基于非线性规划的定日镜场优化设计

摘　要

随着世界能源问题愈发严重，人类迫切地需要环保持久的新型能源来代替传统能源。本文通过建立几何光路图，分析塔式太阳能光热发电中出现的效率及功率模型，并通过对定日镜场内部定日镜、集热器、吸收塔等设备参数的**非线性规划设计**，通过**随机模拟镜场**与**规划权值**，给出了问题中三种参数设定下镜场的最佳规划与对应产能。

针对问题一：首先根据定日镜、集热器位置建立镜场坐标系，对所给数据进行方案规范检测。再在每一个计算时点，构建“日塔镜”几何光学模型，基于**欧式空间基本理论**与**离散化思想**，对效率、功率模型进行了创新性研究，进而准确求解了每月平均光学效率(1 月 21 日：0.416361)及输出功率和年平均光学效率（0.443487537）及输出功率等，结果详细见正文中表 1，表 2。

针对问题二：首先考虑镜场和吸收塔的相对位置关系，建立对应坐标系。基于问题一所构建的效率模型，通过在镜场内多次**模拟随机分布**大量合理的定日镜，实现对镜场等分十块区域的权重划分。再通过对定日镜场多参数**非线性规划**，得出定日镜最优尺寸$4m\times7m$，从而求解定日镜数（3290）并按权重在对应区域划分数目。通过在扇形从内到外符合镜场规划要求地均匀排布，求解对应点坐标，再将其回带至问题一中模型，反证产能，进行误差检测，并求得此时参数设定下光学效率及输出功率和年平均平均光学效率及输出功率，结果详细见表 4，表 5。

针对问题三：在问题二的基础上，进一步分析非线性规划模型中镜面尺寸、镜高对目标功率函数的影响。结合前两问效率模型，**弱化阴影遮挡效果**，简化权重划分模型。重复问题二中非线性规划做法，同求出最优尺寸$4m\times7m$，在改进后的权重模型中，多次模拟不同特征的定日镜随机分布，得到每个区域权值的优化解。在求解定日镜数（2307）后，仍采用均匀排布法求解点坐标。再将其回带至问题一中模型，反证产能，进行误差检测，并求得此时参数设定下光学效率及输出功率和年平均平均光学效率及输出功率，结果详细见表 8，表 9。

本文所建模型基本为原创，创新性较强，代码函数封装性好，具有较好的移植性，对于解决实际定日镜场规划问题具有一定参考价值。

关键词：几何模型 非线性规划 随机分布 规划权值 定日镜场

一、 问题重述

1.1 问题背景

随着人类社会经济科技的高速发展，能源需求日益紧张，环境问题愈发严重，人们需要寻找环保持久的新能源来代替传统能源。为实现“碳达峰”“碳中和”目标，我国也在积极构建以新能源为主体的新型电力系统，其中塔式太阳能光热发电正是一种低碳环保的新型清洁能源技术。

一种塔式太阳能发电站原理为通过控制定日镜场内大量定日镜上平面反射镜的俯仰角，使太阳光反射汇聚到吸收塔塔端的集热器中，实现导热介质将太阳能以热能储存，最终实现向电能的转化。

1.2 问题提出

该塔式太阳能发电站本质为一半径为$350m$的圆形定日镜场，由高度为$80m$的吸收塔、高$8m$直径$7m$的圆柱形集热器、吸收塔周围$100m$范围外的定日镜以及用于发电、储能、控制的设备组成。通过调整太阳光、定日镜、集热器三者之间的几何模型，可以达到单镜面的输出功率最大化。进一步优化定日镜场内吸收塔的位置以及定日镜的分布、高度、尺寸及数目，可以有效减少阴影遮挡损失、余弦损失及集热器截断损失，从而提高定日镜光学效率，实现定日镜场内年平均输出热功率最大化。

基于所给的背景及附录信息，我们需要建立数学模型来解决以下问题：

问题一：将吸收塔建设在圆形镜场中心，固定所有定日镜的尺寸为$6m \times 6m$，安装高度为$4m$，在已知所有定日镜位置下，建立太阳、定日镜和集热器的几何模型，计算镜场的年平均光学效率、年平均输出热功率和单位镜面面积年平均输出热功率。

问题二：按定日镜场额定年平均功率$60MW$及所有定日镜尺寸及安装高度相同的设计要求，优化吸收塔的位置和定日镜尺寸、高度、数目及位置，以达到最大化单位镜面年平均输出热功率的效果。

问题三：功率设置同问题二，允许镜场内定日镜尺寸及安装高度不同，重新优化设计各参数，实现最大化单位镜面面积年平均输出热功率。

二、 问题分析

2.1 问题一的分析

该题本质上是一个几何问题，需要按照题目给定的定日镜位置、集热器位置先建立镜场坐标系，然后根据镜场的规划方案对数据进行异常检测。再在每一个计算时点，构建日镜塔几何光学模型，根据附录相关计算公式求出相应解。本题难点在于求解余弦效率、阴影遮挡效率及截断效率模型。对此，我们基于欧氏空间基本理论与离散化思想，理清题目数据内在联系，灵活选择向量，实现了三者的高精度求解。

2.2 问题二的分析

该题本质上是一个优化问题，本质上只需要考虑镜场和吸收塔的相对位置关系，

为方便起见，我们在优化过程中将吸收塔位置始终定于原点。基于问题一所建立的效率计算模型，通过大量生成不同定日镜坐标及数目的案例，选定权值判定因子，对镜场内等分的十块区域进行权重划分。在给定额定年平均输出热功率及参数范围的条件下，对定日镜场内多参数进行非线性规划优化设计，求出最大单位镜面面积年平均输出热功率下定日镜尺寸和高度，从而求得当前状态下镜场内定日镜的数目。根据权重可得出每一块区域内定日镜的数目。再通过随机种子生成一定数目的定日镜坐标，在不违背镜场布局的情况下，反向验证额定年平均输出，求得题目中表 1，表 2，表 3 的最优解。

2.3 问题三的分析

该题本质上是问题二的变式，即在问题二的条件不同定日镜尺寸和高度可自由更改。在此条件下，定日镜的安排空间得到最大化利用，阴影遮挡效率影响可降至最低。由于光学效率与距离负相关，定日镜面需要尽量靠近吸收塔，且因为尺寸、高度变化可实现密铺。在此基础下重复问题二的操作，优化所生成的案例，提高实验次数，求得最优解。

三、 模型假设

1. 忽略定日镜、集热器仪器自身损耗。
2. 假设年均计算指标时间时点对应天气均为晴。
3. 忽略镜场地形变化对定日镜及集热器高度的改变。
4.

四、 符号说明

符号	说明	单位
α_s	太阳高度角	rad
γ_s	太阳方位角	rad
φ	当地纬度	度
ω	太阳时角	/
ST	当地时间	h
δ	太阳赤纬角	rad
DNI	法相直接辐射辐照度	kW/m^2
G_0	太阳常数	kW/m^2
H	海拔高度	km
E_{field}	定日镜场的输出热功率	kW
N	定日镜总数	面

A_i	第 i 面定日镜采光面积	m^2
η_i	为第 i 面镜子的光学效率	/
η_{sb}	阴影遮挡效率	/
$\eta_{\cos}$	余弦效率	/
η_{at}	大气透射率	/
η_{trunc}	集热器截断效率	/
η_{ref}	镜面反射率	/
d_{HR}	镜面中心到集热器中心的距离	m
l	镜面高度的一半	m
w	镜面宽度	m
h	镜面安装高度	m

五、 模型的建立与求解

5.1 问题一模型的建立与求解

5.1.1 数据预处理

根据附录中所给的1745台定日镜的坐标数据及题目背景中指定的定日镜场规划，利用 SPSSPRO 判断附录数据是否合理，即定日镜是否均位于吸收塔周围100m 范围之外。

经分析，所给数据均合理。以圆形镜场中心为坐标原点，正东方向为x 轴正向，正北方向为y 轴正向，垂直地面向上方向为z 轴正向，建立立体镜场坐标系。利用 Matlab 作出定日镜分布图，如图 1 所示。

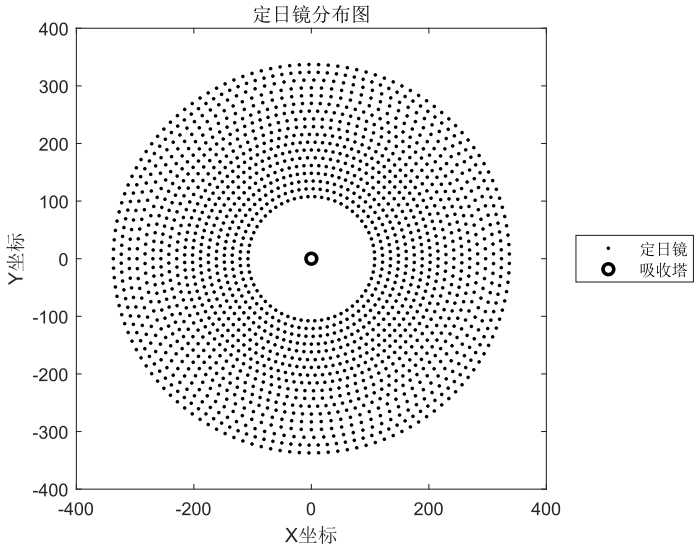


图 1：定日镜分布图

5.1.2 部分变量求解

根据题目附录所给出的相关计算公式及问题一内容，分析画出各变量求解关系的有向无环图，如图 2 所示。

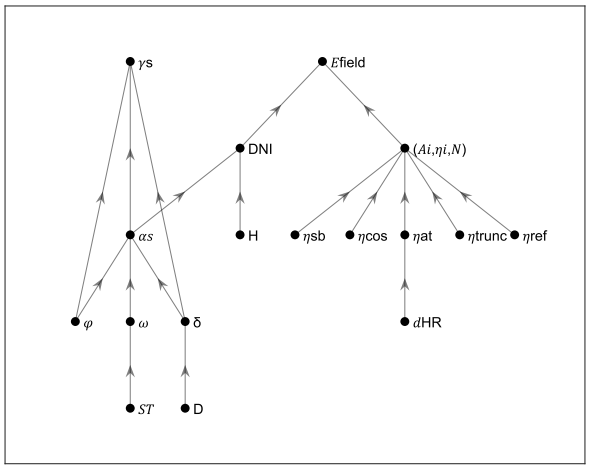


图 2：问题一各变量求解关系

太阳高度角α_s满足：

$$\sin\alpha_s = \cos\delta\cos\varphi\cos\omega + \sin\delta\sin\varphi \tag{1}$$

太阳方位角γ_s满足：

$$\cos\gamma_s = \frac{\sin\delta - \sin\alpha_s\sin\varphi}{\cos\alpha_s\cos\varphi} \tag{2}$$

其中φ为当地纬度，以北纬为正，题目中给出镜场中心东经$98.5°$，北纬$39.4°$；太阳时角ω满足：

$$\omega = \frac{\pi}{12}(ST - 12) \tag{3}$$

其中ST为当地时间，计算指标时点为9：00、10:30、12:00、13:30、15:00；

太阳赤纬角δ满足：

$$\sin\delta = \sin\frac{2\pi D}{365}\sin\left(\frac{2\pi}{360}23.45\right) \tag{4}$$

其中 D 为以 3 月 21 日为第 0 天起算的天数，计算指标日均为每月 21 日。

根据公式(1)(2)(3)(4)，利用 C 语言编写程序求解每一个计算时点下的ω、δ、α_s、γ_s，具体代码见附录 first.cpp，

又已知定日镜场的海拔H为$3km$，根据公式：

$$\begin{cases} DNI = G_0\left[a + b\exp\left(-\dfrac{c}{\sin\alpha_s}\right)\right] \\ a = 0.4237 - 0.00821(6 - H)^2 \\ b = 0.5055 + 0.00595(6.5 - H)^2 \\ c = 0.2711 + 0.01858(2.5 - H)^2 \end{cases} \tag{5}$$

利用 Python 编程求得对应计算时点的太阳对定日镜的法相直接辐射辐强度DNI。

因为本题所求的定日镜场输出热功率E_{field}满足：

$$E_{\text{field}} = \text{DNI} \cdot \sum_{i}^{N} A_i\eta_i \tag{6}$$

且定日镜的光学效率η由阴影遮挡效率η_{nb}、余弦效率$\eta_{\cos}$、大气透射率η_{at}、集热器截断效率η_{trunc}及镜面反射率η_{ref}组成：

$$\eta = \eta_{\text{sb}}\eta_{\cos}\eta_{\text{at}}\eta_{\text{trunc}}\eta_{\text{ref}} \tag{7}$$

所以本题关键在于建立相关模型求解效率问题，其中η_{ref}由于每一面镜子考虑材质工艺均相同，可统一取常数0.92，大气透射率η_{at}满足：

$$\eta_{\text{at}} = 0.99321 - 0.0001176d_{\text{HR}} + 1.97\times10^{-8}\times d_{\text{HR}}^2 \quad (d_{\text{HR}} \le 1000) \tag{8}$$

其中d_{HR}表示每一个对日镜镜面中心与集热器中心的距离，利用 C 语言编程遍历附录所有坐标进行带入求解，算出每一点的η_{at}，求得平均值。

5.1.3 余弦效率模型求解

由于太阳入射光线方向与镜面法向存在夹角，使得反射效率下降，吸收能量减

少，基于欧式空间基本理论，作出“太阳-定日镜-集热器”光路图，如图 3 所示。

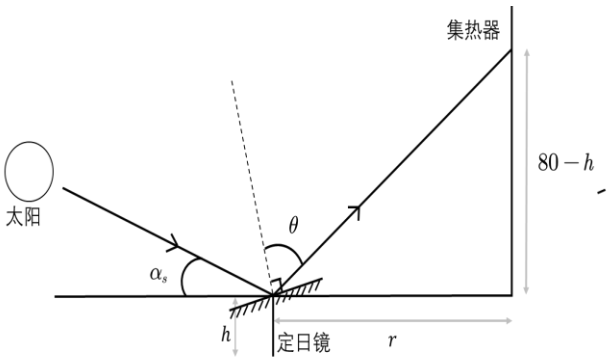


图 3：“太阳-定日镜-集热器”光路图

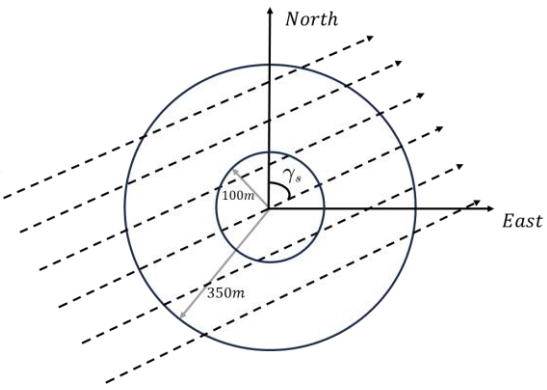


图 4：太阳光线在地面投影图

根据图示可以表示出太阳光的方向向量：

$$(x,y,z) = (\sin\gamma_s cos\alpha_s, \cos\gamma_s cos\alpha_s, -sin\alpha_s) \tag{9}$$

利用向量内积与余弦关系，可知：

$$\theta = \frac{1}{2}\arccos\left(\frac{x\sin\gamma_s\cos\alpha_s + y\cos\gamma_s\cos\alpha_s - (h-80)\sin\alpha_s}{\sqrt{x^2+y^2+(h-80)^2}}\right. \tag{10}$$

其中α_s为太阳高度角，γ_s为太阳方位角（如图 4 所示）。由余弦效率定义可得：

$$\eta_{\cos} = \cos\theta \tag{11}$$

5.1.4 集热器截断效率模型求解

集热器截断效率η_{trunc}满足：

$$\eta_{\text{trunc}} = \frac{\text{集热器接收能量}}{\text{镜面全反射能量} - \text{阴影遮挡损失能量}} \tag{12}$$

可理解为镜面反射光经大气透射损耗，既由于光线锥形原因，部分打在吸收塔集热器区域，部分打在非集热器区域所导致的损耗，又由于反射光未从柱形集热器法向射入导致产生余弦损失。

Step1：考虑锥形损耗：

根据日地半径比例，有相似公式：

$$\frac{R_E}{R_S} = \frac{\Delta l_m}{x_m} \tag{13}$$

其中R_E为地球半径，R_S为太阳半径，x_m为镜面中心到集热器中心的最大距离，Δl_m为锥形光导致的边长半增量，根据附录数据可得$\Delta l_m = 0.01532m$，从而可得$l + \Delta l_m <$集热器半径$(3.5m)$，其中l为镜面高度的一半，故在此小问中可以认为无锥形损耗。

Step2：考虑余弦损失：

将集热器视为圆柱体模型，由于定日镜反射光线方向与柱面法向不平行，会导致集热器吸收能量损失。

设定日镜反射光线与横截面所成线面角为α，如图 5 所示。反射光线与柱面交点到该点所处横截面圆心的连线和反射光线在该横截面的投影所成角为ρ，如图 6 所示。因为光线在圆柱体表面分布连续且均匀，ρ可用积分平均化处理得：

$$\cos\rho = \frac{1}{2r}\int_{-r}^{r}\frac{\sqrt{r^2-x^2}}{r}dx \tag{14}$$

由二面角公式可得：

$$\text{集热器余弦损失}\eta_{\cos'} = \cos\alpha\cos\rho \tag{15}$$

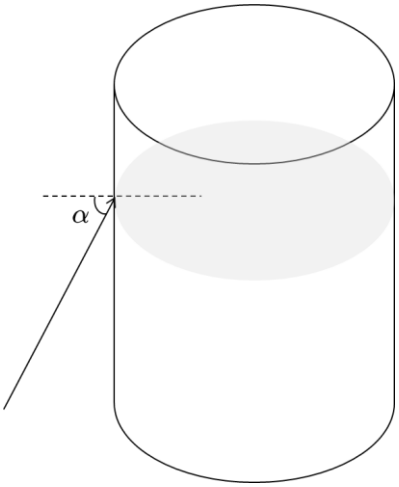


图 5：余弦损失侧视图

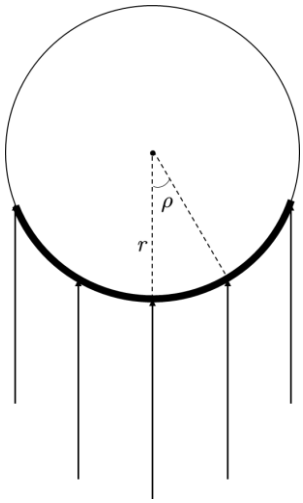


图 6：余弦损失俯视图

Step3：考虑高斯分布：

在实际情况中，定日镜中心并不能每次都准确定位吸收塔中心，根据概率统计定位合格率符合高斯分布 3σ 原则，取 $P(\mu-3\sigma\leqslant x\leqslant\mu+3\sigma)=0.997$ (下记为 $\eta_P=0.997$)

根据以上分析，可得：

$$\eta_{trunc} = \eta_{at}\eta_{\cos'}\eta_P \tag{16}$$

5.1.5 阴影遮挡效率模型求解

由于各定日镜面间的相对位置关系，会出现两种不同的阴影遮挡，即镜面对入射光线和反射光线的遮挡（以下简称入射遮挡与反射遮挡），如图 7、图 8 所示：

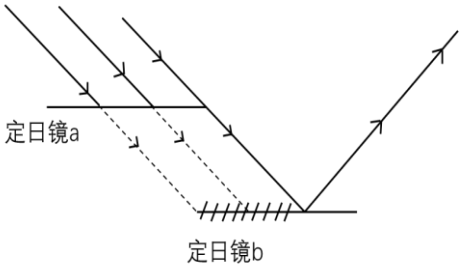


图 7：入射遮挡图

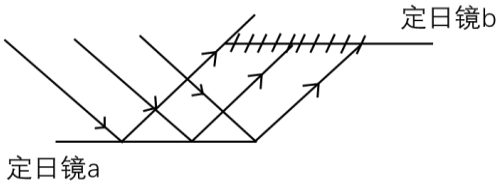


图 8：反射遮挡图

对于入射遮挡求解我们建立定日镜间相互关系的几何模型，算出单个矩形镜面的四个顶角并将其延太阳光方向后方投影(以面向太阳方向为前方)，对与单个镜面，距离与之太远或在其后方的镜面的遮挡情况均不考虑。若被多个镜面遮挡，则要注意减去重叠部面积。以下我们计算一个定日镜的相关值即可，设定日镜中心坐标为 $i_0=(x,y,h)$ 则四点坐标如图 9 示意方法求解。[1]

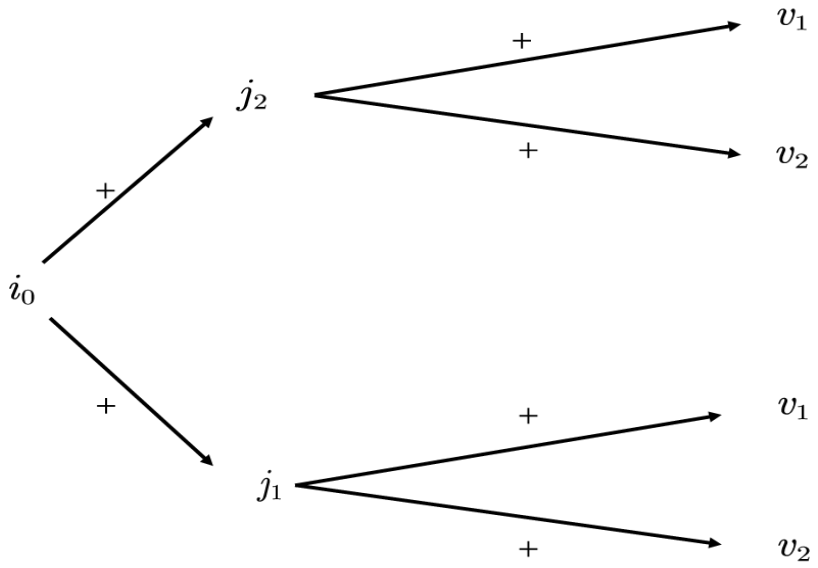


图 9：定日镜四个顶角投影相关计算图

其中：

$$\begin{cases} i_0=(x,y,h) \\ j_1=\left(|\alpha_z|\dfrac{\alpha_x}{\sqrt{\alpha_x^2+\alpha_y^2}}l,|\alpha_z|\dfrac{\alpha_y}{\sqrt{\alpha_x^2+\alpha_y}}l,-l\sqrt{\alpha_x^2+\alpha_y^2}\right) \\ j_2=\left(-|\alpha_2|\dfrac{\alpha_x}{\sqrt{\alpha_x^2+\alpha_y^2}}l,-|\alpha_8|\dfrac{\alpha_y}{\sqrt{\alpha_x^2+\alpha_y^2}}l,l\sqrt{\alpha_x^2+\alpha_y^2}\right) \\ v_1=\left(w\dfrac{-\alpha y}{\sqrt{\alpha_x^2+\alpha_y^2}},w\dfrac{\alpha x}{\sqrt{\alpha_x^2+\alpha_y^2}},0\right) \\ v_2=\left(w\dfrac{\alpha y}{\sqrt{\alpha_x^2+\alpha y^2}},-w\dfrac{\alpha x}{\sqrt{\alpha_x^2+\alpha y^2}},0\right) \end{cases} \tag{17}$$

$(\alpha_x, \alpha_y, \alpha_z)$为太阳光方向向量

注：求反射遮挡时，只用将太阳方向向量角换为反射方向的负向量方向即可。

5.1.6 整体模型计算

由公式(7)及上文所求出的各效率，某月单位面积镜面平均输出热功率P_{mi}满足：

$$P_{mi} = DNI_i \times \eta_i \tag{18}$$

其中DNI_i和η_i代表该月 21 日法相直接辐射强度和定日镜光学效率的均值。

因本小题镜子尺寸固定为$S = 6m \times 6m$，定日镜场的输出热功率 E_{field}满足：

$$E_{field} = \overline{P_{mi}} \times S \times N \tag{19}$$

其中 $\overline{P_{mi}}$为单位面积镜面年平均输出热功率，根据公式(19)求解出对应 12 个月的值的平均值，N 为定日镜总数，附件中给定为 1745 面。

5.1.7 问题一求解结果

表 1：问题一 每月 21 日平均光学效率及输出功率

日期	平均光学效率	平均余弦效率	平均阴影遮挡效率	平均截断效率	单位面积镜面平均输出热功率（Kw/m2）
1 月 21 日	0.416361	0.719009	0.931474	0.709308	0.363018
2 月 21 日	0.437487	0.739909	0.947014	0.709308	0.412545
3 月 21 日	0.452503	0.76095	0.950773	0.709308	0.450137
4 月 21 日	0.46497	0.77934	0.952211	0.709308	0.478528
5 月 21 日	0.4719	0.789317	0.952984	0.709308	0.493069
6 月 21 日	0.474037	0.792359	0.953247	0.709308	0.497402
7 月 21 日	0.471833	0.789212	0.952988	0.709308	0.492925
8 月 21 日	0.464492	0.778637	0.952169	0.709308	0.477487
9 月 21 日	0.451765	0.759841	0.950623	0.709308	0.448386
10 月 21 日	0.435334	0.73726	0.946046	0.709308	0.407129
11 月 21 日	0.414341	0.717401	0.929805	0.709308	0.358209
12 月 21 日	0.404343	0.709629	0.91888	0.709308	0.336451

表 2：问题一 年平均光学效率及输出功率

年平均光学效率	年平均余弦效率	年平均阴影遮挡效率	年平均截断效率	年平均输出热功率（MW）	单位面积镜面年平均输出热功率（kW/m2）
0.443487537	0.756072	0.944851167	0.709308	27.30202221	0.434607167

5.2 问题二模型的建立与求解

5.2.1 随机模拟镜场分布与规划权值

由于镜场和吸收塔的相对位置不会因吸收塔位置变化而变化，所以在此优化问题中，我们不妨将吸收塔位置始终置于原点，便于后续计算。

将圆形定日镜场均匀分为十个区，每个区中心线恰为某一计算时点时太阳光线向量在地面的投影，如图 10 l_2 灰色所示。

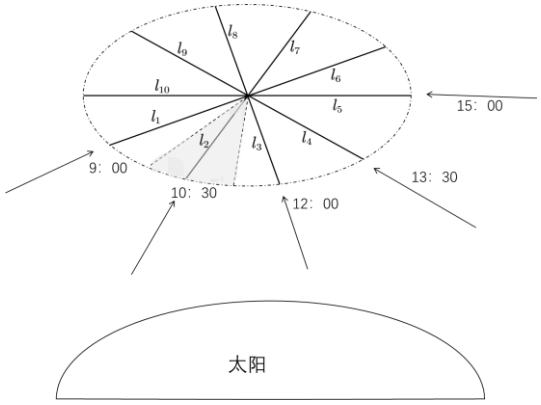


图 10：镜场分区图

只考虑太阳方位对权值的影响，因大气透射效率η_{at}、集热器截断效率η_{tranc}不随太阳方位改变而改变，所以不用于权值判定参考。记权值判定因子为J_i，权重为P_i，满足方程组：

$$\begin{cases} J_i = \overline{\eta_{\cos i}}\,\overline{\eta_{sbi}} \\ P_i = \dfrac{J_i}{\sum_{i=1}^{10} J_i} \end{cases} \tag{20}$$

为获得由大量样本数据平均而成的$\overline{\eta_{\cos i}}$和$\overline{\eta_{sbi}}$的，可以利用 C++语言编写定日镜在镜场内随机分布的程序，保证符合镜场规划的同时，获得多种方案下定日镜的数目和在镜场坐标系中的坐标分布。

由于传统随机数为假随机数，不适用于拟合足够多的随机模拟镜场，我们改良了随机数的选择方法，利用 C++的$srand(\ )$函数，获取当前时间的时间戳，转换为

无符号整数，作为种子传递给随机数生成函数。由于本题生成的每个随机点需要满足：1.两两随机点之间距离大于$12m$；2.每个随机点均位于$R_{小圆}=100m, R_{大圆}=350m$的圆环区域内。而种子函数可以根据需要选择不同的种子策略，确保生成的随机数可以满足特定的要求。同时由于随机分布量大的时，最后取得的点可以在平面内可以看作均匀分布，在多次拟合过程中更易于寻得最优解。

将获得的多种方案下定日镜数目及位置坐标带入已建模型分别求出每一块区域的$\overline{\eta_{\cos i}}$、$\overline{\eta_{sbi}}$等平均效率及$\overline{DNI_i}$，根据公式(20)，可求得对应J_i和P_i，记录为表 3。

表 3：问题二 区域权重表

区域	$\overline{\eta_{sbi}}$	$\overline{\eta_{\cos i}}$	J_i	P_i
I1	0.9357647	0.660224	0.617814407	0.086183
I2	0.9384723	0.843482	0.791584586	0.110424
I3	0.9418835	0.949242	0.894075189	0.124721
I4	0.9321777	0.84346	0.786254137	0.10968
I5	0.9315874	0.660879	0.615666922	0.085884
I6	0.9375644	0.842925	0.790296378	0.110244
I7	0.9642055	0.687668	0.663053461	0.092494
I8	0.9901042	0.578509	0.572783993	0.079902
I9	0.9548581	0.687525	0.656488529	0.091578
I10	0.9268727	0.842185	0.780598563	0.108891

5.2.2 镜面面积优化模型的建立与求解

第二问需要在定日镜场年平均输出热功率固定情况下，优化吸收塔及定日镜各参数，实现单位镜面面积年平均输出热功率最大化，根据所给数据及决策变量，可以列出如下方程：

$$\max\ g(x)=\frac{6\times10^4}{x}$$

$$s.t.\begin{cases} xf(l,w)=6\times10^4 \\ 1\leqslant v\leq 4 \\ 2\leq h\leq 7 \\ \cos\left(\dfrac{197\pi-180\arctan\left(\frac{80-h}{100}\right)}{360}\right)<h \end{cases} \tag{21}$$

其中x代表镜面数量，由于此时目标函数中决策变量仅有x，且位于分母不便于计算，我们可以将其进行逻辑运算转化，得：

$$\max\ f(l,w)=2lw\sum_{i=1}^{10}\overline{DNI_i}P_i\overline{\eta_i}$$

$$s.t.\quad\begin{cases}1\le l\le 4\\2\le h\le 6\\2\le w\le 7\\\cos\left(\dfrac{197\pi-180\arctan\left(\frac{80-h}{100}\right)}{360}\right)<h\end{cases}\tag{22}$$

其中 $\overline{DNI_i}$、P_i、$\overline{\eta_i}$ 均为常数，目标函数中的决策变量只有 l、w，l 为镜面半高，h 为安装高度，w 为镜面边长。当 $w>7$ 时，由于镜面边长大于集热器直径，占用地面更多面积的同时，会产生较多的锥形损耗，导致集热器截断效率降低，不利于本题规划参数求最优解，故只考虑 $w\leqslant 7$ 的情况。

由于上述方程内不是所有的变量均为线性，所以我们利用 Matlab 中的 fmincon 函数进行非线性约束规划。

fmincon 是 Matlab 中的一个优化函数，用于解决具有限制条件的非线性连续优化问题，支持等式和不等式同时约束及局部和全局双向优化。针对本题，方程(20)中约束既有等式也有不等式，模型中优化参数需要局部观察和全局优化，故可以选择 fmincon 函数进行非线性规划处理。

经 Matlab 处理，绘制出非线性规划拟合图像，如图 11 所示，求得最大值时对应解 $w=7,l=4$。

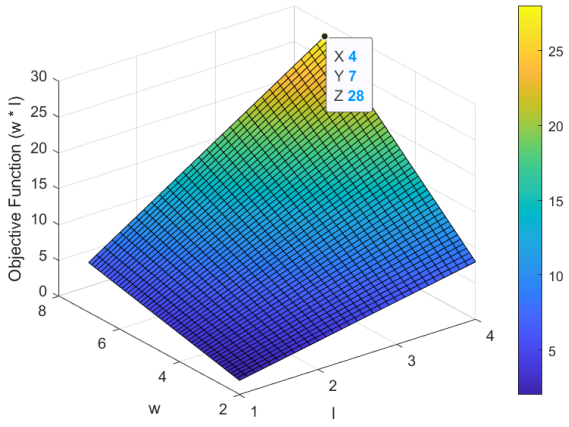


图 11：非线性规划拟合图像

5. 2. 3 区域规划分布

根据上一小节所得镜面面积最优解 $S=lw=28m^2$，对于方程组：

$$\begin{cases} xf(l,w) = 6\times10^4 \\ f(l,w) = 2lw\sum_{i=1}^{10}\overline{DNI_i}P_i\overline{\eta_i} \end{cases} \tag{23}$$

可求解得镜面数量$x=3300$，根据每个区域的权值，计算得每个区域内定日镜的数量，如表 4 所示。

表 4：问题二 各区域内定日镜分布数量

区域	L1	L2	L3	L4	L5	L6	L7	L8	L9	L10	sum
分布数量	284	363	410	361	283	363	304	263	301	358	3290

在每一个扇形区域内，根据所求的定日镜分布数量，利用 C 语言，保证符合镜场规划的同时，实现将定日镜从中心向外沿逐层均匀排列，获得了所求所有点的坐标。

根据所得坐标，利用 Matlab 作出定日镜散点分布图，如图 12 所示。

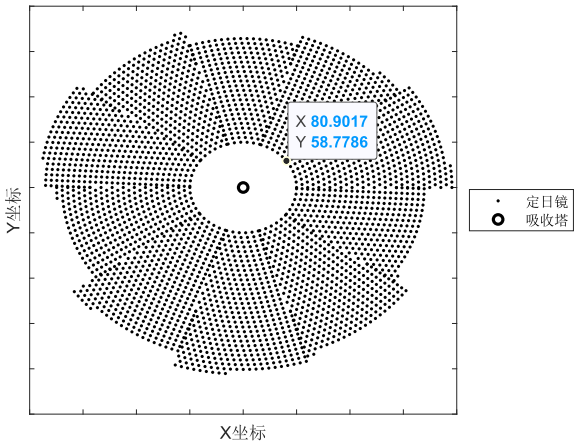


图 12：问题 2 定日镜分布

在得到每个点坐标后，将新的坐标数据读入程序，重新调用问题一所用的函数求解各效率及功率，检查是否符合额定功率的同时，比较各拟合镜场的单位镜面面积年平均输出热功率，在重复数量足够多后选择值最大的一组数据作为最优解。

5.2.4 问题二求解结果

表 5：问题二每月 21 日平均光学效率及输出功率

日期	平均光学效率	平均余弦效率	平均阴影遮挡效率	平均截断效率	单位面积镜面平均输出热功率（Kw/m2）
1 月 21 日	0.296114	0.731515	0.67943	0.709308	0.258717
2 月 21 日	0.324863	0.748859	0.715134	0.709308	0.30677
3 月 21 日	0.342575	0.765903	0.727825	0.709308	0.340931
4 月 21 日	0.355198	0.780134	0.733447	0.709308	0.365557

5 月 21 日	0.36027	0.78726	0.73347	0.709308	0.37642
6 月 21 日	0.36153	0.789226	0.733026	0.709308	0.379339
7 月 21 日	0.360205	0.78719	0.733442	0.709308	0.376296
8 月 21 日	0.354771	0.779607	0.733324	0.709308	0.3647
9 月 21 日	0.341781	0.765057	0.727342	0.709308	0.339386
10 月 21 日	0.321723	0.746675	0.711743	0.709308	0.301335
11 月 21 日	0.293863	0.730029	0.676563	0.709308	0.254597
12 月 21 日	0.28233	0.723936	0.659256	0.709308	0.235567

表 6：问题二 年平均光学效率及输出效率

年平均光学效率	年平均余弦效率	年平均阴影遮挡效率	年平均截断效率	年平均输出热功率（MW）	单位面积镜面年平均输出热功率（kW/m2）
0.33293525	0.761282583	0.713666833	0.709308	60	0.324967917

5.3 问题三模型的建立与求解

5.3.1 模型分析

问题三的模型与问题二模型基本相同，只是在规划设计中引入了定日镜尺寸和高度的决策变量。采用改进过的启发式算法固然可以得到相对优解，但其所要考虑的参数过多，建立的方程过于繁琐，求解代码的时间复杂度较高。

所以可以换一个角度，由于定日镜尺寸和高度可以自由设置，其空间安排得到了最大化利用。在一片排放紧密科学的定日镜场中，通过调整每一个定日镜的尺寸和高度，可以使得阴影遮挡效率影响降到最低。在此优化模型内，不妨就忽略阴影遮挡效率所带来的影响，将权值判定因子定为每一区域内的余弦效率。

$$\begin{cases} J_i = \overline{\eta_{\cos i}} \\ P_i = \dfrac{J_i}{\sum_{i=1}^{10} J_i} \end{cases} \tag{24}$$

5.3.2 随机模拟镜场分布与规划权值

通过大量模拟随机分布一定特征的镜场对象（提前预制镜面全大、全小或大小成一定比例的镜场），根据公式(23)得到每一块区域的平均权值判断因子，进而求得十个区域的权值分布。由于在非线性规划中，定日镜的尺寸及高度不是所需要的精确解，方便模型计算，我们可以继续仍用 $4m \times 7m$ 的尺寸模型，按问题二的模型解法，求出镜场内定日镜数目 $x = 2307$,根据权重得出各区域定日镜分布数量，如表 7 所示。

表 7：问题三 区域权重表

区域	J_i	P_i	分布数量
I1	0.660224	0.086916	201
I2	0.843482	0.111041	256
I3	0.949242	0.124964	288
I4	0.84346	0.111039	256
I5	0.660879	0.087002	201
I6	0.842925	0.110968	256
I7	0.687668	0.090529	209
I8	0.578509	0.076159	176
I9	0.687525	0.09051	209
I10	0.842185	0.110871	256

5.3.3 区域规划分布

根据以上得出的不同区域内定日镜分布数量，可以至少得出一个定日镜尺寸固定，满足额定功率的镜场分布方案。在对应的区域内重复问题二中的规划分布操作，实现将定日镜从中心向外延一圈圈均匀高效排列，根据 C 语言求得相应的点坐标，导入 Matlab 中可作出相应的散点分布，如图 13 所示。

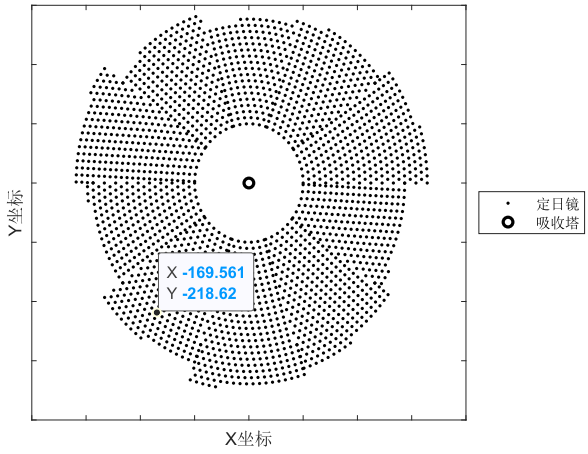


图 13：问题 3 定日镜分布图

由于不考虑阴影遮挡效率，空间效能已然达到最大的情况下，定日镜面积越大，年平均光学效率和年平均输出热功率也就最大。面积增大的对单位面积镜面平均输出热功率的影响不及输出热功率的影响快，所以，在模型宽松限制下，可以将此坐标分布视为本题最优解。

5.3.4 问题三求解结果

表 8：问题三 每月 21 日平均光学效率及输出功率

日期	平均光学效率	平均余弦效率	平均阴影遮挡效率	平均截断效率	单位面积镜面平均输出热功率（Kw/m2）
1 月 21 日	0.464789	0.740908	1	0.709308	0.404507
2 月 21 日	0.476049	0.759006	1	0.709308	0.448814

3 月 21 日	0.487036	0.776678	1	0.709308	0.48446
4 月 21 日	0.496123	0.791317	1	0.709308	0.51057
5 月 21 日	0.500609	0.79857	1	0.709308	0.52305
6 月 21 日	0.501827	0.800547	1	0.709308	0.526547
7 月 21 日	0.500566	0.798499	1	0.709308	0.522926
8 月 21 日	0.495788	0.790777	1	0.709308	0.50964
9 月 21 日	0.486493	0.775803	1	0.709308	0.482821
10 月 21 日	0.474636	0.756734	1	0.709308	0.443765
11 月 21 日	0.463821	0.739352	1	0.709308	0.400188
12 月 21 日	0.459845	0.732963	1	0.709308	0.38141

表 9：问题三 年平均光学效率及输出效率

年平均光学效率	年平均余弦效率	年平均阴影遮挡效率	年平均截断效率	年平均输出热功率（MW）	单位面积镜面年平均输出热功率（kW/m2）
0.483965167	0.771762833	1	0.709308	60	0.4698915

六、 模型的评价、改进

6.1 模型的优点

1. 三问所用模型基本原创，创新性强。
2. 部分几何模型采用计算机离散化思想处理，准确性较高。
3. 代码函数封装性好，调用功能快捷方便。
4. 问题二、三采用的随机分布做法比启发式算法更为便捷，思路也更为明确。

6.2 模型的缺点

1. 第一问求解虽然精确但过程较为繁琐，还有简化空间。
2. 第二三两问虽用分区加权简化了计算和思维过程，但却使求解精度降低。

6.3 模型的改进

问题三中使用问题二的非线性规划结果来计算最大产能的过程不准确，可以利用该结果求得符合规划要求的方案后，利用算法动态调整定日镜的尺寸和高度，寻找是否有比之产能更高解。

七、 参考文献

[1]谢飞. 塔式太阳能热电系统定日镜场光学仿真与应用研究[D].浙江大学,2013.

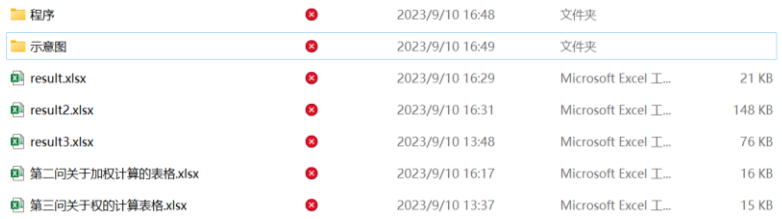
附录

（由于最终计算结果数目太多，将计算结果统一放在支撑材料中）

附录 1

介绍：支撑材料的文件列表

表格文件及计算结果

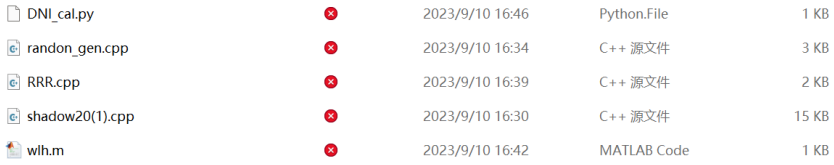


程序	2023/9/10 16:48	文件夹	
示意图	2023/9/10 16:49	文件夹	
result.xlsx	2023/9/10 16:29	Microsoft Excel 工...	21 KB
result2.xlsx	2023/9/10 16:31	Microsoft Excel 工...	148 KB
result3.xlsx	2023/9/10 13:48	Microsoft Excel 工...	76 KB
第二问关于加权计算的表格.xlsx	2023/9/10 16:17	Microsoft Excel 工...	16 KB
第三问关于权的计算表格.xlsx	2023/9/10 13:37	Microsoft Excel 工...	15 KB

图片文件

untitled.svg	2023/9/10 13:23	Microsoft Edge HT...	934 KB
untitled1.svg	2023/9/10 13:28	Microsoft Edge HT...	658 KB
定日镜分布图(1).svg	2023/9/7 22:28	Microsoft Edge HT...	531 KB
非线性规划拟合图.svg	2023/9/10 16:46	Microsoft Edge HT...	136 KB
思路示意图(1).svg	2023/9/8 14:08	Microsoft Edge HT...	54 KB
问题二定日镜分布图.svg	2023/9/10 13:23	Microsoft Edge HT...	934 KB
问题三定日镜分布图.svg	2023/9/10 13:28	Microsoft Edge HT...	658 KB

程序文件



DNI_cal.py	2023/9/10 16:46	Python.File	1 KB
randon_gen.cpp	2023/9/10 16:34	C++ 源文件	3 KB
RRR.cpp	2023/9/10 16:39	C++ 源文件	2 KB
shadow20(1).cpp	2023/9/10 16:30	C++ 源文件	15 KB
wlh.m	2023/9/10 16:42	MATLAB Code	1 KB

附录 2

介绍：该 C 语言程序用于求解各种效率

```
#include<stdio.h>
#include<iostream>
#include<cmath>
#include<cstring>
using namespace std;
#define LL long long
const double G0=1.366;//kW/m2
double H0=3;//km
const int range=2000;
const int N=2290;
double AAA=0.4237-0.00821*(6-H0)*(6-H0);
double BBB=0.5055+0.00595*(6.5-H0)*(6.5-H0);
double CCC=0.2711+0.01858*(2.5-H0)*(2.5-H0);
int MAP[range+100][range+100];
```

```
double ans[12][5][N+10];
double ans2[12][5][N+10];
double ans3[12][5][N+10];
double ans4[12][5][N+10];
struct vec3{      //结构体3维向量
    double x,y,z;
    void fang(){
        x=-x;
        y=-y;
        z=-z;
    }
    void refine(int id){
        double mo=sqrt(x*x+y*y+z*z);
        if((id==1&&x<0)||(id==2&&y<0)||(id==3&&z<0))this->fang();
        x/=mo;
        y/=mo;
        z/=mo;
    }
    void efine(){
        double mo=sqrt(x*x+y*y+z*z);
        x/=mo;
        y/=mo;
        z/=mo;
    }
    double operator*(vec3 other)const{
        return x*other.x+y*other.y+z*other.z;
    }
    vec3 operator+(vec3 other)const{
        vec3 res;
        res.x=x+other.x;
        res.y=y+other.y;
        res.z=z+other.z;
        return res;
    }
    vec3 operator-(vec3 other)const{
        vec3 res;
        res.x=x-other.x;
        res.y=y-other.y;
        res.z=z-other.z;
        return res;
    }
    vec3 operator*(double k)const{
        vec3 res;
        res.x=x*k;
        res.y=y*k;
        res.z=z*k;
        return res;
    }
};
struct vec2{   //结构体向量2维
    double x,y;
```

```
    void fang(){
        x=-x;
        y=-y;
    }
    void refine(int id){
        double mo=sqrt(x*x+y*y);
        if((id==1&&x<0)||(id==2&&y<0))this->fang();
        x/=mo;
        y/=mo;
    }
    void efine(){
        double mo=sqrt(x*x+y*y);
        x/=mo;
        y/=mo;
    }
    double operator*(vec2 other)const{
        return x*other.x+y*other.y;
    }
};
const double ref=0.92;
int TPX[5];
int TPY[5];
int Grid[100][100];  //离散化网格
double xx[N+10],yy[N+10],hh[N+10];
double DD[12]={-59,-28,0,31,61,92,122,153,184,214,245,275};
double SSTT[5]={9,10.5,12,13.5,15};

const double L=4;
const double fi=39.4;
const double W=3.5;
double dX[5],dY[5];
double revv(double a){
    return sqrt(1-a*a);
}
 double mysin(double x){
     return sin(x/180*M_PI);
 }
 double mycos(double x){
     return cos(x/180*M_PI);
 }
 //角度制sin,cos
 void asys(double ST,double D,double &as,double &ys){   //根据日期求太阳高度角和方位角
     double w=(M_PI/12)*(ST-12);
     double siq=sin(2*M_PI*D/365)*sin(2*M_PI*23.45/360);
     double q=asin(siq);
     as=asin(revv(siq)*mycos(fi)*cos(w)+siq*mysin(fi));
     double yky=(sin(q)-sin(as)*mysin(fi))/(cos(as)*mycos(fi));
     if(yky>1){
         ys=0;
         return;
```

```
116.      }
117.      if(yky<-1){
118.          ys=M_PI;
119.          return;
120.      }
121.      ys=acos((sin(q)-sin(as)*mysin(fi))/(cos(as)*mycos(fi)));
122.      if(ST>12)ys=2*M_PI-ys;
123.  }
124.  double atm(vec3 goal){    //求大气透射率
125.      double ddx=goal.x;
126.      double ddy=goal.y;
127.      double ddz=goal.z-80;
128.      double hr=sqrt(ddx*ddx+ddy*ddy+ddz*ddz);
129.      return 0.99321-0.0001176*hr+1.97*(1e-8)*hr*hr;
130.  }
131.  double err(vec3 goal,double a,double o){    //求余弦效率
132.      double x=goal.x;
133.      double y=goal.y;
134.      double h=goal.z;
135.      double zz=(1.0/2)*acos((x*sin(o)*cos(a)+y*cos(o)*cos(a)-(h-
    80)*sin(a))/sqrt(x*x+y*y+(h-80)*(h-80)));
136.      return cos(zz);
137.  }
138.  double getDNI(double as){     //求DNI
139.      return G0*(AAA+BBB*exp(-CCC/sin(as)));
140.  }
141.  void refine(double& ax,double& ay,double& az){  //一系列单位化
142.      if(az<0){ax=-ax,ay=-ay,az=-az;}
143.      double mo=sqrt(ax*ax+ay*ay+az*az);
144.      ax=ax/mo;
145.      ay=ay/mo;
146.      az=az/mo;
147.  }
148.  void refine(double& ax,double& ay){
149.      if(ax<0){ax=-ax;ay=-ay;}
150.      double mo=sqrt(ax*ax+ay*ay);
151.      ax=ax/mo;
152.      ay=ay/mo;
153.  }
154.  void efine(double& ax,double& ay){
155.      double mo=sqrt(ax*ax+ay*ay);
156.      ax=ax/mo;
157.      ay=ay/mo;
158.  }
159.
160.  void geti(double x,double y,double h,double a,double o,vec3& I,vec3&
    A,vec3& B){    //根据位置,太阳角度,得到法向量,阳光射入方向,阳光射出方向
161.      A.x=cos(a)*sin(o);
162.      A.y=cos(a)*cos(o);
163.      A.z=-sin(a);
164.      A.refine(3);
```

```
165.      B.x=x;
166.      B.y=y;
167.      B.z=h-80;
168.      B.refine(3);
169.      I=A+B;
170.      I.refine(3);
171.  }
172.  void mod1(vec3& K,vec3 I,double l,int fg){  //由中心点得到4 个顶点步骤
      1
173.      double moo=sqrt(I.x*I.x+I.y*I.y);
174.      K.x+=fg*I.z*I.x*l/moo;
175.      K.y+=fg*I.z*I.y*l/moo;
176.      K.z-=fg*moo*l;
177.  }
178.  void mod2(vec3& K,vec3 I,double w,int fg){   //由中心点得到4 个顶点步骤
      2
179.      double moo= sqrt(I.x*I.x+I.y*I.y);
180.      K.x+=fg*w*(-I.y)/moo;
181.      K.y+=fg*w*I.x/moo;
182.  }
183.  void get4(double x,double y,double h,double l,double w,vec3 I,vec3 d
      ot[5]){   //根据镜面中心点求4 个顶点
184.      if(I.x==0&&I.y==0){
185.          dot[1].x=x+w;
186.          dot[1].y=y+l;
187.          dot[2].x=x-w;
188.          dot[2].y=y+l;
189.          dot[3].x=x-w;
190.          dot[3].y=y-l;
191.          dot[4].x=x+w;
192.          dot[4].y=y-l;
193.          for(int i=1;i<5;i++){
194.              dot[i].z=h;
195.          }
196.          return;
197.      }
198.      int num=0;
199.      vec3 tp={x,y,h};
200.      for(int i=1;i>=-1;i-=2){
201.          mod1(tp,I,l,i);
202.          for(int j=1;j>=-1;j-=2){
203.              mod2(tp,I,w,j);
204.              dot[++num]=tp;
205.              mod2(tp,I,w,-j);
206.          }
207.          mod1(tp,I,l,-i);
208.      }
209.      vec3 temp=dot[3];
210.      dot[3]=dot[4];
211.      dot[4]=temp;
212.  }
```

```
213.  vec2 yinshe(vec3 goal,vec3 yuan,vec3 I,vec3 zhao){      //计算某个空间中的点的投影在某个平面内的二维坐标表示
214.      vec2 res;
215.      if(I.x==0&&I.y==0){
216.          res.x=goal.x-(goal.z-yuan.z)*zhao.x/zhao.z;
217.          res.y=goal.y-(goal.z-yuan.z)*zhao.y/zhao.z;
218.          return res;
219.      }
220.      vec3 duan=goal-yuan;
221.      double k=(duan*I)/(zhao*I);
222.      duan=duan-zhao*k;
223.      double moo=I.x*I.x+I.y*I.y;
224.      vec3 ex={-I.y,I.x,0};
225.      vec3 ey={I.x*I.z,I.y*I.z,-(moo)};
226.      ex.refine(1);
227.      ey.refine(3);
228.      res.x=duan*ex;
229.      res.y=duan*ey;
230.      return res;
231.  }
232.  int shunni(vec2 q1,vec2 q2,vec2 q3){   //判断p1,p2,p3三点位置是否顺时针排列
233.      double x2=q2.x-q1.x;
234.      double y2=q2.y-q1.y;
235.      double x3=q3.x-q1.x;
236.      double y3=q3.y-q1.y;
237.      double fin=x2*y3-x3*y2;
238.      if(fin>0)return 1;
239.      if(fin<0)return -1;
240.      return 0;
241.  }
242.  bool isnei(vec2 t,vec2 p1,vec2 p2,vec2 p3,vec2 p4){   //判断t是否在凸形p1p2p3p4中
243.      if(shunni(t,p1,p2)==-shunni(t,p3,p4))return false;
244.      if(shunni(t,p4,p1)!=shunni(t,p2,p3))return false;
245.      return true;
246.  }
247.  void draw(vec2 DOT[5],double l,double w){    //将二维平面的4个点对应的凸形的区域标记为影子区域
248.      int Xm=20*w,XM=0,Ym=20*l,YM=0;
249.      for(int i=1;i<5;i++){
250.          TPX[i]=int((DOT[i].x+w)*10);
251.          TPY[i]=int((DOT[i].y+l)*10);
252.          Xm=min(Xm,TPX[i]);
253.          XM=max(XM,TPX[i]);
254.          Ym=min(Ym,TPY[i]);
255.          YM=max(YM,TPY[i]);
256.      }
257.      Xm=max(0,Xm);
258.      Ym=max(0,Ym);
259.      XM=min(XM,int(20*w));
```

```
260.      YM=min(YM,int(20*l));
261.      vec2 now;
262.      vec2 p1,p2,p3,p4;
263.      p1.x=TPX[1];
264.      p1.y=TPY[1];
265.      p2.x=TPX[2];
266.      p2.y=TPY[2];
267.      p3.x=TPX[3];
268.      p3.y=TPY[3];
269.      p4.x=TPX[4];
270.      p4.y=TPY[4];
271.      for(int i=Xm;i<XM;i++){
272.          for(int j=Ym;j<YM;j++){
273.              now.x=i;
274.              now.y=j;
275.              if(isnei(now,p1,p2,p3,p4)){
276.                  Grid[i][j]=1;
277.              }
278.          }
279.      }
280.  }
281.  const double WW=3.5;
282.  const double LK=4;
283.  vec3 PPP[5];
284.  vec2 POI[5];
285.  LL count(double l,double w){      //计算阴影效率时需要数影子格点数
286.      LL res=0;
287.      for(int xi=0;xi<=20*w;xi++){
288.          for(int yi=0;yi<=20*l;yi++){
289.              if(Grid[xi][yi]){
290.                  res++;
291.              }
292.          }
293.      }
294.      return res;
295.  }
296.  void yuxian(){   //计算平均余弦效率
297.      for(int i=0;i<12;i++){
298.          double sum=0;
299.          for(int j=0;j<5;j++){
300.              double sum2=0;
301.              for(int k=1;k<=N;k++){
302.                  sum2+=ans2[i][j][k];
303.              }
304.              sum2/=N;
305.              sum+=sum2;
306.          }
307.          sum/=5;
308.          printf("%d--%lf\n",i+1,sum);
309.      }
310.  }
```

```
311.  void yingzi(){    //计算平均阴影遮挡效率
312.      for(int i=0;i<12;i++){
313.          double sum=0;
314.          for(int j=0;j<5;j++){
315.              double sum2=0;
316.              for(int k=1;k<=N;k++){
317.                  sum2+=ans[i][j][k];
318.              }
319.              sum2/=N;
320.              sum+=sum2;
321.          }
322.          sum/=5;
323.          printf("%d--%lf\n",i+1,sum);
324.      }
325.  }
326.  void toushe(){   //结算平均大气透射率
327.      for(int i=0;i<12;i++){
328.          double sum=0;
329.          for(int j=0;j<5;j++){
330.              double sum2=0;
331.              for(int k=1;k<=N;k++){
332.                  sum2+=ans3[i][j][k];
333.              }
334.              sum2/=N;
335.              sum+=sum2;
336.          }
337.          sum/=5;
338.          printf("%d--%lf\n",i+1,sum);
339.      }
340.  }
341.  void trunn(){   //计算平均截断效率
342.      for(int i=0;i<12;i++){
343.          double sum=0;
344.          for(int j=0;j<5;j++){
345.              double sum2=0;
346.              for(int k=1;k<=N;k++){
347.                  sum2+=ans4[i][j][k];
348.              }
349.              sum2/=N;
350.              sum+=sum2;
351.          }
352.          sum/=5;
353.          printf("%d--%lf\n",i+1,sum);
354.      }
355.  }
356.  void Light(){   //计算平均光学效率
357.      for(int i=0;i<12;i++){
358.          double sum=0;
359.          for(int j=0;j<5;j++){
360.              double sum2=0;
361.              for(int k=1;k<=N;k++){
```

```
                    sum2+=ans2[i][j][k]*ans3[i][j][k]*ans4[i][j][k]*ref;
                }
                sum2/=N;
                sum+=sum2;
            }
            sum/=5;
            printf("%d--%lf\n",i+1,sum);
        }
    }
    void TOTT(){
        double tg1=0,tg2=0,tg3=0,tg4=0;
        for(int i=0;i<12;i++){
            for(int j=0;j<5;j++){
                for(int u=1;u<=N;u++){
                    tg1+=ans[i][j][u];
                    tg2+=ans2[i][j][u];
                    tg3+=ans3[i][j][u];
                    tg4+=ans4[i][j][u];
                }
            }
        }
        tg1/=60*N;
        tg2/=60*N;
        tg3/=60*N;
        tg4/=60*N;
        printf("%lf",0.98*tg1*tg2*tg3*tg4*ref*56*N);
    }
    double shuchu[12][5];
    bool checkk(int k,double ys,int fg){
        double x1=cos(ys+M_PI/2);
        double y1=sin(ys+M_PI/2);
        double x2=xx[k];
        double y2=yy[k];
        efine(x2,y2);
        return fg*(x1*x2+y1*y2)>cos(M_PI/10);
    }
    double eva1[30];
    double eva2[30];
    int main(){
        freopen("win-1","w",stdout);
        for(int i=1;i<=N;i++){
            cin>>xx[i]>>yy[i];
            MAP[int(xx[i]+range/2)][int(yy[i]+range/2)]=i;
            hh[i]=4;
        }
        double SAM=0;
        for(int i=0;i<12;i++){
            double sum=0;
            for(int j=0;j<5;j++){
                double as,ys;
                asys(SSTT[j],DD[i],as,ys);
```

```
413.                //printf("%d-%d-DNI:%lf\n",i,j,getDNI(as));
414.                double ttr=ys-M_PI*3/2;
415.                vec2 n1={3.5*sin(ttr),-3.5*cos(ttr)};
416.                vec2 n2={-3.5*sin(ttr),3.5*cos(ttr)};
417.                vec2 n3={n2.x+84*cos(ttr)/tan(as),n2.y+84*sin(ttr)/tan(a
    s)};
418.                vec2 n4={n1.x+84*cos(ttr)/tan(as),n1.y+84*sin(ttr)/tan(a
    s)};
419.                double evg=0;
420.                for(int k=1;k<=N;k++){
421.                    vec3 yuan={xx[k],yy[k],hh[k]};
422.                    memset(Grid,0, sizeof Grid);
423.                    vec2 p1,p2,p3,p4;
424.                    vec3 rr[2],II;
425.                    geti(xx[k],yy[k],hh[k],as,ys,II,rr[0],rr[1]);
426.                    for(int u=0;u<2;u++){
427.                        double mo= sqrt(rr[u].x*rr[u].x+rr[u].y*rr[u].y)
    ;
428.                        p1.x=xx[k]+WW*rr[u].y/mo;
429.                        p1.y=yy[k]-WW*rr[u].x/mo;
430.                        p4.x=xx[k]-WW*rr[u].y/mo;
431.                        p4.y=yy[k]+WW*rr[u].x/mo;
432.                        p2.x=p1.x+2*LK*rr[u].x/(rr[u].z*mo);
433.                        p2.y=p1.y+2*LK*rr[u].y/(rr[u].z*mo);
434.                        p3.x=p4.x+2*LK*rr[u].x/(rr[u].z*mo);
435.                        p3.y=p4.y+2*LK*rr[u].y/(rr[u].z)*mo;
436.                        p1.x=int(p1.x+range/2);
437.                        p1.y=int(p1.y+range/2);
438.                        p2.x=int(p2.x+range/2);
439.                        p2.y=int(p2.y+range/2);
440.                        p3.x=int(p3.x+range/2);
441.                        p3.y=int(p3.y+range/2);
442.                        p4.x=int(p4.x+range/2);
443.                        p4.y=int(p4.y+range/2);
444.                        int Xm=min(min(p1.x,p2.x),min(p3.x,p4.x));
445.                        int XM=max(max(p1.x,p2.x),max(p3.x,p4.x));
446.                        int Ym=min(min(p1.y,p2.y),min(p3.y,p4.y));
447.                        int YM=max(max(p1.y,p2.y),max(p3.y,p4.y));
448.                        for(int xi=Xm;xi<=XM;xi++){
449.                            for(int yi=Ym;yi<=YM;yi++){
450.                                vec2 ttt={(double)xi,(double)yi};
451.                                if(MAP[xi][yi]&&MAP[xi][yi]!=k&&isnei(tt
    t,p1,p2,p3,p4)){
452.                                    int idx=MAP[xi][yi];
453.                                    vec3 rta,tta,Ita;
454.                                    geti(xx[idx],yy[idx],hh[idx],as,ys,I
    ta,rta,tta);
455.                                    get4(xx[idx],yy[idx],hh[idx],LK,WW,I
    ta,PPP);
456.                                    for(int ii=1;ii<=4;ii++){
```

```
457.                                     POI[ii]=yinshe(PPP[ii],yuan,II,r
   r[u]);
458.                                 }
459.                                 draw(POI,LK,WW);
460.                             }
461.                         }
462.                     }
463.                 }
464.                 double numm=(double)count(LK,WW);
465.                 ans[i][j][k]=1-(numm/(LK*WW*400));
466.                 if(isnei({xx[k],yy[k]},n1,n2,n3,n4)){ans[i][j][k]=0;
   }
467.                 ans2[i][j][k]=err({xx[k],yy[k],hh[k]},as,ys);
468.                 ans3[i][j][k]=atm({xx[k],yy[k],hh[k]});
469.                 if(1-rr[1].z*rr[1].z<0){
470.                     ans4[i][j][k]=0;
471.                 }else{
472.                     ans4[i][j][k]=M_PI*ans3[i][j][k]*0.997*sqrt(1-
   rr[1].z*rr[1].z)/4;
473.                 }
474.                 double totot=ans2[i][j][k]*ans3[i][j][k]*ans4[i][j][
   k]*ref;
475.                 double DNI=getDNI(as);
476.                 evg+=DNI*totot;
477.             }
478.             evg/=N;
479.             sum+=evg;
480.         }
481.         sum/=5;
482.         SAM+=sum;
483.         printf("%d,%lf\n",i+1,sum);//打印月平均每平方米镜面发热功率
484.     }
485.     //printf("SAM:%lf\n",SAM*7*8*N/12);
486.     for(int i=0;i<12;i++){
487.         for(int j=0;j<5;j++){
488.             double as,ys;
489.             asys(SSTT[j],DD[i],as,ys);
490.             eva1[j]+=ys;
491.             eva2[j]+=ys-M_PI;
492.         }
493.     }
494.     //yuxian();
495.     //yingzi();
496.     //Light();
497.    // trunn();
498.    //toushe();
499.    //TOTT();
500.    //yuxian();
501.     fclose(stdout);
502.     return 0;
```

附录 3

介绍：该 C 语言程序用于根据各区域定日镜数目随机生成定日镜坐标

```
#include<time.h>
#include<stdio.h>
#include<stdlib.h>
#include<cmath>
#include<iostream>
#include <random>
using namespace std;
const double rr=6;
struct P{
    double x,y;
}PP[8000];
int idx;
const int TOTOT=2290;   //镜子总个数
double yyyy=M_PI;
double dis(P a,P b){
    return (a.x-b.x)*(a.x-b.x)+(a.y-b.y)*(a.y-b.y);
}
double sum2=0;
double sum3=0;
double num3[10]={197     //q2 各区权重
        ,252
        ,284
        ,252
        ,197
        ,252
        ,205
        ,173
        ,205
        ,252
};
double num2[10]={208,266,300,264,207,266,223,192,221,262};//q3 各区权重
void init(){
    for(int i=0;i<10;i++) {
        sum2+=num2[i];
        sum3+=num3[i];
    }
}
double ysy[10]={M_PI*3/5,M_PI*4/5,M_PI,yyyy+M_PI/5,yyyy+M_PI*2/5,yyyy+M_PI*3/5,yyyy+M_PI*4/5,0,M_PI/5,M_PI*2/5};  //用于表示各区中心轴的方位角
void efine(double& ax,double& ay){
    double mo=sqrt(ax*ax+ay*ay);
    ax=ax/mo;
    ay=ay/mo;
}
bool checkk(double x,double y,double ys){   //检查某个点是否在某个区域内
    double x1=cos(ys+M_PI/2);
    double y1=sin(ys+M_PI/2);
    double x2=x;
```

```
    double y2=y;
    efine(x2,y2);
    return (x1*x2+y1*y2)>=cos(M_PI/10);
}
int alll=0;
int main(){
    init();
    freopen("data.txt","w",stdout);
    int tit;
    for(int i=0;i<10;i++){
        tit=0;
        double r=100;
        double doo=-M_PI/10;
        int ck=1;
        while(tit<num3[i]*TOTOT/sum3){    //确保每个区域的顶点数为对应的分配量
            double oo=doo+ysy[i]+M_PI/2;
            double X=r*cos(oo);
            double Y=r*sin(oo);
            tit++;
            alll++;
            cout<<X<<" "<<Y<<endl;
            if(alll==TOTOT)return 0;
            doo+=2*asin(rr/r);
            if(doo>=M_PI/10){    //即将越出区域时重新回归起始边
                if(ck){
                    ck=0;
                    doo=-M_PI/10+asin(rr/r);
                }else{
                    ck=1;
                    doo=-M_PI/10;
                }
                r+=rr*sqrt(3);  //向外一圈摆放
            }
        }
    }
    fclose(stdout);
    return 0;
   }
```

附录 4
介绍：根据给定数目随机生成点坐标

```
#include<time.h>
#include<stdio.h>
```

```
3. #include<stdlib.h>
4. #include<cmath>
5. #include<iostream>
6. #include <random>
7. using namespace std;
8. const double len=12;
9. const int rage=700;
10.struct P{
11.    double x,y;
12.}PP[8000];
13.int idx;
14.const int TOTOT=2000;
15.double dis(P a,P b){
16.    return (a.x-b.x)*(a.x-b.x)+(a.y-b.y)*(a.y-b.y);
17.}
18.//double ysy[10]={2.103849,2.510487,3.141593,3.772698,4.179337,-
   1.037744,-0.631105,-0.000000,0.631105,1.037744};
19.void efine(double& ax,double& ay){
20.    double mo=sqrt(ax*ax+ay*ay);
21.    ax=ax/mo;
22.    ay=ay/mo;
23.}
24.bool checkk(double x,double y,double ys){
25.    double x1=cos(ys+M_PI/2);
26.    double y1=sin(ys+M_PI/2);
27.    double x2=x;
28.    double y2=y;
29.    efine(x2,y2);
30.    return (x1*x2+y1*y2)>=cos(M_PI/10);
31.}
32.
33.int main(){
34.    freopen("data2.txt","w",stdout);
35.    //srand((unsigned)time(NULL))
36.    // 生成随机浮点数
37.    std::random_device rd;
38.    std::mt19937 gen(rd());
39.
40.    // 指定浮点数范围
41.    std::uniform_real_distribution<double> distribution(-
   350.0, 350.0); // 生成0.0到1.0之间的随机浮点数
42.    int has=0;
43.    do{
44.        double X=distribution(gen);
45.        double Y=distribution(gen);
46.        bool fg= true;
47.        for(int i=1;i<=idx;i++){
48.            if(dis(PP[i],{X,Y})<len*len){
49.                fg= false;
50.                break;
51.            }
```

```
        }
        if(fg){
            PP[++idx]={X,Y};
        }else{
            has++;
        }
    }while(has<1000&&idx<TOTOT);
    for(int i=1;i<=idx;i++){
        printf("%lf %lf\n",PP[i].x,PP[i].y);
    }
    fclose(stdout);
    return 0;
}
```

附录 5

介绍：利用 python 计算 DNI

```
import math

DD = [-59, -28, 0, 31, 61, 92, 122, 153, 184, 214, 245, 275]
SSTT = [9, 10.5, 12, 13.5, 15]

for ST in SSTT:
    for D in DD:
        DNI = 0
        G0 = 1.366
        H = 3
        φ = (39.4 / 180) * math.pi
        ω = (ST - 12) * math.pi / 12
        sin_δ = math.sin(2 * math.pi * D / 365) * math.sin(23.45 * math.pi / 180)
        δ = math.asin(sin_δ)
        sin_αs = math.cos(δ) * math.cos(φ) * math.cos(ω) + math.sin(δ) * math.sin(φ)
        αs = math.asin(sin_αs)
        print(f"ST = {ST}, D = {D}, DNI = {DNI}")
        a = 0.4237 - 0.00821 * (6 - H) ** 2
        b = 0.5055 + 0.00595 * (6.5 - H) ** 2
        c = 0.2711 + 0.01858 * (2.5 - H) ** 2
        DNI = G0 * (a + b * math.exp((-c) / sin_αs))
            #print(f"ST = {ST}, D = {D}, DNI = {DNI}")
```

附录 6

介绍：利用 matlab 求解非线性规划

```
clear all;
clc;
% 定义目标函数
fun = @(x) -x(1)*x(3);

```

```
% 初始点
x0 = [0,1,0];

% 定义非线性约束函数(包括不等式约束和等式约束)
nonlcon = @(x)deal(x(1)*cos((197*pi-180*atan((80-x(2))/100))/360)-x(2), []);

% 设置优化选项
options = optimoptions('fmincon', 'Display', 'iter', 'Algorithm', 'sqp');

% 使用 fmincon 进行非线性约束规划
[x, fval] = fmincon(fun, rand(3,1), [], [], [], [], [1,2,2], [4,6,7], nonlcon, options);
%fmincon(goal,init,A,b,Aeq,beq,lb,ub,nonlcon,options);

fprintf('最优解: x = [%.4f, %.4f, %.4f]\n', x(1), x(2),x(3));
fprintf('最优目标值: %.4f\n', -fval);
```

附录 7

介绍：第二问关于加权计算的表格

权值判定因子	权重 pi	截断效率	ref	光学效率η	DNI	f(l,w)	每区分布数量	定日镜总数
0.617814407	0.086183	0.709308	0.92	0.387949688	0.920667	1.262073	284	3290
0.791584586	0.110424	0.709308	0.92	0.49706674	0.992915	2.23446	363	
0.894075189	0.124721	0.709308	0.92	0.561424574	1.011635	2.904276	410	
0.786254137	0.10968	0.709308	0.92	0.493719543	0.992915	2.204468	361	
0.615666922	0.085884	0.709308	0.92	0.386601198	0.920667	1.253315	283	
0.790296378	0.110244	0.709308	0.92	0.496257824	0.920667	2.065135	363	
0.663053461	0.092494	0.709308	0.92	0.416357049	0.992915	1.567744	304	
0.572783993	0.079902	0.709308	0.92	0.35967334	1.011635	1.191987	263	
0.656488529	0.091578	0.709308	0.92	0.412234673	0.992915	1.536853	301	
0.780598563	0.108891	0.709308	0.92	0.490168189	0.920667	2.014763	358	