den):

return np.nan, np.nan

return X / den, Y / den

```

81 def mccamy_cct(x, y):
82     """
83     McCamy 经验公式,适用于常见相关色温区间的快速估算。
84     """
85     if not np.isfinite(x) or not np.isfinite(y) or (y -
86         0.1858) == 0:
87         return np.nan
88     n = (x - 0.3320) / (y - 0.1858)
89     return -437.0 * n**3 + 3601.0 * n**2 - 6861.0 * n +
90         5514.31
91
92 def xy_to_uv1960(x, y):
93     """xy -> CIE 1960 UCS (u, v)。"""
94     den = (-2.0 * x + 12.0 * y + 3.0)
95     if den == 0 or not np.isfinite(den):
96         return np.nan, np.nan
97     u = 4.0 * x / den
98     v = 6.0 * y / den
99     return u, v
100
101 def planck_uv_from_cct(cct):
102     """
103     由 CCT 计算普朗克轨迹上的 (u_p, v_p)。
104     """
105     if not np.isfinite(cct) or cct < 1000:
106         cct = 1000.0
107     u_p = (0.860117757 + 1.54118254e-4 * cct + 1.28641212e-7 *
108         cct**2) / \
109             (1 + 8.42420235e-4 * cct + 7.08145163e-7 * cct**2)
110     v_p = (0.317398726 + 4.22806245e-5 * cct + 4.20481691e-8 *
111         cct**2) / \
112             (1 - 2.89741816e-5 * cct + 1.61456053e-7 * cct**2)
113     return u_p, v_p

```

```

112
113
114 def compute_duv_from_xy_cct(x, y, cct):
115     """
116         把 (x,y) 转到 (u,v), 与普朗克点 (u_p, v_p) 的带符号距离作
117         为 Duv。
118     """
119     u, v = xy_to_uv1960(x, y)
120     if not np.isfinite(u) or not np.isfinite(v):
121         return np.nan
122     u_p, v_p = planck_uv_from_cct(cct)
123     du = u - u_p
124     dv = v - v_p
125     duv = np.sqrt(du * du + dv * dv)
126     # 方向: v - v_p 的符号 (v 高于普朗克轨迹为正 -> 偏绿; 低
127     # 于为负 -> 偏品红)
128     return float(np.sign(v - v_p) * duv)

129
130
131
132
133 def q1_cct_duv(excel_path, sheet_spd="Problem 1", sheet_cie="CIE1931"):
134     """
135         第一问: 返回 CCT (K), Duv, 以及中间量 (x,y,X,Y,Z) 。
136
137         """
138         λ_test, spd_test = load_spd_from_excel(excel_path,
139         sheet_spd)
140         λ_cie, x_bar, y_bar, z_bar = load_cie1931_from_excel(
141         excel_path, sheet_cie)

142
143         X, Y, Z = compute_XYZ_from_spd(λ_test, spd_test, λ_cie,
144         x_bar, y_bar, z_bar)
145         x, y = xyz_to_xy(X, Y, Z)
146         cct = mccamy_cct(x, y)
147         duv = compute_duv_from_xy_cct(x, y, cct)
148         return {

```

```

141     "X": X, "Y": Y, "Z": Z,
142     "x": x, "y": y,
143     "CCT": cct, "Duv": duv
144 }
145
146
147 #
=====

148 # 第二问：TM-30-18 的 Rf / Rg
149 #
=====

150 def q2_rf_rg(excel_path, sheet_spd="Problem 1"):
151     """
152     第二问：计算 TM-30 Rf / Rg。
153     依赖 colour-science (内部的 tm3018 实现)。
154     """
155     if not _HAS_COLOUR:
156         raise RuntimeError("未检测到 colour-science 库，请先安装：pip install colour-science")
157
158     # 读取与清洗
159     df = pd.read_excel(excel_path, header=None)
160     wl = df.iloc[:, 0].map(_to_float).dropna().astype(float).
161     to_numpy()
162     val = df.iloc[:, 1].map(_to_float).dropna().astype(float).
163     to_numpy()
164     L = min(len(wl), len(val))
165     wl, val = wl[:L], val[:L]
166
167     # 构造光谱分布对象
168     sd_test = colour.SpectralDistribution(dict(zip(wl, val)))
169
170     # TM-30 计算

```

```

169     spec = tm3018.colour_fidelity_index_ANSIESTM3018(sd_test,
170     additional_data=True)
171     Rf = float(spec.R_f)
172     Rg = float(spec.R_g)
173     return {"Rf": Rf, "Rg": Rg}
174
175 #
176 # 第三问: melanopic DER
177 #
178 def q3_melanopic_der(excel_path, cs_path, d65_csv,
179                     sheet_spd="Problem 1", sheet_cie="CIE1931
180                     "):
181     """
182         第三问: 计算 melanopic DER (相对 D65)。
183         先把 photopic 视与 melanopic 灵敏度插值到各自波长, 再做离
184         散积分。
185     """
186     # 1) 测试光源 SPD
187     spd_test = pd.read_excel(excel_path, sheet_name=sheet_spd,
188     skiprows=1)
189     λ_test = spd_test.iloc[:, 0].map(_to_float).dropna().
190     astype(float).to_numpy()
191     spd_val = spd_test.iloc[:, 1].map(_to_float).dropna().
192     astype(float).to_numpy()
193     L = min(len(λ_test), len(spd_val))
194     λ_test, spd_val = λ_test[:L], spd_val[:L]
195
196     # 2) photopic 视
197     cie = pd.read_excel(excel_path, sheet_name=sheet_cie,
198     skiprows=1)

```

```

193     λ_cie = cie.iloc[:, 0].map(_to_float).dropna().astype(
194         float).to_numpy()
195     y_bar = cie.iloc[:, 2].map(_to_float).dropna().astype(
196         float).to_numpy()
197     fy = _interp(λ_cie, y_bar, fill=None) # 允许外推（与 q3.
198     py 的 'extrapolate' 等价）
199     ybar_test = fy(λ_test)
200
201     # 3) melanopic 灵敏度
202     mel = pd.read_excel(cs_path)
203     λ_mel = mel.iloc[:, 0].map(_to_float).dropna().astype(
204         float).to_numpy()
205     mel_val = mel.iloc[:, 1].astype(str).str.replace(",","").str.replace(" ", "")
206     mel_val = mel_val.map(_to_float).dropna().astype(float).
207     to_numpy()
208     fmel = _interp(λ_mel, mel_val, fill=0.0)
209     melanopic_test = fmel(λ_test)
210
211     # 4) D65 SPD
212     d65 = pd.read_csv(d65_csv)
213     λ_d65 = d65["Wavelength"].map(_to_float).dropna().astype(
214         float).to_numpy()
215     spd_d65 = d65["RelativeSPD"].map(_to_float).dropna().
216     astype(float).to_numpy()
217     ybar_d65 = fy(λ_d65)
218     melanopic_d65 = fmel(λ_d65)
219
220     # 5) 离散积分（步长按 1nm 近似；若波长不规则，因两边都同处
221     # 处理，该比值仍具可比性）
222     Δλ = 1.0
223     E_mel_test = float(np.sum(spd_val * melanopic_test) * Δλ)
224     E_v_test = float(np.sum(spd_val * ybar_test) * Δλ)
225     E_mel_d65 = float(np.sum(spd_d65 * melanopic_d65) * Δλ)
226     E_v_d65 = float(np.sum(spd_d65 * ybar_d65) * Δλ)

```

```

219
220     mel_DER = (E_mel_test / E_v_test) / (E_mel_d65 / E_v_d65)
221     return {"mel_DER": float(mel_DER)}
222
223
224 #
225 # 统一入口: 一次性跑完 5 个指标
226 #
227 def run_all(excel_path="Problem 1.xlsx",
228             cs_path="cs.xlsx",
229             d65_csv="CIE_D65_380_780.csv",
230             sheet_spd="Problem 1",
231             sheet_cie="CIE1931"):
232
233     # 第一问
234     q1 = q1_cct_duv(excel_path, sheet_spd=sheet_spd, sheet_cie=sheet_cie)
235     # 第二问
236     try:
237         q2 = q2_rf_rg(excel_path, sheet_spd=sheet_spd)
238     except Exception as e:
239         # 若暂未装 colour-science, 可以先得到其余 3 项
240         q2 = {"Rf": np.nan, "Rg": np.nan}
241         print("【提示】TM-30 计算失败 (可能未安装 colour-science) : ", e)
242     # 第三问
243     q3 = q3_melanopic_der(excel_path, cs_path, d65_csv,
244                           sheet_spd=sheet_spd, sheet_cie=sheet_cie)
245
246     # 合并为一行结果
247     row = {

```

```

247     **{k: q1[k] for k in ["X", "Y", "Z", "x", "y"]},  

248     "CCT": q1["CCT"],  

249     "Duv": q1["Duv"],  

250     "Rf": q2["Rf"],  

251     "Rg": q2["Rg"],  

252     "mel_DER": q3["mel_DER"],  

253 }
254 df = pd.DataFrame([row])
255 return df  

256  

257  

258 def parse_args():
259     p = argparse.ArgumentParser(description="整合版：CCT/Duv +  

260     Rf/Rg + melanopic DER 一次性输出")
261     p.add_argument("--excel", default="Problem 1.xlsx", help="  

含 SPD 与 CIE1931 的 Excel 文件路径")
262     p.add_argument("--sheet_spd", default="Problem 1", help="  

Excel 中 SPD 的工作表名")
263     p.add_argument("--sheet_cie", default="CIE1931", help="  

Excel 中 CIE1931 的工作表名")
264     p.add_argument("--cs", default="cs.xlsx", help="melanopic  

灵敏度表路径 (xlsx)")
265     p.add_argument("--d65", default="CIE_D65_380_780.csv",  

help="D65 SPD (csv)")
266     p.add_argument("--save_csv", action="store_true", help="是否另存结果至 outputs/metrics_table.csv")
267
268
269 def main():
270     args = parse_args()
271     out = run_all(
272         excel_path=args.excel,
273         cs_path=args.cs,
274         d65_csv=args.d65,

```

```

275     sheet_spd=args.sheet_spd,
276     sheet_cie=args.sheet_cie,
277 )
278
279 # 终端打印
280 row = out.iloc[0]
281 print("\n==== 统一结果 ===")
282 print(f"X={row['X']:.4f}, Y={row['Y']:.4f}, Z={row['Z']:.4f}")
283 print(f"x={row['x']:.5f}, y={row['y']:.5f}")
284 print(f"CCT={row['CCT']:.2f} K")
285 print(f"Duv={row['Duv']:.5f}")
286 print(f"Rf={row['Rf']} if np.isfinite(row['Rf']) else 'NaN'")
287 print(f"Rg={row['Rg']} if np.isfinite(row['Rg']) else 'NaN'")
288 print(f"mel-DER={row['mel_DER']:.6f}")
289
290
291
292 if __name__ == "__main__":
293     main()

```

Problem2.py

```

1 # ===== 导包与基础配置
2
3 import os
4 import warnings
5
6
7 warnings.filterwarnings("ignore", message="Mean of empty slice",
8                         category=RuntimeWarning)
9 warnings.filterwarnings("ignore", message="invalid value
10                        encountered in double_scalars", category=RuntimeWarning)

```

```
9  warnings.filterwarnings("ignore", category=RuntimeWarning,
   module="colour")
10 try:
11     from colour.utilities import ColourUsageWarning
12     warnings.filterwarnings("ignore", category=
ColourUsageWarning)
13 except Exception:
14     pass
15
16 import numpy as np
17 import pandas as pd
18 import matplotlib
19 matplotlib.use("Agg")
20 import matplotlib.pyplot as plt
21
22 from scipy.interpolate import interp1d
23 from scipy.optimize import nnls, minimize,
   differential_evolution
24
25 import colour
26 from colour.quality import tm3018
27
28 # 中文显示
29 plt.rcParams["font.sans-serif"] = ["SimHei"]
30 plt.rcParams["axes.unicode_minus"] = False
31
32
33 # ===== 全局配置
34 =====
34 LED_XLSX      = "Problem 2_LED_SPD.xlsx"
35 FILE_CIE       = "Problem 1.xlsx"
36 SHEET_CIE      = "CIE1931"
37 FILE_MEL       = "cs.xlsx"
38 FILE_D65       = "CIE_D65_380_780.csv"
39
```

```

40 # True: 仅 SLSQP; False: DE + SLSQP
41 FAST_MODE = True
42 # 是否打印通道体检与采样检查
43 PRINT_CHECK = True
44
45 # 题面指标约束
46 SPECS = {
47     "day": {    # 昼间: 高色温, 高显色
48         "CCT_min": 6000.0, "CCT_max": 7000.0,
49         "Duv_abs_max": 0.0060,
50         "Rg_min": 95.0,      "Rg_max": 105.0,
51         "Rf_min": 90.0
52     },
53     "night": { # 夜间: 低色温, 限制 melanopic DER
54         "CCT_min": 2700.0, "CCT_max": 3300.0,
55         "Duv_abs_max": 0.0060,
56         "Rg_min": 95.0,      "Rg_max": 105.0,
57         "Rf_min": 80.0,
58         "melDER_max": 0.50
59     }
60 }
61
62
63 # ===== I/O 与基础工具
64 =====
64 def read_led_spd_from_excel(xlsx_path: str, sheet_name=0) ->
65     Tuple[np.ndarray, np.ndarray]:
66     """读取 5 通道 LED 光谱, 返回: 波长向量 wl、通道矩阵
67     channels (列顺序: B,G,R,WW,CW) """
68     df = pd.read_excel(xlsx_path, sheet_name=sheet_name,
69     header=0)
70     if df.shape[1] < 6:
71         raise ValueError("Excel 需至少 6 列 (波长 + 5 通道: B,
72         G,R,WW,CW) 。")
73     wl = df.iloc[:, 0].to_numpy(float)

```

```

70     ch = df.iloc[:, 1:6].to_numpy(float)
71     ch[ch < 0] = 0.0
72     return wl, ch
73
74
75 def sanitize_wl_channels(wl: np.ndarray, channels: np.ndarray) -> Tuple[np.ndarray, np.ndarray]:
76     """清洗波长与通道数据（去无效行、对齐波长、保证单调）"""
77     df = pd.DataFrame(np.column_stack([wl, channels]), columns=["wl", "B", "G", "R", "WW", "CW"])
78     df = df.replace([np.inf, -np.inf], np.nan).dropna(how="any")
79     df = df.groupby("wl", as_index=False).mean().sort_values("wl")
80     wl_clean = df["wl"].to_numpy(float)
81     ch_clean = df[["B", "G", "R", "WW", "CW"]].to_numpy(float)
82     if wl_clean.size < 2:
83         raise ValueError("清洗后有效波长点 < 2, 请检查输入数据。")
84     return wl_clean, ch_clean
85
86
87 def is_1nm_grid_380_780(wl: np.ndarray) -> bool:
88     """判断是否已是 380 - 780 nm, 1 nm 等间隔网格"""
89     if wl.min() > 380 or wl.max() < 780:
90         return False
91     step = np.diff(wl)
92     return np.allclose(step, 1.0) and abs(wl[0] - 380) <= 1e-6
93     and abs(wl[-1] - 780) <= 1e-6
94
95 def resample_to_1nm(wl: np.ndarray, spd: np.ndarray, lo=380, hi=780) -> Tuple[np.ndarray, np.ndarray]:
96     """重采样到 1 nm 网格（闭区间），线性插值 + 越界置零"""
97     mask = np.isfinite(wl) & np.isfinite(spd)

```

```

98     wl, spd = wl[mask], spd[mask]
99     if wl.size == 0:
100         raise ValueError("resample_to_1nm: 输入为空。")
101     order = np.argsort(wl)
102     wl, spd = wl[order], spd[order]
103     wl_u, idx = np.unique(wl, return_index=True)
104     wl, spd = wl_u, spd[idx]
105
106     lo_eff = max(lo, int(np.ceil(wl.min())))
107     hi_eff = min(hi, int(np.floor(wl.max())))
108     if hi_eff < lo_eff:
109         raise ValueError(f"resample_to_1nm: 范围 [{wl.min():.1f},{wl.max():.1f}] 与 [{lo},{hi}] 无交集。")
110     wl_grid = np.arange(lo_eff, hi_eff + 1, 1.0)
111     spd_grid = np.interp(wl_grid, wl, spd, left=0.0, right=0.0)
112     if not np.any(spd_grid > 0):
113         raise ValueError("resample_to_1nm: 重采样后全为 0。")
114     return wl_grid, spd_grid
115
116
117 def maybe_resample_1nm(wl: np.ndarray, spd: np.ndarray) ->
118     Tuple[np.ndarray, np.ndarray]:
119     """若非 1 nm 网格，则重采样到 380 – 780 nm"""
120     return (wl, spd) if is_1nm_grid_380_780(wl) else
121     resample_to_1nm(wl, spd, 380, 780)
122
123
124 def normalize_spd(spd: np.ndarray, wl: np.ndarray) -> np.
125     ndarray:
126     """按面积归一化 SPD (积分为 1) """
127     area = np.trapz(spd, wl)
128     return spd if area <= 1e-20 else spd / area

```

```

128 def combine_spd(channels: np.ndarray, w: np.ndarray) -> np.
129     ndarray:
130         """通道线性混光"""
131         spd = channels @ w
132         spd[spd < 0] = 0.0
133         return spd
134
135 def project_to_simplex(w: np.ndarray, s: float = 1.0) -> np.
136     ndarray:
137         """投影到概率单纯形 ( $w \geq 0$  且  $\sum(w) = s$ )"""
138         u = np.sort(w)[::-1]
139         cssv = np.cumsum(u)
140         rho = np.nonzero(u * np.arange(1, len(w) + 1) > (cssv - s)
141                           )[0][-1]
142         theta = (cssv[rho] - s) / (rho + 1.0)
143         return np.maximum(w - theta, 0.0)
144
145 # ===== 参考数据与色度计算
146 =====
147 class Refs:
148     """封装 CIE 1931 / melanopic / D65 数据与插值函数"""
149     def __init__(self, cie_path: str, cie_sheet: str, mel_path
150 : str, d65_csv: str):
151         # CIE 1931
152         cie_raw = pd.read_excel(cie_path, sheet_name=cie_sheet
153 , skiprows=1)
154         cie = cie_raw.iloc[:, :4].copy()
155         cie.columns = ["wl", "xbar", "ybar", "zbar"]
156         for c in ["wl", "xbar", "ybar", "zbar"]:
157             cie[c] = pd.to_numeric(cie[c], errors="coerce")
158         cie = cie.dropna(subset=["wl"]).fillna(0.0).
159         sort_values("wl")
160         self._lambda_cie = cie["wl"].to_numpy(float)

```

```

156     self._xbar = cie["xbar"].to_numpy(float)
157     self._ybar = cie["ybar"].to_numpy(float)
158     self._zbar = cie["zbar"].to_numpy(float)
159     self._ix = interp1d(self._λ_cie, self._xbar,
160                         bounds_error=False, fill_value=0.0)
161     self._iy = interp1d(self._λ_cie, self._ybar,
162                         bounds_error=False, fill_value=0.0)
163     self._iz = interp1d(self._λ_cie, self._zbar,
164                         bounds_error=False, fill_value=0.0)
165
166     # melanopic
167     mel_df = pd.read_excel(mel_path).iloc[:, :2].copy()
168     mel_df.columns = ["wl", "mel"]
169     mel_df["wl"] = pd.to_numeric(mel_df["wl"], errors="coerce")
170     mel_df["mel"] = (mel_df["mel"].astype(str)
171                       .str.replace(",", "")
172                       .str.replace(" ", ""))
173     mel_df["mel"] = pd.to_numeric(mel_df["mel"], errors="coerce").fillna(0.0)
174     mel_df = mel_df.dropna(subset=["wl"]).sort_values("wl")
175
176     self._λ_mel = mel_df["wl"].to_numpy(float)
177     self._mel = mel_df["mel"].to_numpy(float)
178     self._im = interp1d(self._λ_mel, self._mel,
179                         bounds_error=False, fill_value=0.0)
180
181     # D65
182     d65 = pd.read_csv(d65_csv).iloc[:, :2].copy()
183     d65.columns = ["wl", "spd"]
184     d65["wl"] = pd.to_numeric(d65["wl"], errors="coerce")
185     d65["spd"] = pd.to_numeric(d65["spd"], errors="coerce")
186     d65.fillna(0.0)
187     d65 = d65.dropna(subset=["wl"]).sort_values("wl")
188     self._λ_d65 = d65["wl"].to_numpy(float)

```

```

183     self._spd_d65 = d65["spd"].to_numpy(float)
184     self._ybar_d65 = self._iy(self._λ_d65)
185     self._mel_d65 = self._im(self._λ_d65)
186
187     # 插值访问器
188     def xbar(self, wl): return self._ix(wl)
189     def ybar(self, wl): return self._iy(wl)
190     def zbar(self, wl): return self._iz(wl)
191     def melanopic_sens(self, wl): return self._im(wl)
192
193     @property
194     def d65_wl(self): return self._λ_d65
195     @property
196     def d65_spd(self): return self._spd_d65
197     @property
198     def d65_ybar(self): return self._ybar_d65
199     @property
200     def d65_mel(self): return self._mel_d65
201
202
203 def compute_XYZ(wl: np.ndarray, spd: np.ndarray, refs: Refs)
204     -> Tuple[float, float, float]:
205         """积分计算三刺激值 XYZ (并按 Y=100 归一)"""
206         xbar, ybar, zbar = refs.xbar(wl), refs.ybar(wl), refs.zbar(wl)
207         X = float(np.sum(spd * xbar))
208         Y = float(np.sum(spd * ybar))
209         Z = float(np.sum(spd * zbar))
210         k = 100.0 / max(Y, 1e-20)
211         return k * X, k * Y, k * Z
212
213 def XYZ_to_xy(X: float, Y: float, Z: float) -> Tuple[float,
214     float]:
215     """XYZ → CIE1931 xy"""

```

```

215     den = X + Y + Z
216     return (np.nan, np.nan) if den <= 1e-20 else (X / den, Y /
den)
217
218
219 def xy_to_1960uv(x: float, y: float) -> Tuple[float, float]:
220     """CIE1931 xy → 1960 uv"""
221     den = -2.0 * x + 12.0 * y + 3.0
222     return (np.nan, np.nan) if abs(den) <= 1e-20 else (4.0 * x
/ den, 6.0 * y / den)
223
224
225 def compute_CCT_Duv_from_spd(wl: np.ndarray, spd: np.ndarray,
refs: Refs) -> Tuple[float, float, float, float]:
226     """基于 1960 uv 与黑体轨迹近似求 CCT 与 Duv (插值 + 二次细
化) """
227     wl1, spd1 = maybe_resample_1nm(wl, spd)
228     X, Y, Z = compute_XYZ(wl1, spd1, refs)
229     x, y = XYZ_to_xy(X, Y, Z)
230     u, v = xy_to_1960uv(x, y)
231     if not np.isfinite(u) or not np.isfinite(v):
232         return np.nan, np.nan, float(u), float(v)
233
234     def uv_p(t):
235         up = (0.860117757 + 1.54118254e-4 * t + 1.28641212e-7
* t**2) / (1 + 8.42420235e-4 * t + 7.08145163e-7 * t**2)
236         vp = (0.317398726 + 4.22806245e-5 * t + 4.20481691e-8
* t**2) / (1 - 2.89741816e-5 * t + 1.61456053e-7 * t**2)
237         return up, vp
238
239     cct_grid = np.arange(1000.0, 25000.0 + 1e-9, 20.0)
240     up, vp = uv_p(cct_grid)
241     dist2 = (u - up) ** 2 + (v - vp) ** 2
242     idx = int(np.argmin(dist2))
243     cct_best = float(cct_grid[idx])

```

```

244
245     i0 = max(1, idx - 1)
246     i1 = min(len(cct_grid) - 2, idx + 1)
247     xs = cct_grid[i0 - 1:i1 + 2] if (i0 - 1 >= 0 and i1 + 1 <
248         len(cct_grid)) else cct_grid[i0:i1 + 1]
249     if xs.size >= 3:
250         us = (0.860117757 + 1.54118254e-4 * xs + 1.28641212e-7
251             * xs**2) / (1 + 8.42420235e-4 * xs + 7.08145163e-7 * xs**2)
252         vs = (0.317398726 + 4.22806245e-5 * xs + 4.20481691e-8
253             * xs**2) / (1 - 2.89741816e-5 * xs + 1.61456053e-7 * xs**2)
254         ys = (u - us) ** 2 + (v - vs) ** 2
255         A = np.vstack([xs**2, xs, np.ones_like(xs)]).T
256         try:
257             a, b, _ = np.linalg.lstsq(A, ys, rcond=None)[0]
258             if a > 0:
259                 cct_refined = -b / (2 * a)
260                 if 1000.0 <= cct_refined <= 25000.0:
261                     cct_best = float(cct_refined)
262             except Exception:
263                 pass
264
265             upb, vpb = uv_p(cct_best)
266             Duv = float(np.hypot(u - upb, v - vpb))
267             Duv *= np.sign(v - vpb) or 1.0
268             return cct_best, Duv, float(u), float(v)
269
270
271
272
273 def compute_TM30_Rf_Rg(wl: np.ndarray, spd: np.ndarray) ->
274     Tuple[float, float]:
275     """TM-30 (Rf, Rg) """
276     wl1, spd1 = maybe_resample_1nm(wl, spd)
277     sd = colour.SpectralDistribution(dict(zip(wl1, spd1)))
278     spec = tm3018.colour_fidelity_index_ANSIESTM3018(sd,
279     additional_data=True)
280     return float(spec.R_f), float(spec.R_g)

```

```

274
275
276 def compute_melanopic_DER(wl: np.ndarray, spd: np.ndarray,
277     refs: Refs) -> float:
277     """melanopic DER (相对 D65 的 melanopic/luminous 比值)"""
278     wl1, spd1 = maybe_resample_1nm(wl, spd)
279     mel = refs.melanopic_sens(wl1)
280     ybar = refs.ybar(wl1)
281     E_mel_t = np.trapz(spd1 * mel, wl1)
282     E_v_t = np.trapz(spd1 * ybar, wl1)
283     E_mel_d = np.trapz(refs.d65_spd * refs.d65_mel, refs.
284         d65_wl)
284     E_v_d = np.trapz(refs.d65_spd * refs.d65_ybar, refs.
285         d65_wl)
285     return np.inf if abs(E_v_t) <= 1e-20 else float((E_mel_t /
286         E_v_t) / (E_mel_d / E_v_d))
287
288 def evaluate_metrics(wl: np.ndarray, spd: np.ndarray, refs:
289     Refs) -> Dict[str, float]:
289     """完整指标集合 (昼/夜抛光使用)"""
290     CCT, Duv, u, v = compute_CCT_Duv_from_spd(wl, spd, refs)
291     Rf, Rg = compute_TM30_Rf_Rg(wl, spd)
292     melDER = compute_melanopic_DER(wl, spd, refs)
293     return dict(CCT=CCT, Duv=Duv, u=u, v=v, Rf=Rf, Rg=Rg,
294         melDER=melDER)
295
296 def evaluate_metrics_night_fast(wl: np.ndarray, spd: np.
297     ndarray, refs: Refs) -> Dict[str, float]:
297     """夜间快速阶段指标 (不算 TM-30, 加速)"""
298     CCT, Duv, u, v = compute_CCT_Duv_from_spd(wl, spd, refs)
299     melDER = compute_melanopic_DER(wl, spd, refs)
300     return dict(CCT=CCT, Duv=Duv, u=u, v=v, melDER=melDER)
301

```

```

302
303 # ===== 约束罚则
304 =====
305
306 def hard_penalty_day(m: Dict[str, float]) -> float:
307     """昼间硬约束罚则: CCT、|Duv|、Rg、Rf 全部合规"""
308     s = SPECS["day"]; pen = 0.0
309     if (not np.isfinite(m["CCT"])) or not (s["CCT_min"] <= m["CCT"] <= s["CCT_max"]):
310         pen += 1e3 + 10.0 * min(abs(m["CCT"] - s["CCT_min"]),
311                                   abs(m["CCT"] - s["CCT_max"]))
312         if (not np.isfinite(m["Duv"])) or abs(m["Duv"]) > s["Duv_abs_max"]:
313             pen += 2e3 + 2e5 * max(0.0, abs(m["Duv"]) - s["Duv_abs_max"])
314         if (not np.isfinite(m["Rg"])) or not (s["Rg_min"] <= m["Rg"] <= s["Rg_max"]):
315             pen += 1e3 + 50.0 * (max(0.0, s["Rg_min"] - m["Rg"]) +
316                                   max(0.0, m["Rg"] - s["Rg_max"]))
317         if (not np.isfinite(m["Rf"])) or m["Rf"] < s["Rf_min"]:
318             pen += 1e3 + 50.0 * max(0.0, s["Rf_min"] - m["Rf"])
319     return pen
320
321
322
323
324
325 def hard_penalty_night_full(m: Dict[str, float]) -> float:
326     """夜间硬约束罚则: CCT、|Duv|、Rg、Rf、melDER"""
327     s = SPECS["night"]; pen = 0.0
328     if (not np.isfinite(m["CCT"])) or not (s["CCT_min"] <= m["CCT"] <= s["CCT_max"]):
329         pen += 1e3 + 10.0 * min(abs(m["CCT"] - s["CCT_min"]),
330                                   abs(m["CCT"] - s["CCT_max"]))
331         if (not np.isfinite(m["Duv"])) or abs(m["Duv"]) > s["Duv_abs_max"]:
332             pen += 2e3 + 2e5 * max(0.0, abs(m["Duv"]) - s["Duv_abs_max"])
333         if (not np.isfinite(m["Rg"])) or not (s["Rg_min"] <= m["Rg"] <= s["Rg_max"]):
334             pen += 1e3 + 50.0 * (max(0.0, s["Rg_min"] - m["Rg"]) +
335                                   max(0.0, m["Rg"] - s["Rg_max"]))
336     return pen

```

```

    " ] <= s[ "Rg_max" ]):
326     pen += 1e3 + 50.0 * (max(0.0, s[ "Rg_min" ] - m[ "Rg" ]) +
327     max(0.0, m[ "Rg" ] - s[ "Rg_max" ]))
328     if (not np.isfinite(m[ "Rf" ])) or m[ "Rf" ] < s[ "Rf_min" ]:
329         pen += 1e3 + 50.0 * max(0.0, s[ "Rf_min" ] - m[ "Rf" ])
330     if (not np.isfinite(m[ "meldER" ])) or m[ "meldER" ] > s[ "meldER_max" ]:
331         pen += 2e3 + 1e4 * max(0.0, m[ "meldER" ] - s[ "meldER_max" ])
332
333
334 # ===== 目标函数
335 =====
335 def loss_day(w: np.ndarray, wl: np.ndarray, channels: np.
336 ndarray, refs: Refs) -> float:
337     """昼间目标: 先硬约束, 再尽量最大化 Rf、轻度压 |Duv|"""
338     if not np.all(np.isfinite(w)): return 1e12
339     w = project_to_simplex(w)
340     spd = normalize_spd(combine_spd(channels, w), wl)
341     m = evaluate_metrics(wl, spd, refs)
342     pen = hard_penalty_day(m)
343     if pen >= 1e3:
344         return pen
345
346
347 def loss_night_fast(w: np.ndarray, wl: np.ndarray, channels:
348 np.ndarray, refs: Refs) -> float:
349     """夜间快速目标: 先把 CCT/Duv/meldER 拉入框, 然后最小化
350     meldER + 轻度压 |Duv|"""
351     if not np.all(np.isfinite(w)): return 1e12
352     w = project_to_simplex(w)
353     spd = normalize_spd(combine_spd(channels, w), wl)
354     m = evaluate_metrics_night_fast(wl, spd, refs)

```

```

353     s = SPECS["night"]
354     pen = 0.0
355     if (not np.isfinite(m["CCT"])) or not (s["CCT_min"] <= m["CCT"] <= s["CCT_max"]):
356         pen += 1e3 + 10.0 * min(abs(m["CCT"] - s["CCT_min"]),
357                                   abs(m["CCT"] - s["CCT_max"]))
358     if (not np.isfinite(m["Duv"])) or abs(m["Duv"]) > s["Duv_abs_max"]:
359         pen += 2e3 + 2e5 * max(0.0, abs(m["Duv"]) - s["Duv_abs_max"])
360
361
362 # ===== 启发式初始化与优化器
363 =====
363 def make_reference_spd(wl: np.ndarray, target_cct: float) ->
364     np.ndarray:
364     """构造简单参考 SPD (高 CCT 偏蓝、低 CCT 偏红)，用于 NNLS
365     初值"""
365     center, width = (460.0, 70.0) if target_cct >= 5000 else
366     (600.0, 90.0)
366     ref = np.exp(-0.5 * ((wl - center) / width) ** 2)
367     return normalize_spd(ref, wl)
368
369
370 def nnls_seed(channels: np.ndarray, ref_spd: np.ndarray) -> np
370 .ndarray:
371     """以 NNLS 结果为初值，并投影到简单形"""
372     w0, _ = nnls(channels, ref_spd)
373     return project_to_simplex(w0)
374
375
376 def run_local_only(loss_fn, wl, channels, refs, seeds: List[np
376 .ndarray]) -> Tuple[np.ndarray, float]:
377     """仅 SLSQP 局部优化 (快) """

```

```

378     dim = channels.shape[1]
379     bounds = [(0.0, 1.0)] * dim
380     cons = ({'type': 'eq', 'fun': lambda w: np.sum(w) - 1.0},)
381     best_w, best_f = None, np.inf
382     for s in seeds:
383         s = project_to_simplex(s)
384         res = minimize(lambda w: loss_fn(w, wl, channels, refs),
385                       x0=s, method="SLSQP", bounds=bounds,
386                       constraints=cons,
387                       options=dict(maxiter=300, ftol=1e-8,
388                       disp=False))
389         if res.fun < best_f:
390             best_w, best_f = project_to_simplex(res.x), float(res.fun)
391     return best_w, best_f
392
393 def run_global_local_optim(loss_fn, wl, channels, refs, seeds:
394     List[np.ndarray], maxiter_de=120) -> Tuple[np.ndarray,
395     float]:
396     """DE 全局 + SLSQP 抛光（稳）"""
397     dim = channels.shape[1]
398     bounds = [(0.0, 1.0)] * dim
399     init_pop = [project_to_simplex(s) for s in seeds]
400     while len(init_pop) < 10:
401         init_pop.append(project_to_simplex(np.random.rand(dim)))
402     init_pop = np.array(init_pop)
403
404     def de_loss(w): return loss_fn(w, wl, channels, refs)
405
406     result_de = differential_evolution(
407         de_loss, bounds=bounds, maxiter=maxiter_de, popsize
408         =10,

```

```

405     init=init_pop, polish=False, tol=1e-6, mutation=(0.5,
406     1.0),
407     recombination=0.7, workers=-1
408 )
409 w_de = project_to_simplex(result_de.x)
410
411 cons = ({'type': 'eq', 'fun': lambda w: np.sum(w) - 1.0},)
412 res_local = minimize(lambda w: loss_fn(w, wl, channels,
413 refs),
414                         x0=w_de, method="SLSQP", bounds=
415 bounds, constraints=cons,
416                         options=dict(maxiter=500, ftol=1e-9,
417 disp=False))
418 return project_to_simplex(res_local.x), float(res_local.
419 fun)

420
421
422 # ===== 体检与输出
423 =====
424
425 def channel_health_report(wl: np.ndarray, channels: np.ndarray
426 , name=("B", "G", "R", "WW", "CW")) -> None:
427     """简单通道体检: >0 点数与 420 – 680 nm 中段最大连续零长度
428     """
429
430     print("\n[CHANNEL HEALTH]")
431     for i, n in enumerate(name):
432         nz = channels[:, i] > 0
433         cover = int(nz.sum())
434         mask_mid = (wl >= 420) & (wl <= 680)
435         mid = nz[mask_mid]
436         zeros = (~mid).astype(int)
437         max_gap = cur = 0
438         for z in zeros:
439             cur = cur + 1 if z == 1 else 0
440             max_gap = max(max_gap, cur)
441     print(f"{n}: >0点数={cover}, 中段最大连续零长度={max_gap}")

```

```

    max_gap} (1nm 步长)")

432
433
434 def plot_channels_and_mix(wl: np.ndarray, channels: np.ndarray
435     , w: np.ndarray, title: str, fname: str) -> None:
436     """绘制通道与合成光谱并保存"""
437     spd_mix = combine_spd(channels, w)
438     plt.figure(figsize=(9, 4))
439     labels = ["B", "G", "R", "WW", "CW"]
440     for i in range(channels.shape[1]):
441         plt.plot(wl, channels[:, i], linewidth=1.0, alpha
442 =0.85, label=labels[i])
443         plt.plot(wl, spd_mix, linewidth=2.0, linestyle="--", label
444 ="Mix")
445     plt.xlabel("波长 (nm)")
446     plt.ylabel("相对光谱功率 (a.u.)")
447     plt.title(title)
448     plt.grid(alpha=0.3)
449     plt.legend(ncol=6, fontsize=9, frameon=False)
450     plt.tight_layout()
451     plt.savefig(fname, dpi=220)
452     plt.close()

453
454
455 def print_solution(tag: str, wl: np.ndarray, channels: np.
456 ndarray, w: np.ndarray, refs: Refs) -> None:
457     """打印权重与关键指标"""
458     spd = normalize_spd(combine_spd(channels, w), wl)
459     m = evaluate_metrics(wl, spd, refs)
460     print(f"\n==== {tag} ====")
461     print("weights (B, G, R, WW, CW):", np.round(w, 4))
462     for k in ["CCT", "Duv", "Rf", "Rg", "melDER"]:
463         v = m.get(k, np.nan)
464         print(f"{k}: {v:.4f}" if np.isfinite(v) else f"{k}:
NaN")

```

```

461
462
463 # ===== 主流程 =====
464 def main() -> None:
465     # ---- 读取与清洗 ----
466     wl, channels = read_led_spd_from_excel(LED_XLSX)
467     wl, channels = sanitize_wl_channels(wl, channels)
468
469     # ---- 重采样到 1 nm 网格（逐通道）----
470     wl_grid, _ = resample_to_1nm(wl, channels[:, 0], 380, 780)
471     channels = np.column_stack([
472         np.interp(wl_grid, wl, channels[:, i], left=0.0, right
473 =0.0)
474         for i in range(channels.shape[1])
475     ])
476     wl = wl_grid
477
478     # ---- 通道面积归一化（便于混光）----
479     for i in range(channels.shape[1]):
480         channels[:, i] = normalize_spd(channels[:, i], wl)
481
482     for p in [FILE_CIE, FILE_MEL, FILE_D65]:
483         if not os.path.exists(p):
484             raise FileNotFoundError(f"缺少参考文件: {p}")
485     refs = Refs(FILE_CIE, SHEET_CIE, FILE_MEL, FILE_D65)
486
487     # ---- 检查信息 ----
488     if PRINT_CHECK:
489         print(f"[CHECK] wl = [{wl.min():.1f}, {wl.max():.1f}],"
490             N = {wl.size})
491         print(f"[CHECK] channels shape = {channels.shape}")
492         area = [np.trapz(channels[:, i], wl) for i in range(5)]
493     ]
494         print(f"[CHECK] per-channel area = {np.round(area, 6)}")
495

```

```

492     channel_health_report(wl, channels)
493     w_demo = np.ones(5) / 5
494     m_demo = evaluate_metrics(wl, normalize_spd(
495         combine_spd(channels, w_demo), wl), refs)
496         print("[CHECK] metrics sample:", {k: float(m_demo[k])
497             for k in ["CCT", "Duv", "Rf", "Rg", "meldER"]})
498
499     # ====== 昼间优化 (严格约束) ======
500     ref_day = make_reference_spd(wl, 6500.0)
501     seeds_day = [
502         nnls_seed(channels, ref_day),
503         project_to_simplex(np.array([0.10, 0.15, 0.10, 0.10,
504             0.55])),
505         project_to_simplex(np.ones(5))
506     ]
507     if FAST_MODE:
508         w_day, _ = run_local_only(loss_day, wl, channels, refs,
509             seeds_day)
510     else:
511         w_day, _ = run_global_local_optim(loss_day, wl,
512             channels, refs, seeds_day, maxiter_de=120)
513
514     print_solution("Daytime (maximize Rf with hard constraints",
515         wl, channels, w_day, refs)
516     plot_channels_and_mix(wl, channels, w_day, "Daytime Mix",
517         "mix_daytime.png")
518
519     # ====== 夜间优化 (快速 + 抛光) ======
520
521     ref_night = make_reference_spd(wl, 3000.0)
522     seeds_night = [
523         nnls_seed(channels, ref_night),
524         project_to_simplex(np.array([0.25, 0.20, 0.03, 0.45,
525             0.07])),
526         project_to_simplex(np.array([0.30, 0.25, 0.02, 0.40,

```

```

0.03]))
518     ]
519     if FAST_MODE:
520         w_night, _ = run_local_only(loss_night_fast, wl,
521         channels, refs, seeds_night)
521     else:
522         w_night, _ = run_global_local_optim(loss_night_fast,
522         wl, channels, refs, seeds_night, maxiter_de=120)
523
524     # 抛光阶段 (完整指标 + 硬约束)
525     def night_polish_loss(w, wl_, ch_, refs_):
526         w = project_to_simplex(w)
527         spd = normalize_spd(combine_spd(ch_, w), wl_)
528         m = evaluate_metrics(wl_, spd, refs_)
529         pen = hard_penalty_night_full(m)
530         if pen >= 1e3:
531             return pen
532         return m["meldER"] + 20.0 * abs(m["Duv"])
533
534     dim = channels.shape[1]
535     bounds = [(0.0, 1.0)] * dim
536     cons = ({"type": "eq", "fun": lambda w: np.sum(w) - 1.0},)
537     res = minimize(lambda w: night_polish_loss(w, wl, channels,
537         refs),
538                     x0=w_night, method="SLSQP", bounds=bounds,
539         constraints=cons,
539         options=dict(maxiter=150, ftol=1e-8, disp=
540 False))
540     w_night = project_to_simplex(res.x)
541
542     print_solution("Nighttime (hard-constrained, polished)",
542     wl, channels, w_night, refs)
543     plot_channels_and_mix(wl, channels, w_night, "Nighttime
543 Mix", "mix_nighttime.png")
544

```

```
545     print("\n[Saved] 图像文件: mix_daytime.png, mix_nighttime.  
png")  
546     print("[Done] 优化完成。")  
547  
548  
549 # ====== 入口 ======  
550 if __name__ == "__main__":  
    np.random.seed(42)  
    main()  
552
```