基于光谱组合优化的光环境模拟与其对睡眠影响的研究

摘要

本文主要内容为光环境的参数计算、光源的优化组合、昼夜节律模拟以及组合光源对人类睡眠影响的分析。

针对第一题，我们根据参考文献建立了五个核心指标的计算公式，通过构建的公式运算，我们得出样本数据的 **CCT 为 4040.11K，Rf=91.79、Rg=106.07、Duv=-0.368147、mel-DER=0.767**。该算法准确有效地描述了光源的物理性质。

针对第二题，我们利用五种独立 LED 通道的光谱分布，建立非线性优化模型，进行光谱组合。在“日间照明”场景下，目标函数为最大化 Rf。通过 SLSQP 算法求解，我们得出最优结果为 **CCT = 5500K，Rf = 92.81、Rg = 102.88，mel-DER = 0.9937**。在“夜间助眠场景”中，目标为 CCT＝3000±500K 并最小化 mel-DER，最终得到 **CCT 为 3000K、Rf = 87.38、Rg = 103.45、mel-DER = 0.5972**，合成光谱显示暖白与红光成分占主导。

针对第三题，我们基于一天 15 个时刻的太阳光谱数据，目标函数为最小化合成光与太阳光的 mel-DER 之差(节律最接近)；同时最小化合成光与太阳光的光谱差异。我们使用 SLSQP 求解出的最优权重进行拟合，结果显示，合成光的 mel-DER 曲线与自然太阳光高度一致（**RMSE=0.085**），能够较好反映一天的光节律变化。此外，光谱形状在多数波段上也与真实太阳光保持较高拟合度。

针对第四题，我们基于 11 名被试者在三种光照环境下的睡眠实验数据，构建六个关键指标。通过可视化发现，优化光照可有效缩短入睡时间并稳定 REM 比例，而普通光照在 TST 上波动较大。进一步地，我们构建了**方差分析，仅 N3%**在三组均值间存在显著差异。事后检验进一步说明，**优化光照下的深睡眠比例显著低于黑暗环境约 5.65 个百分点**，其余指标差异不显著。

关键词：**相关色温 光谱优化 数学规划 方差分析 睡眠质量**

一、 问题重述

LED 光源因其高效节能与光谱可调等特性，已成为健康照明领域的研究热点。光照不仅提供视觉功能，还会通过非视觉通路影响人体的生理节律，如睡眠与情绪。本题以 LED 光谱设计为核心，研究其颜色特性与生物节律效应，共包含四个问题：

问题 1：根据给定 SPD 数据，计算颜色参数（CCT, Duv）、显色参数（Rf, Rg）与节律参数（mel-DER），综合分析光的物理特性。

问题 2：利用五通道 LED 光谱数据，通过加权合成满足日间高保真与夜间低节律干扰的光谱，并优化通道权重。

问题 3：基于自然日光时间序列数据，设计多通道 LED 的控制策略，模拟太阳光谱的全天节律变化。

问题 4：通过临床睡眠实验数据，分析不同光照环境对睡眠质量的影响，并进行统计检验。

二、 问题分析

本文的总体分析流程图如下:

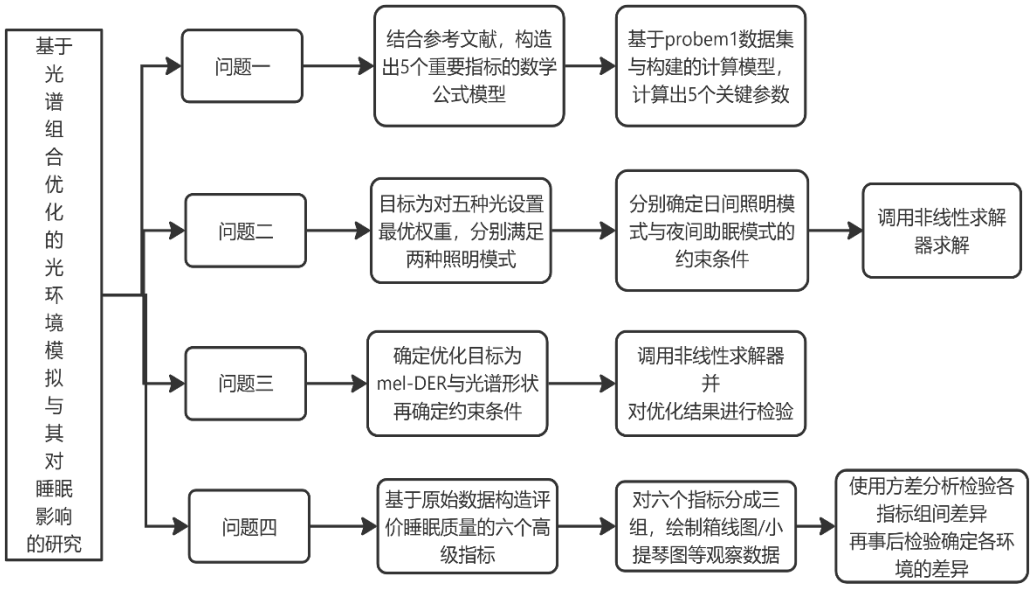

图 1 总体分析流程图

2.1 问题一的分析

问题一要求根据国际标准建立五个关键光学参数的数学模型。首先需要深入研究参考文献，明确 CCT、Duv、Rf、Rg 和 mel-DER 的计算原理和公式，构建完整的计算框架。然后基于提供的 problem1 数据集中的 SPD 数据，通过编程实现这些数学模型，准确计算出每个光源样本的五个参数值。

2.2 问题二的分析

问题二需要利用五个 LED 通道的光谱数据，通过权重优化合成满足特定需求的光谱。针对日间和夜间两种照明模式，分别确定约束条件：日间模式要求 CCT 在 6000±500K 范围内，Rf 尽可能高且 Rg 在 95-105 之间；夜间模式要求 CCT 在 3000±500K 范围内，mel-DER 尽可能低且 Rf 不低于 80。建立优化模型后，需要调用非线性求解器寻找各通道的最佳权重组合。

2.3 问题三的分析

问题三要求模拟自然日光全天的节律变化。首先需要确定优化目标，既要考虑 mel-DER 值的匹配度，也要关注光谱形状的相似性。同时需要设定适当的约束条件以保证合成光谱的实用性。通过非线性求解器进行优化求解后，需要对优化结果进行检验，选取早晨、正午、傍晚三个代表性时间点，对比合成光谱与目标太阳光谱的匹配程度。

2.4 问题四的分析

问题四需要评估不同光照环境对睡眠质量的影响。首先基于原始睡眠分期数据，构造六个评价睡眠质量的高级指标：总睡眠时间、睡眠效率、入睡潜伏期、深睡眠比例、REM 睡眠比例和夜间醒来次数。然后将数据按三种光照环境分成三组，通过箱线图、小提琴图等可视化方法观察数据分布特征。最后使用方差分析检验各指标在组间是否存在显著差异，若结果显著则进一步进行事后检验，确定具体哪些环境之间存在差异。

三、 模型假设

1.线性叠加假设（光谱可加性）:假设合成光的光谱为各 LED 通道在每个波长处“按权重线性相加”的结果，且权重不随波长而变。

2.标准视觉模型与指标有效性假设:假设采用的 CIE 视觉函数与颜色评价指标在本

实验装置与测量条件下成立

3.理想光源假设：假设每个 LED 通道的光谱功率分布（SPD）是稳定且恒定的

4.假设太阳光谱在给定的时间间隔内（例如 8:30 到 9:00）是相对稳定的

5.假设三种光照条件的比较不受顺序等其他变量的影响，且测试者在不同条件下的睡眠表现可直接比较

四、 符号说明

符号	符号含义
L	长波视锥细胞的光谱灵敏度
M	中波视锥细胞的光谱灵敏度
S	短波视锥细胞的光谱灵敏度
X, Y, Z	三刺激值，由 *LMS* 线性变换得到的色度学基础量
x, y, z	归一化色品坐标，由 *XYZ* 转换得到
u, v	CIE 1960 均匀色品坐标，用于计算 Duv
$f(t)$	Lab*色度空间中的非线性转换函数
w_r,w_g,w_b	红光、绿光、蓝光 LED 的权重因子
w_{ww},w_{cw}	暖白光、冷白光 LED 的权重因子
$I(\lambda)$	光强分布，某波长 λ 下的光谱辐射强度
$V(\lambda)$	CIE 明视觉函数
Smel(λ)	Melanopic 光视效能函数

五、 模型建立与求解

5.1 对核心参数建立计算模型

本问题需要计算的核心参数有五个：相关色温(CCT)，距离普朗克轨迹的距离(Duv)，保真度指数(Rf)，色域指数(Rg)，褪黑素日光照度比。其中，相关色温与距离普朗克轨迹的距离共同描述了颜色的外观（偏向于哪个颜色）。保真度指数与色域指数二者评估的是光源与标准光源相比，还原物体真实色彩的能力。褪黑素日光照度比用于量化光照对

人体生理节律的影响强度。

5.1.1 计算相对色温

我们目前拥有数据集：各个波长对应的光强，各个波长对应的 L-cone-opic，M-cone-opic，S-cone-opic（数据来源 CIES026/E）。

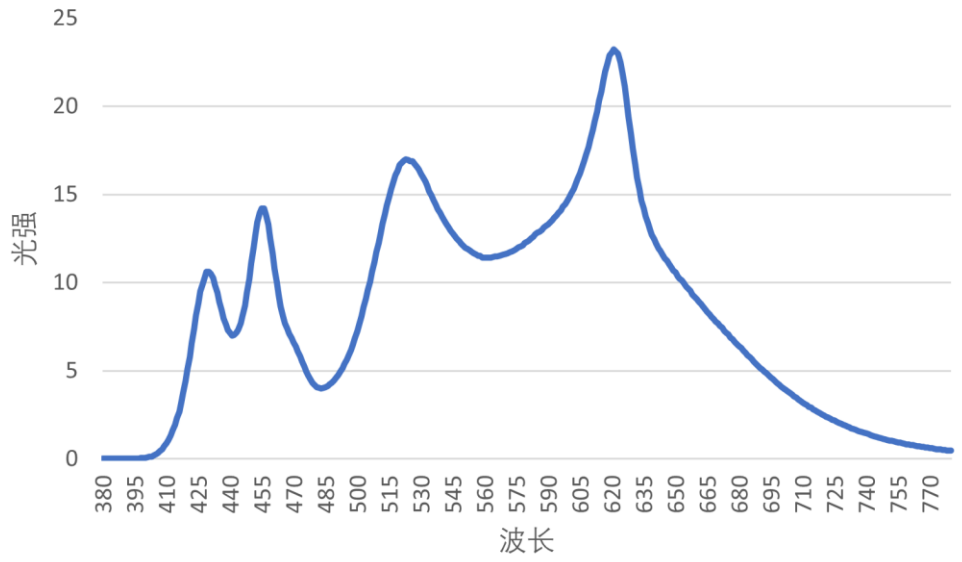

图 2 问题一数据集光强-波长图

我们对矩阵 *LMS* 做线性变换得到矩阵 *xyz*：

$$\begin{bmatrix} \overline{x}(\lambda) \\ \overline{y}(\lambda) \\ \overline{z}(\lambda) \end{bmatrix} = \mathbf{M} \begin{bmatrix} L(\lambda) \\ M(\lambda) \\ S(\lambda) \end{bmatrix} \tag{1}$$

其中，M 为：

$$\mathbf{M} = \begin{bmatrix} 1.94735469 & -1.41445123 & 0.36476327 \\ 0.68990272 & 0.34832189 & 0 \\ 0 & 0 & 1.93485343 \end{bmatrix}$$

对于计算相关色温(CCT),首先要计算色品坐标，计算公式如下：

$$\begin{aligned} X &= k\int_{380}^{780} S(\lambda)\,\overline{x}(\lambda)\,d\lambda \\ Y &= k\int_{380}^{780} S(\lambda)\,\overline{y}(\lambda)\,d\lambda \\ Z &= k\int_{380}^{780} S(\lambda)\,\overline{z}(\lambda)\,d\lambda \end{aligned} \tag{2}$$

其中，k 为归一化系数。

我们得出的 X,Y,Z 为三刺激值，再对其进行线性变换得出色品坐标：

$$x = \frac{X}{X+Y+Z}$$
$$y = \frac{Y}{X+Y+Z} \tag{3}$$

根据参考文献的里 McCamy 近似公式法，我们定义 n：

$$n = \frac{x-0.3320}{y-0.1858} \tag{4}$$

CCT 公式为：

$$CCT = -437n^3 + 3601n^2 - 6818n + 5514.31 \tag{5}$$

基于以上原理，我们在 python 中编写程序，输入为 problem1 中的各个波长对应的光强，CIES026/E 中的 L-cone-opic，M-cone-opic，S-cone-opic 。进行 M 矩阵线性变换后，我们得出了 $\overline{x}(\lambda)$，$\overline{y}(\lambda)$，$\overline{z}(\lambda)$。下面进行积分求出三刺激值。该定积分等价于离散求和，即：

$$\begin{aligned} X &= k\int_{380}^{780} S(\lambda)\,\overline{x}(\lambda)\,d\lambda = k\sum_{\lambda=380}^{780} S(\lambda)\,\overline{x}(\lambda) \\ Y &= k\int_{380}^{780} S(\lambda)\,\overline{y}(\lambda)\,d\lambda = k\sum_{\lambda=380}^{780} S(\lambda)\,\overline{y}(\lambda) \\ Z &= k\int_{380}^{780} S(\lambda)\,\overline{z}(\lambda)\,d\lambda = k\sum_{\lambda=380}^{780} S(\lambda)\,\overline{z}(\lambda) \end{aligned} \tag{6}$$

由三刺激值得出色品坐标后，我们利用 McCamy 近似公式法，得出了 problem1 数据集的**相对色温为：4040.11 K。**

5.1.2 计算距离普朗克轨迹的距离(Duv)

Duv 的计算是在 CIE 1960 (*u, v*) 均匀色品图上进行的。故我们要将 *x,y,z* 变换到(*u,v*)坐标轴中。变换公式如下：

$$\begin{aligned} u &= \frac{4x}{-2x+12y+3} = \frac{4X}{X+15Y+3Z} \\ v &= \frac{6y}{-2x+12y+3} = \frac{6Y}{X+15Y+3Z} \end{aligned} \tag{7}$$

得出(*u,v*)坐标后，进行如下距离公式计算：

$$D_{uv}=\sqrt{(u_s-u_t)^2+(v_s-v_t)^2}\times sign(v_s-v_t) \tag{8}$$

其中，$sign(v_s-v_t)$为符号函数。即(v_s-v_t)大于 0 时为 1，小于 0 时为-1.

根据以上流程，在 python 中编写程序，我们得出此时的 *Duv*=-0.368147。

5.1.3 计算 Rf 与 Rg

Rg 用于衡量测试光源（或显示设备）的色域面积相对于参考标准色域的覆盖比例，反映颜色的丰富度。

Rf 用于评价测试光源还原物体颜色的准确性（与参考光源相比），取值范围通常为 0~100，数值越高表示色保真度越好（100 表示完全还原）。其计算基于 CIE 224:2017 标准，替代传统 CRI 以更精准反映视觉感知。

转换为 CIE 1976 Lab* 坐标:

$$\begin{cases} L_k^*=116f\left(\dfrac{Y_k}{Y_n}\right)-16 \\ a_k^*=500\left[f\left(\dfrac{X_k}{X_n}\right)-f\left(\dfrac{Y_k}{Y_n}\right)\right] \\ b_k^*=200\left[f\left(\dfrac{Y_k}{Y_n}\right)-f\left(\dfrac{Z_k}{Z_n}\right)\right] \end{cases} \tag{9}$$

其中，X，Y，Z 为三刺激值。$f(t)$为非线性转换函数：

$$f(t)=\begin{cases} t^{1/3} & \text{if } t>(\epsilon)^3 \\ \dfrac{t}{3\epsilon^2}+\dfrac{4}{29} & \text{otherwise} \end{cases},\quad \epsilon=\frac{216}{24389}\approx 0.008856 \tag{10}$$

对每个色样 *k*，计算参考光源与测试光源下的 CIEDE2000 色差值：

$$\Delta E_{00,k}=\sqrt{\left(\frac{\Delta L'}{k_L S_L}\right)^2+\left(\frac{\Delta C'}{k_C S_C}\right)^2+\left(\frac{\Delta H'}{k_H S_H}\right)^2+R_T\left(\frac{\Delta C'}{k_C S_C}\right)\left(\frac{\Delta H'}{k_H S_H}\right)} \tag{11}$$

Rf 即为：

$$R_f=100-4.6\,\overline{\Delta}E_{00} \tag{12}$$

由上述算法求得，*Rf* = 91.79 ； *R*g = 106.07.

5.1.4 计算褪黑素日光照度比(mel-DER)

褪黑素日光照度比是描述昼夜光照强度差异与褪黑素分泌节律关联性的量化指标，其核心是通过日间与夜间光照度的比值，反映光环境对褪黑素分泌的调控效应。在自然环境中，该比值通常较高（日间强光、夜间暗光），可维持褪黑素的强昼夜波动（夜间

峰值与日间低谷差异显著），从而保障睡眠 - 觉醒周期等生理节律的正常运行。而人工光环境下（如夜间光污染、日间室内光照不足），该比值可能降低，导致褪黑素分泌节律紊乱（如峰值延迟、振幅减小），进而与失眠、代谢异常等健康问题相关。

褪黑素日光照度比的定义公式如下：

$$\text{mel}-\text{DER}=\frac{\int_{380}^{730} SPD(\lambda)\cdot s_{\text{mel}}(\lambda)\,d\lambda}{\int_{380}^{730} SPD(\lambda)\cdot V(\lambda)\,d\lambda}\times 1.218 \tag{13}$$

其中，$SPD(\lambda)$为所在波长的光强，$s_{\text{mel}}(\lambda)$为 melanopic 光视效能函数，数据可由 CIES026/E 获得。$V(\lambda)$为 CIE 明视觉函数。积分区间为 380 到 730。根据以上原理，我们建立数学模型，求出 mel-DER = 0.767。

5.1.5 结果汇总

表 1 结果汇总

指标	参数
CCT	4040.11 K
Rf	91.79
Rg	106.07
Duv	-0.368147
mel-DER	0.767

5.2 问题二的模型建立与求解

本题要求我们利用五个独立 LED 通道：深红光、绿光、蓝光，暖白光，冷白光，通过建立合适的线性组合，合成一个新的光源。在新的光源满足一定的约束条件的同时，达到最优目标。我们的数据有五个独立 LED 通道的各个波长下的光强。对五个光强的线性组合可以得到新的光强。并且求出最优组合下，CCT，Rg，Rf，Duv，mel-DER 等参数的取值。五个 LED 的光强对波长变化曲线如图所示：

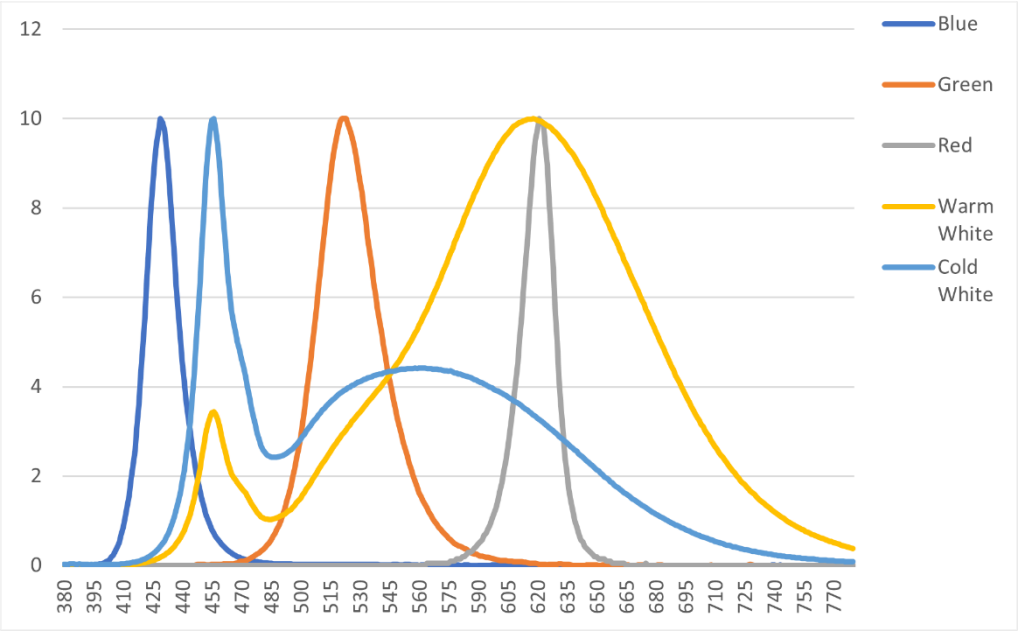

图 3 五种光的光谱分布

图中五种光源（蓝光、绿光、红光、暖白光、冷白光）的光谱分布差异显著：蓝光在 425、440 nm 形成双峰，绿光集中于 515 nm 单峰，红光峰值约 620 nm；暖白光以 590–605 nm **黄红光** 为主（残余蓝光少，适配夜间照明），冷白光呈现 450 nm **蓝光与** 550 nm **绿光** 双峰（高蓝光成分增强日间提神效果），单色光（蓝、绿、红）则针对显示、光疗等特定场景设计。

5.2.1 场景一的建模与求解

我们设 w_r, w_g, w_b, w_{ww}, w_{cw} 分别为对各个光强权重。即合成光的光强为：

$$\begin{aligned}\mathrm{SPD}_{\mathrm{mix}}(\lambda) = w_r \cdot \mathrm{SPD}_{\mathrm{red}}(\lambda) + w_g \cdot \mathrm{SPD}_{\mathrm{green}}(\lambda) + w_b \cdot \mathrm{SPD}_{\mathrm{blue}}(\lambda) \\ + w_{ww} \cdot \mathrm{SPD}_{\mathrm{ww}}(\lambda) + w_{cw} \cdot \mathrm{SPD}_{cw}(\lambda)\end{aligned} \tag{14}$$

下面我们将的到各个波长对应的合成光强。我们的约束指标如下：

我们需要合成光谱的 CCT 在正午日光范围，即 5500 到 6500K 之间：

$$5500\ K \le CCT \le 6500\ K$$

色域指数 Rg 在 95~105 之间：

$$95 \le Rg \le 105$$

Rg 值在 95%~105% 区间中时，表明该光源的颜色覆盖范围与标准光源接近。这一范围能保证颜色丰富度接近自然或标准状态，实现对物体颜色的自然还原与均衡呈现，兼顾色彩丰富度与真实性。

保证颜色自然：

$$Rf > 88$$

在该光照下，物体的颜色外观会非常接近理想自然光下的样子，色彩看起来真实，自然，不会有明显的失真或色偏。同时可保障视觉舒适度。

我们的目标函数时使得合成光谱的保真度指数 (Rf) 尽可能高：

$$max\ Rf$$

将合成光谱的保真度指数（*Rf*）设定为尽可能高的优化目标，其物理意义在于追求合成光源对物体颜色的还原能力非常接近于理想参考光源。最大程度模拟正午的日光效果。

综合以上，我们的数学规划模型为：

$$max\ Rf(w_r, w_g \ldots w_{cw})$$

$$s.t. \begin{cases} 5500 \leqslant CCT \leqslant 6500 \\ 95 \leqslant Rg \leqslant 105 \\ Rf \geqslant 88 \\ \sum_{i=1}^{5} w_i = 1 \end{cases} \tag{15}$$

该数学规划问题为典型的非线性规划。我们采用序列二次规划（SLSQP）求解带约束的非线性优化问题，平衡精度与效率。各参数的计算模型均利用上述模型。在 python 中求解，我们得出的最优组合结果的光照效果参数为：

表 2 场景一最优参数

指标	参数
CCT	5500 K
Rf	92.81
Rg	102.88
mel-DER	0.993667

可以看出，此时我们的 *CCT* 达到正午要求，为 5500K;*Rf* 大于 88，达到 92.8，效果较为理想；*Rg* 在给定范围内。可知该模型顺利实现了在正午环境下，满足合成光的约束。

此时的**最优权重组合为：[0.152616, 0.157249, 0.245801, 0.001581, 0.442753]**

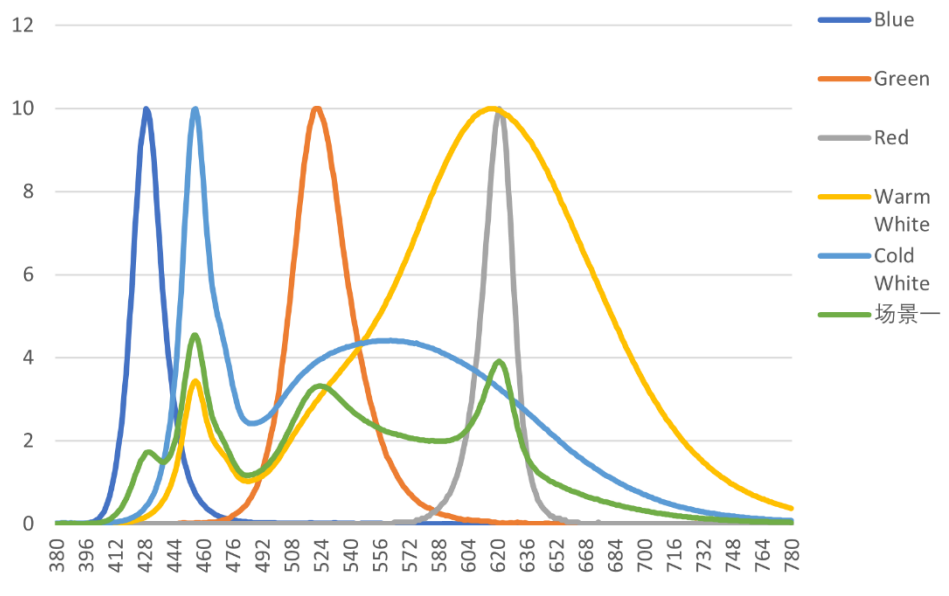

图 4 场景一最优化光谱曲线

绿色曲线为此时场景一的最优化光谱曲线。从图中可以看出生成曲线的波峰与大多数的曲线都重合，曲线的准确度较高，实现的效果也较好。

5.2.2 场景二的建模与求解：

此时，我们需要营造温馨的低色温环境，光谱的 CCT=3000±500 K；而且需要使得视黑素日光效率比尽可能的低。保真度指数(*Rf*)不低于 80。

我们的模型建立如下：

目标函数：最小化视黑素日光效率比

$$min\ \text{mel-DER}(w_r,w_g\ldots w_{cw})$$

$$s.t.\begin{cases}2500 \leqslant CCT \leqslant 3500\\ Rf \geqslant 80\\ \sum_{i=1}^{5} w_i = 1\end{cases} \tag{16}$$

该数学规划问题为典型的非线性规划。我们采用序列二次规划（SLSQP）求解带约束的非线性优化问题，平衡精度与效率。各参数的计算模型均利用上述模型。在 python 中求解，我们得出的参数为：

表 3 场景二最优参数

指标	参数
CCT	300.84 K
Rf	87.38
Rg	103.45
mel-DER	0.5972

权重为：[0.00043,0.00043,0.3043,0.3913,0.3043].

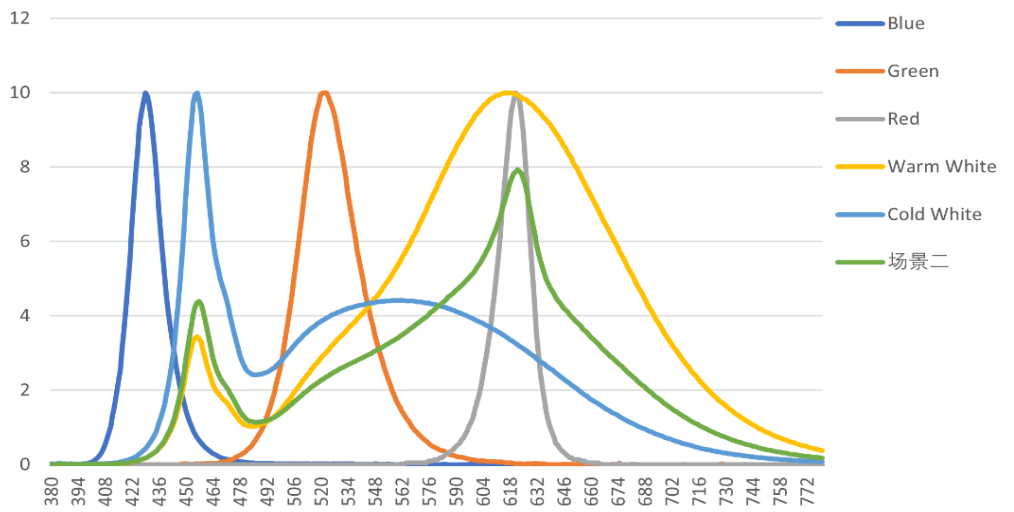

图 5 场景二最优化光谱曲线

可以看出，合成光谱与 Warm White 与 Red，Cold Warm 的相关性较高，他们对应的权重也更高。

5. 3 问题三的模型建立与求解

第三题给出了一天中，多个时刻的太阳光谱数据，我们要用题二的五种 LED 光谱，通过加权，合成新的光谱，来模拟一天中的太阳光谱数据。由题可知，参数褪黑素日光照度比(mel-DER)用于量化光照对人体节律的影响，故该问题的目标函数为最小化各个时刻的太阳 mel-DER 与合成光 mel-DER 之差。使得我们的合成光与真实的太阳光有着相似的节律效果。

我们先利用问题一的 mel-DER 求解函数来计算出 15 个时刻的 mel-DER 值。绘制出时序图观察变化趋势。

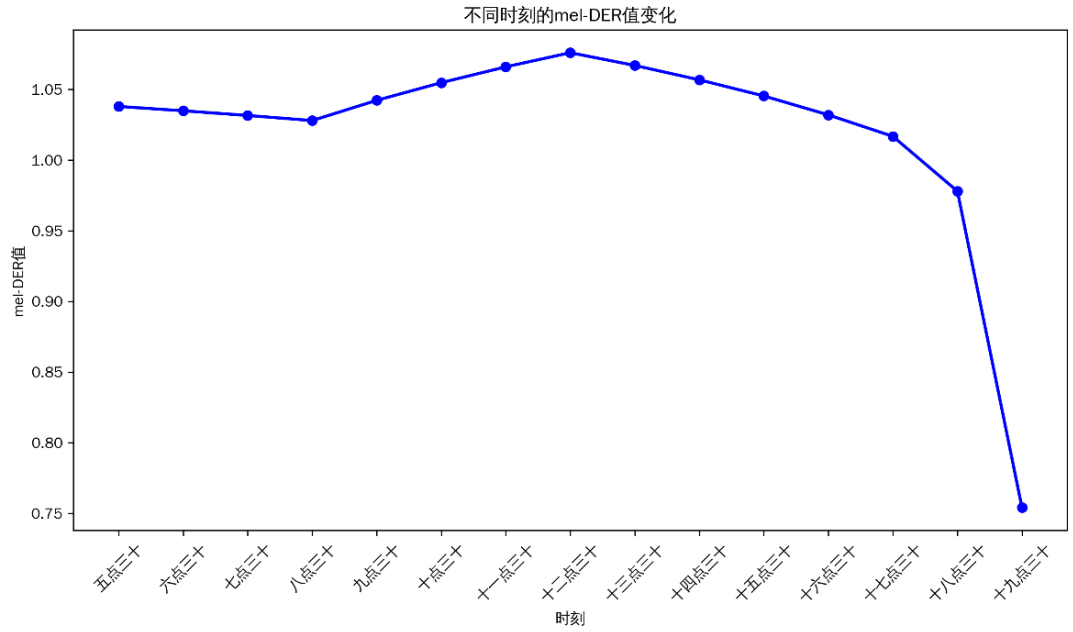

图 6 mel-DER 变化趋势图

可以看出，mel-DER 值**从早上到正午处于上升趋势**，在 12:30 处于最大值。在 12:30 到 16:30 区间里，mel-DER 值处于缓慢下降趋势。太阳彻底下山后，mel-DER 值快速下降。具体值如下：

表 4 一天中 mel-DER 数值(部分，完整见附录)

5:30	1.038	**9:30**	1.0423	**13:30**	1.067
6:30	1.0349	**10:30**	1.0548	**14:30**	1.0568
7:30	1.0316	**11:30**	1.066	**15:30**	1.0454
8:30	1.028	**12:30**	1.076	**16:30**	1.032

5.3.1 建立数学规划模型

根据以上分析，我们的目标函数为：

1.节律效果误差最小化:

$$\min \ \mathrm{Error}_{\mathrm{mel}} = mel\ DER_{mix} - mel\ DER_{sun} \tag{17}$$

2.光谱形状误差最小化:

$$\min \ \mathrm{Error}_{\mathrm{shape}} = \sqrt{\frac{1}{N}\sum_{\lambda}\left(SPD_{mix}(\lambda) - SPD_{sun}(\lambda)\right)^2} \tag{18}$$

$$\text{约束条件:} s.t. \begin{cases} \sum_{i=1}^{5} w_i = 1 \\ w_i > 0 \end{cases}$$

该规划问题为非线性规划，我们在 python 中调用 scipy 的 SLSQP 求解器。序列二次规划（Sequential Least Squares Programming，SLSQP）是一种用于求解带约束非线性优化问题的高效数值算法，由 Dietrich Kraft 于 1988 年提出，广泛应用于工程优化、参数估计等领域。其核心思想是通过逐次构建并求解二次规划（QP）子问题，逼近原非线性优化问题的最优解，兼具收敛速度快、求解精度高的特点。

我们求解出的 15 组 w 结果如下：

表 5 w 结果(部分，完整见附录)

7:30	权重	12:30	权重	17:30	权重
w_1	0.679	w_1	0.612	w_1	0.774
w_2	0.119	w_2	0.129	w_2	0.093
w_3	0	w_3	0	w_3	0
w_4	0.117	w_4	0.124	w_4	0.091
w_5	0.0843	w_5	0.135	w_5	0.0416

5. 3. 2 模型的检验

我们的新的合成光在一天 15 个时刻的 mel-DER 与太阳光的 mel-DER 的对比图如下：

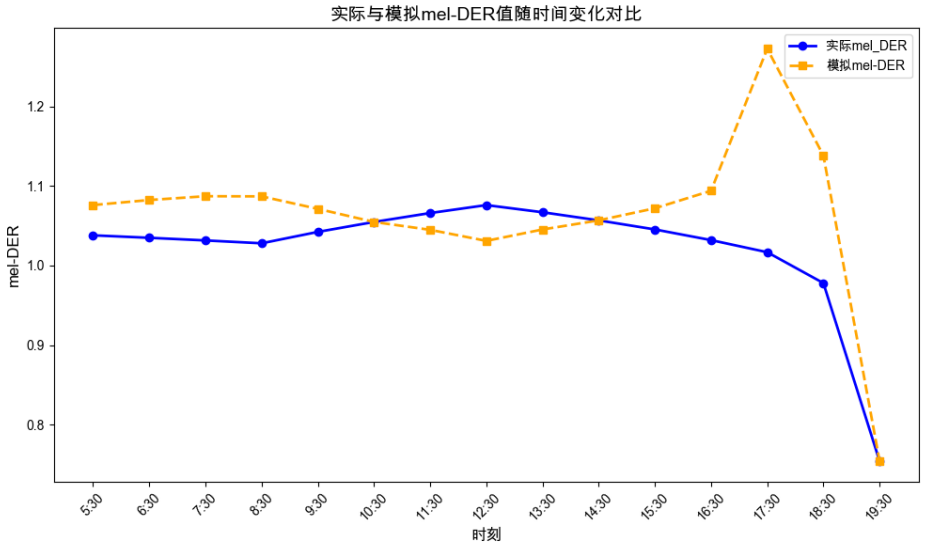

图 7 合成 mel-DER 与太阳光 mel-DER 对比

可以看出，除了在 17:30，16:30 相差较大外，在其他时间点上拟合效果较好，可以很好的模拟出太阳光对人体的节律效果。计算模型拟合的 RMSE：

$$\mathrm{RMSE}=\sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i-\hat{y}_i)^2} \tag{19}$$

RMSE=0.085，模拟太阳光节律效果较好，完成了优化目标一。

以 7:30，12:30，17:30 分别代表早，中，晚为例，我们观察合成光对太阳光的光谱拟合效果：

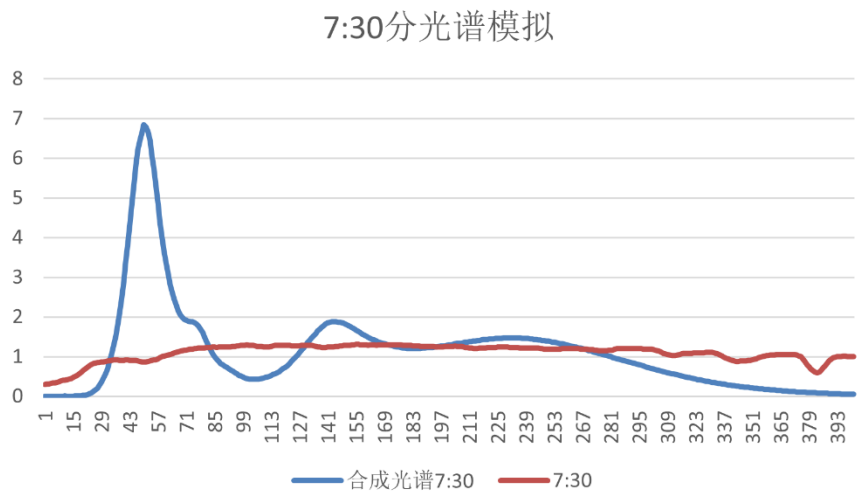

图 8 7:30 模拟光谱与太阳光对比

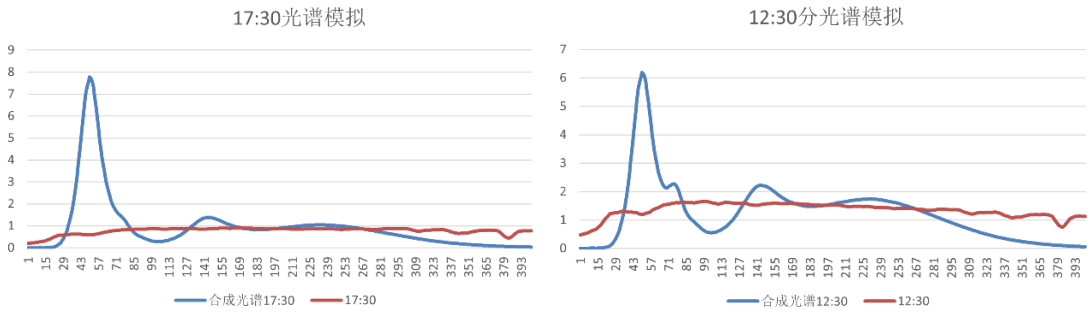

图 9　12:30 与 17:30 模拟光谱与太阳光对比

在绝大多数波长下，我们拟合出的光谱形状光谱形状与太阳光可以拟合。

对于 7:30 序列，RMSE 值为: 1.1718

对于 12:30 序列，RMSE 值为: 1.0452

对于 17:30 序列，RMSE 值为: 1.3617

可以认为我们完成了目标函数二。

综合以上，在我们计算出的五种光的权重下，我们可以实现接近太阳光的节律模拟

(RMSE= 0.085)，且最大限度的保证了合成光的光谱形状与太阳光误差最小。

5.4 问题四的建模与求解

5.4.1 构建指标并统计

本题中，我们有 11 位测试者在三次测试中的睡眠情况数据。我们需要根据这些基础睡眠数据，构造高级特征，使用统计学方法检验三种光照环境对睡眠的影响是否存在显著差异。从而得出三种光照(优化光照，普通光照，黑暗环境)对睡眠的影响。

我们的构造指标如下：(第 i 个测试者在第 j 次测试)；

1.总睡眠时间 $TST_{ij} = T_{N1}+T_{N2}+T_{N3}+T_{REM}$；

2.睡眠效率 $SE = \frac{TST}{T} \times 100\%$ (T 为关灯到睡醒总时长)；

3.入睡潜伏期 SOL = 从关灯到首次进入任何睡眠阶段；

4.深睡眠比例 $RatioN_3 = \frac{T_{N3}}{TST} \times 100\%$；

5.REM 睡眠比例 $REM\% = \frac{T_{REM}}{TST} \times 100\%$；

6.夜间醒来次数=睡眠开始后，清醒总次数；

根据原始数据，我们计算出的六个指标结果如下表：

表 7 六个指标的结果(部分)

被试	TSTmin	SE%	SOLmin	N3%	REM%	Awakenings	环境
被试 1	310	86.95%	15	14.67%	20.48%	19	优化
被试 2	348	75.24%	5	30.89%	29.74%	13	普通
被试 3	259	68.25%	54.5	21.62%	24.13	12	黑暗
被试 4	334	83.71%	40.5	24.70%	33.98%	13	优化

我们对 TST 分成三组，优化，普通，黑暗三组做小提琴图：

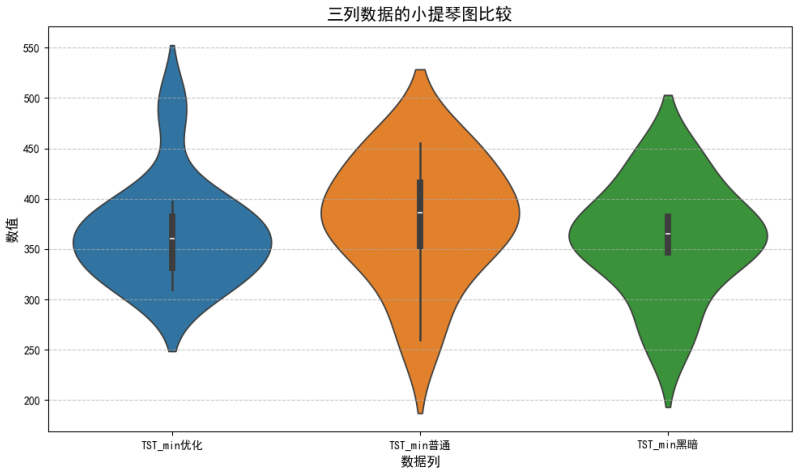

图 10　总睡眠时间小提琴图

该图直观地对比了总睡眠时间的优化光照、普通及黑暗环境这三组条件下，目标变量总睡眠时间的概率分布情况。可以看出普通光照的数据分布最为分散，数值范围最广，表明该条件下数据的波动性最大；且峰值最大。即较多测试者在普通光照下，总睡眠时间较长。优化光照组的数据分布则相对集中，峰值明确，其数值更加稳定。

观察变量 SE(睡眠效率)：

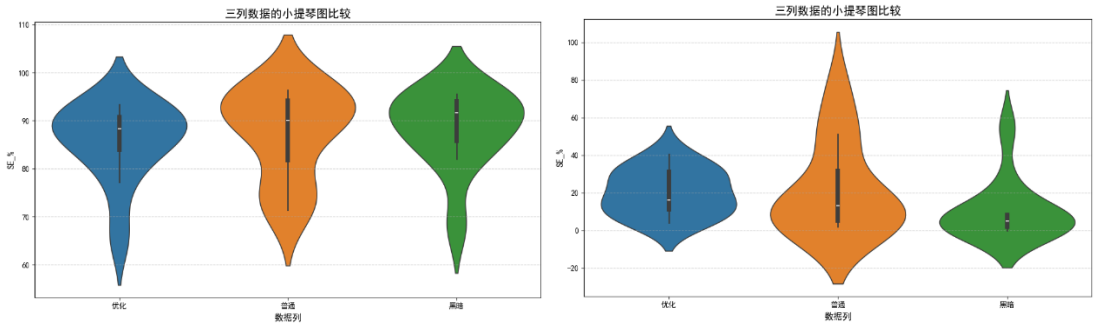

图 11　睡眠效率，入睡时间小提琴图

可以看出，普通光照的睡眠效率峰值最大但有相当一部分处于较低睡眠效率水平。其次是黑暗环境的睡眠效率峰值，略低于普通处理，且绝大多数集中在峰值附近。分布较为稳定。峰值最低为优化光照，大多数测试者的睡眠效率也集中在峰值附近。

可以明显看出，优化光照下，绝大多数测试者的入睡时间集中在 16 分钟左右，虽然峰值略高于其他两组，但非常稳定，50 分钟左右全部测试者入睡，可以有效减少入睡所需时间。普通光照虽然峰值最低，但分布不均匀，在 100 分钟是仍有测试者未入睡；黑暗环境也是类似。

观察 N_3 占比与 REM 睡眠比例变量：

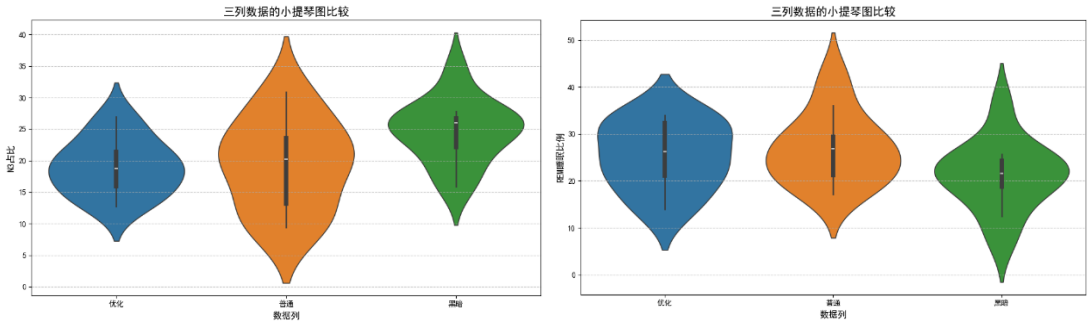

图 12 N3 占比与 REM 睡眠比例

在黑暗环境下，N3 占比情况最为优秀；其次是优化光照。优化光照下，可以使得 REM 睡眠比例稳定集中在较高水平，可以提升睡眠质量。

5.4.2 方差分析检验均值差异

下面我们对 6 个指标下的三组变量做 ANOVA 方差分析；方差分析是一种用于检验多组独立样本均值是否存在统计学显著差异的统计方法，其核心功能是通过分析数据中不同来源的变异（组间变异与组内变异），判断分组因素对观测结果的影响是否具有显著性。

原假设 H_0: 三组数据的总体均值全部相等 $(\mu_1=\mu_2=\mu_3)$。即观测到的样本均值差异是由随机抽样误差引起的。

备择假设 H_1: 至少有一个总体的均值与其他组不相等。

计算 F 统计量 (The F-Statistic):

$$F = \frac{MS_{\text{between}}}{MS_{\text{within}}} \tag{20}$$

组间均方 $MS_{between}$:由分组（潜在的均值差异）引起的变异

组内均方 MS_{within}:代表随机误差造成的变异。

对六组变量的统计结果如下：

表 8 F 检验统计量

变量	*F*	*p-value*
TSTmin	0.363	0.698
SE%	0.274	0.762
SOLmin	1.231	0.306
N3%	3.478	0.044

REM%	1.742	0.192
Awakenings	0.120	0.887

在 95%的置信区间下，仅有变量 N3%的三组变量的总体均值不全相等，我们再对变量 N3%进行事后检验。

表 9　对 N3%进行事后检验结果表

检验方法	基准组	对比组	基准-对比	*p-value*
LSD	1	2	-0.3794	0.875
		3	-5.649	0.025
塔姆黑尼	1	2	-0.3794	0.998
		3	-5.649	0.031

在事后检验中，优化亮度在 N3%指标上，显著低于黑暗环境 5.649 个百分点，其他的环境对比结果不显著。

综合以上分析，我们得出结论：**优化光源环境下的深睡眠比例显著低于黑暗环境 5.649 个百分点。**

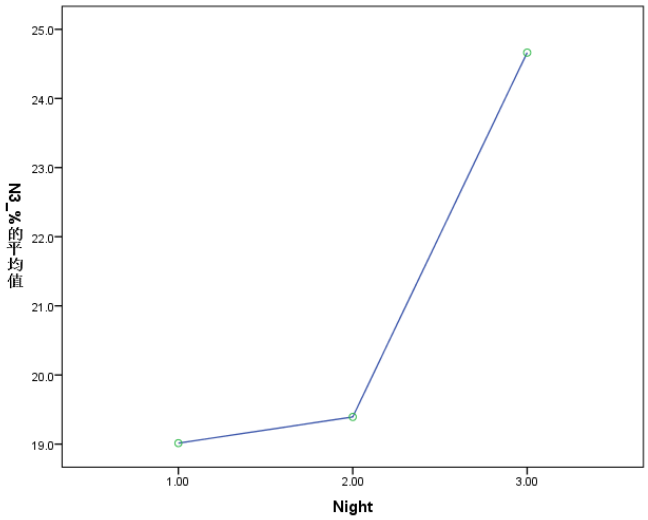

图 21 N3%的三组均值对比图

六、 模型优缺点评价

6.1 模型的优点

(1)．在 LED 光谱合成中引入非线性规划，并以 SLSQP 算法求解，能够处理复杂约束（CCT、Rg 范围、Rf 下限），并获得较高的计算效率和稳定的最优解

(2)．同时考虑了节律效果误差和光谱形状误差（RMSE）。这使得模拟结果不仅在生理效应上接近太阳光，在视觉外观上也尽可能相似。

(3)．优化解稳定且参数组合明确（各 LED 通道的权重），易于在物理意义上解释和实现

6.2 模型的缺点

(1)．SLSQP 属于局部优化方法，容易受初值影响，难以保证全局最优，可能遗漏更优解

(2)．样本量偏小，睡眠实验仅有 11 位测试者，可能会导致结果失真。

(3).实验设计可控性不足,三种光照条件可能存在顺序效应或个体差异未充分控制。

参考文献

[1]张浩,徐海松.光源相关色温算法的比较研究[J].光学仪器,2006,(01):54-58.

[2] Royer, Michael P. "Tutorial: Background and guidance for using the ANSI/IES TM-30 method for evaluating light source color rendition." Leukos 18.2 (2022): 191 231.

[3] Schlangen, Luc JM, et al. "Report on the Workshop Use and Application of the new CIE s 026/e: 2018, Metrology for ipRGC-influenced responses to light “specifying light for its eye-mediated non-visual effects in humans”." Proceedings of the 29th Session of the CIE. CIE, 2019. 114-118.

[4]常雪松.可调色温高显色 pc-LED 光源设计[D].大连工业大学,2023.

[5]李月.光源相关色温及色偏差计算方法研究[D].辽宁科技大学,2023.

[6]李宁. LED 混光的光视效能及显色性评估研究[D].北京大学,2014.

[7] Zhang, Jingjing, et al. "Blue light hazard optimization for white light-emitting diode sources with high luminous efficacy of radiation and high color rendering index." Optics & Laser Technology 94 (2017): 193-198.

附录

附录 1

介绍：完整表格

mel_DER
1.076052
1.082371
1.087143
1.086945
1.071288
1.054822
1.044909
1.031019
1.045356
1.0568
1.071991
1.093959
1.272497
1.138268
0.754201

15 个时刻的 w 数组取值：

w1	w2	w3	w4	w5
0.667836	0.124531	8.320802e-18	0.118134	8.949893e-02
0.673577	0.120927	9.879406e-16	0.117127	8.836943e-02
0.679622	0.119209	4.096138e-18	0.116911	8.425842e-02
0.678094	0.124799	6.235763e-17	0.116585	8.052189e-02
0.663096	0.123109	1.892620e-18	0.118805	9.507972e-02
0.643524	0.128934	1.794345e-17	0.120836	1.067061e-01
0.630166	0.124712	2.888977e-18	0.122069	1.239528e-01

0.611548	0.128785	1.677087e-16	0.124287	1.353801e-01
0.629807	0.125311	1.875416e-15	0.121769	1.231135e-01
0.645470	0.127068	0.000000e+00	0.120457	1.070957e-01
0.63735	0.124113	1.182456e-17	0.118797	9.344473e-02
0.632280	0.118943	5.274433e-17	0.115286	8.251671e-02
0.774490	0.093284	8.554317e-17	0.096630	4.159649e-02
0.615562	0.062214	2.533955e-01	0.955983	1.284544e-02
0.382889	0.023651	5.826353e-01	0.010825	4.607469e-14

附录 2

介绍：python 计算问题一的五个关键参数

```
import numpy as np
import pandas as pd
import colour

#计算 CCT：
def calculate_cct_from_excel():
    file_path=r"C:\Users\31498\OneDrive\Desktop\conbine.xlsx"
    df = pd.read_excel(file_path)
    wavelengths = df['Wavelength (nm)'].to_numpy()
    intensities = df['光强'].to_numpy()
    intensities = intensities / np.max(intensities)

    spectrum = np.column_stack((wavelengths, intensities))
    sd = colour.SpectralDistribution(dict(zip(wavelengths, intensities)))
    # 计算 xy 色度坐标和 CCT
    xy = colour.XYZ_to_xy(colour.sd_to_XYZ(sd))
    cct = colour.xy_to_CCT(xy)
    print(cct)

def Duv():
    file_path = r"C:\Users\31498\OneDrive\Desktop\uv.xlsx"
    df = pd.read_excel(file_path)
    if df is not None:
        us = df['u']
        vs = df['v']
        ut = df['ut']
        vt = df['vt']
        distance = np.sqrt((ut - us)**2 + (vt - vs)**2)
```

```
        sign = np.sign(vs - vt)

        df['Duv'] = sign * distance

        print("=" * 30)
        print(df[['u', 'v', 'ut', 'vt', 'Duv']].round(6))
        print("=" * 30)

def rgrf():
    file_path = r"C:\Users\31498\OneDrive\Desktop\hs2025\C 题\conbine.xlsx"
    wavelength_col_name = 'wavelength'
    intensity_col_name = 'intensity'

    df = pd.read_excel(file_path)

    spd_data = dict(zip(df[wavelength_col_name], df[intensity_col_name]))
    spd = colour.SpectralDistribution(spd_data)

    tm30_report = colour.quality.tm3018.colour_fidelity_index_ANSIIESTM3018(
        spd, additional_data=True
    )
    Rg = tm30_report.R_g
    Rf = tm30_report.R_f
    print(f"Rf={Rf:.2f}")
    print(f"Rg={Rg:.2f}")

def mel_DER():
    df1 = pd.read_excel(r"C:\Users\31498\OneDrive\Desktop\q31.xlsx")

    delt=1
    df1['num']=df1['十九点三十']*df1['melanopic']*delt
    df1['den']=df1['十九点三十']*df1["V(λ)"]*delt

    num = df1['num'].sum()
    den = df1["den"].sum()

    if den == 0:
        print('分母为 0，无法计算 mel_DER')
    else:
        mel_DER = num * 1.218 / den
        print(mel_DER)
if __name__ == '__main__':
    #Duv()
```

```
    #rgrf()
    mel_DER()
```

附录 3

介绍：python 计算问题二的优化问题

```
import numpy as np
import pandas as pd
import colour
from scipy.optimize import minimize, LinearConstraint, Bounds
from colour import SpectralDistribution, SpectralShape
from colour.quality.tm3018 import colour_fidelity_index_ANSIIESTM3018

INPUT_XLSX = r"C:\Users\31498\OneDrive\Desktop\conbine.xlsx"
SHEET_NAME = 0
COL_WL = "Wavelength (nm)"
COLS_CHANNELS = ["Blue", "Green", "Red", "Warm White", "Cold White"]
COL_V = "V(λ)"
COL_MEL = "melanopic"

# 正午日光目标
CCT_MIN, CCT_MAX = 5500.0, 6500.0   # = 6000 ± 500 K
RG_MIN, RG_MAX   = 95.0, 105.0
RF_FLOOR         = 88.0

w_seed = np.array([0.000561, 0.050749, 0.000001, 0.541419, 0.407270], dtype=float)

#计算函数
def combine_intensity(df: pd.DataFrame, w: np.ndarray) -> np.ndarray:
    """按权重线性叠加 5 个通道，得到 SPD 强度列"""
    intens = np.zeros(len(df), dtype=float)
    for wi, col in zip(w, COLS_CHANNELS):
        #zip(w, COLS_CHANNELS) 把权重和列名配对(w[0], 'channel_red')
        intens          +=          wi          *          pd.to_numeric(df[col],
errors="coerce").to_numpy()#errors="coerce"如果某个值无法转换，就把它变成 NaN
    return np.clip(intens, 1e-9, None)
#np.clip(array, a_min, a_max)：将数组中的值限制在 [a_min, a_max] 范围内。

#计算 cct 函数
def calculate_cct(df_xy: pd.DataFrame) -> float:
    """
```

```
    用 colour 库函数计算 CCT（df 需含 'Wavelength (nm)' 与 'intensity'）
    """
    wavelengths = pd.to_numeric(df_xy["Wavelength (nm)"],
errors="coerce").to_numpy()
    intensities = pd.to_numeric(df_xy["intensity"], errors="coerce").to_numpy()

    mask = np.isfinite(wavelengths) & np.isfinite(intensities)
    wavelengths = wavelengths[mask]
    intensities = intensities[mask]

    if len(wavelengths) == 0 or np.all(intensities <= 0):
        return np.nan
    intensities = intensities / np.max(intensities)

    sd = SpectralDistribution(dict(zip(wavelengths, intensities)))
    XYZ = colour.sd_to_XYZ(sd)
    xy = colour.XYZ_to_xy(XYZ)
    cct = colour.xy_to_CCT(xy)
    try:
        return float(cct)
    except Exception:
        return float(cct[0])

#计算 Rf，Rg 函数
def tm30_Rf_Rg(wavelengths_nm: np.ndarray, intensity: np.ndarray) -> tuple[float,
float]:
    """TM-30 Rf、Rg（将 SPD 对齐到 380–780 nm，Δλ=1nm）"""
    shape = SpectralShape(380, 780, 1)
    sd = SpectralDistribution(dict(zip(wavelengths_nm, intensity))).align(shape)
    spec = colour_fidelity_index_ANSIIESTM3018(sd, additional_data=True)
    if hasattr(spec, "R_f"):
        return float(spec.R_f), float(spec.R_g)
    else:
        return float(spec["R_f"]), float(spec["R_g"])

#计算 mel_DER 函数
def calc_mel_DER(df: pd.DataFrame, intensity: np.ndarray) -> float:
    #mel-DER = (Σ SPD*melanopic) / (Σ SPD*V(λ)) * 1.218
    mel = pd.to_numeric(df[COL_MEL], errors="coerce").to_numpy()
    V   = pd.to_numeric(df[COL_V],   errors="coerce").to_numpy()
    num = float(np.nansum(intensity * mel))
    den = float(np.nansum(intensity * V))
    if den <= 1e-12:
        return np.inf
```

```
    return num * 1.218 / den

#优化：最大化 Rf（正午日光模式）
def optimize_max_Rf_noon(w0: np.ndarray):
    df = pd.read_excel(INPUT_XLSX, sheet_name=SHEET_NAME)
    wavelengths = pd.to_numeric(df[COL_WL], errors="coerce").to_numpy()

    # 线性约束 & 边界：sum(w)=1, w_i >= 1e-6
    n = len(COLS_CHANNELS)
    lc = LinearConstraint(np.ones((1, n)), [1.0], [1.0])
#LinearConstraint 表示优化问题中的线性约束条件:形如 lb≤A·x≤ub 的约束,参数 1：
np.ones((1, n))：这是约束的系数矩阵 A；参数 2：约束的下界 lb;参数 3：约束的上界 ub
#这里即为等式约束，即 w1+w2...+wn=1
    bnds = Bounds(np.full(n, 1e-6), np.ones(n))
#Bounds 用于为优化变量设置上下界约束，参数 1：所有变量的下界数组，参数 2：所有变量的
上界数组；np.ones(n)：创建长度为 n 的数组
#np.full(n, 1e-6) 作用是创建一个长度（或元素总数）为 n 的数组，其中所有元素的值都被
填充为 1e-6。

    # 定义非线性不等式约束（全部写成 >= 0）
    def cons_noon(w: np.ndarray) -> np.ndarray:#w: np.ndarray 表示参数 w 的预期类
型是 numpy.ndarray；-> np.ndarray 表示函数的返回值预期类型是 numpy.ndarray
        intensity = combine_intensity(df, w)

        # CCT（库函数）
        df_tmp = df[[COL_WL]].copy()
        df_tmp["intensity"] = intensity
        CCT = calculate_cct(df_tmp)

        # 计算 Rf 函数 tm30_Rf_Rg
        Rf, Rg = tm30_Rf_Rg(wavelengths, intensity)

        if not (np.isfinite(CCT) and np.isfinite(Rf) and np.isfinite(Rg)):
            #isfinite 检查哪些元素是有限的,有限则返回 true
            return np.array([-1.0, -1.0, -1.0, -1.0, -1.0])

        return np.array([
            CCT - CCT_MIN,          # >=0  (CCT 下限)
            CCT_MAX - CCT,          # >=0  (CCT 上限)
            Rg  - RG_MIN,           # >=0
            RG_MAX - Rg,            # >=0
            Rf  - RF_FLOOR          # >=0
        ])
    nlc = {'type': 'ineq', 'fun': cons_noon}#定义了一个非线性约束，type': 'ineq':
```

表示这是一个不等式约束，'fun': cons_noon：指定约束函数为 cons_noon

```
    # 目标：最大化 Rf → 最小化 -Rf
    def neg_Rf(w: np.ndarray) -> float:
        intensity = combine_intensity(df, w)
        Rf, _ = tm30_Rf_Rg(wavelengths, intensity)
        return 1e6 if not np.isfinite(Rf) else -Rf
    #如果 Rf 有效 → 返回 -Rf；RF 为无限，则返回 1e6
    #初始点净化
    w0 = np.clip(np.asarray(w0, float), 1e-6, 1.0)
    w0 = w0 / w0.sum()

    res = minimize(
        neg_Rf, w0, method='SLSQP',
        bounds=bnds, constraints=[lc, nlc],#constraints 参数用于指定优化问题中的约束条件,lc 为线性约束，nlc 为非线性约束
        options={'maxiter': 900, 'ftol': 1e-9, 'disp': True}
    )#'ftol': 1e-9：

    # 结果
    w_star = res.x
    intensity = combine_intensity(df, w_star)
    df_tmp = df[[COL_WL]].copy()
    df_tmp["intensity"] = intensity

    CCT = calculate_cct(df_tmp)
    Rf, Rg = tm30_Rf_Rg(wavelengths, intensity)
    mel_DER = calc_mel_DER(df, intensity)

    print("\n 最大化 Rf")
    print(f"success: {res.success}, message: {res.message}")
    for name, wi in zip(COLS_CHANNELS, w_star):
        print(f"{name:>10s}: {wi:.6f}")
    print(f"CCT    : {CCT:.2f} K  (目标 6000±500K)")
    print(f"Rf     : {Rf:.2f}     (≥88，越高越好)")
    print(f"Rg     : {Rg:.2f}     (95~105)")
    print(f"melDER: {mel_DER:.6f}")

    # 保存
    out = df[[COL_WL]].copy()
    out["intensity"] = intensity
    for name, wi in zip(COLS_CHANNELS, w_star):
        out[f"w*{name}"] = wi * pd.to_numeric(df[name], errors="coerce").to_numpy()
    meta = pd.DataFrame({
```

```
        "Metric": ["CCT[K]", "Rf", "Rg", "mel-DER"] + [f"w_{n}" for n in
COLS_CHANNELS],
        "Value": [CCT, Rf, Rg, mel_DER] + list(w_star),
    })
    with pd.ExcelWriter("noon_Rf_max_result.xlsx", engine="openpyxl") as ew:
        out.to_excel(ew, index=False, sheet_name="combined_spectrum")
        meta.to_excel(ew, index=False, sheet_name="metrics")

    return w_star, {'CCT': CCT, 'Rf': Rf, 'Rg': Rg, 'melDER': mel_DER}, res

if __name__ == "__main__":
    optimize_max_Rf_noon(w_seed)#w_seed：定义的初始权值
```

附录 4

介绍：python 计算问题三的优化问题

```
import pandas as pd
import numpy as np
from scipy.optimize import minimize

df1 = pd.read_excel(r"C:\Users\31498\OneDrive\Desktop\hs2025\C 题\conbine.xlsx")
time_points = [f"{hour}点三十" for hour in range(5, 20)]

melDER = [1.038, 1.0349, 1.0316, 1.028, 1.0423, 1.0548, 1.066, 1.076, 1.067, 1.0568,
1.0454, 1.032, 1.0168, 0.978, 0.7542]

# 定义优化目标函数
def objective_function(weights, df, time_col, melDER_target):
    w1, w2, w3, w4, w5 = weights

    # 计算 mix1
    df["mix1"] = w1 * df["Blue"] + w2 * df["Green"] + w3 * df["Red"] + w4 * df["Warm
White"] + w5 * df["Cold White"]

    total = (df[time_col] - df["mix1"]).abs().sum()

    delt = 1
    df['num'] = df['mix1'] * df['melanopic'] * delt
    df['den'] = df['mix1'] * df["V(λ)"] * delt
    num = df['num'].sum()
    den = df['den'].sum()
```

```
    if den == 0:
        return np.inf
    mel_DER = num * 1.218 / den

    delta_mel = abs(mel_DER - melDER_target)

    #目标函数值和 mel_DER
    return 0.8 * delta_mel + 0.2 * total, mel_DER
# 存储权重结果、mel_DER 和 mix1 数据
results = []
mix1_dfs = []

# 对每个时间点进行优化
for i, time in enumerate(time_points):
    # 复制原始 DataFrame，避免修改原始数据
    df_temp = df1.copy()

    # 定义约束条件：w1 + w2 + w3 + w4 + w5 = 1
    constraints = ({'type': 'eq', 'fun': lambda w: w[0] + w[1] + w[2] + w[3] + w[4]
- 1})

    # 定义边界条件：每个权重在 [0, 1] 之间
    bounds = [(0, 1)] * 5
    initial_guess = [0.2, 0.2, 0.2, 0.2, 0.2]
    result = minimize(
        lambda w: objective_function(w, df_temp, time, melDER[i])[0],  # 仅优化目
标函数
        initial_guess,
        method='SLSQP',
        bounds=bounds,
        constraints=constraints
    )
    # 保存优化结果
    if result.success:
        # 计算优化后的 mix1 和 mel_DER
        df_temp["mix1"] = (result.x[0] * df_temp["Blue"] +
                           result.x[1] * df_temp["Green"] +
                           result.x[2] * df_temp["Red"] +
                           result.x[3] * df_temp["Warm White"] +
                           result.x[4] * df_temp["Cold White"])

        # 计算 mel_DER
        _,  optimized_mel_DER  =  objective_function(result.x,  df_temp,  time,
```

```
melDER[i])

        # 保存 mix1 列，命名为对应时间点
        mix1_dfs.append(df_temp[["mix1"]].rename(columns={"mix1":
f"mix1_{time}"}))

        # 保存权重、mel_DER 和目标函数值
        results.append({
            '时间点': time,
            'w1': result.x[0],
            'w2': result.x[1],
            'w3': result.x[2],
            'w4': result.x[3],
            'w5': result.x[4],
            'mel_DER': optimized_mel_DER,
            '目标函数值': result.fun
        })
results_df = pd.DataFrame(results)
# 合并所有 mix1 列
mix1_combined = pd.concat(mix1_dfs, axis=1)

print(results_df)
```