基于经验布局和局部搜索方法
解决多变量高维优化模型的定日镜场设计策略

摘要

塔式太阳能光热发电技术作为低碳环保的新型清洁能源技术，对我国绿色能源发展至关重要。作为其核心构件的定日镜，能够将太阳光能高效聚焦到吸热器上，推动发电站的能量转化。因此，高效率定日镜场的设计和布局是十分具有研究潜力的方向。

本研究致力于探讨定日镜场的规划问题，考虑影响其工作的多个关键因素，包括太阳的位置、定日镜之间的间距、镜场的排列方式、定日镜与吸热器的距离，以及吸热器的高度等。可以说，定日镜场的性能和效率受许多复杂因素影响，因此定日镜场的布局需要合理的规划。

对于问题 1：本部分深入分析了光学损失的不同类型，充分利用空间几何思维，建立了定日镜光学损失模型。余弦损失是由于光线反射时的能量损失导致的，其计算依赖于入射光线和法线的夹角。通过几何图示，我们得到了余弦效率的计算公式，考虑了太阳方位角和定日镜方位角等因素。其次，集热器截断损失是由于太阳光线呈锥形入射而导致部分光线无法被集热器吸收而产生的损失。通过太阳张角的计算，本文揭示了入射光线的锥形特性，进一步推导出了集热器截断效率的公式。这个效率描述了集热器能够捕获入射光线的能力，是光学系统效率的一个关键部分。最后，本问研究了阴影遮挡损失，其中，我们对吸收塔阴影损失模型采用三角形相似比和面积比值来估算效率；而在定日镜间阴影遮挡损失计算过程中，我们通过两个大胆而巧妙的假设大大简化了计算。

对于问题 2：对于本问有着高维度和复杂多目标的优化问题求解，我们采取先粗略划分定日镜搜索区域，再做局部搜索的策略。先引入前人研究[1]，做出布局预设，在无遮挡（EB）布局、无阻塞（No blocking-dense）布局和经验型（DELSOL）布局基础上给定初始化参数来提高收敛性能。再引入寻找最优布局序列最小二乘法（Sequential Least Squares Quadratic Programming,SLSQP）作为局部调度优化的方法，提高收敛精度和计算速度。

对于问题 3：问题 3 在问题 2 的基础上，将定日镜尺寸和高度调整为可变条件，使得模型具有更高的维度。为解决该问题，本文采用了类似问题 2 的优化框架，借助经典布局和专业软件 SolarPILOT 改进了初始化方法、搜索算法，并对损耗模型进行简化，以提高计算效率和收敛速度。

本研究提供了塔式太阳能光热发电定日镜场设计和优化的理论基础和方法，为我国绿色能源的可持续发展提供了有力支持。

关键字：多变量优化模型　光学效率　序列最小二乘法

一、 问题重述

1.1 问题背景

在当今全球能源供应日益紧张，环境状况日益恶化的背景下，中国正在积极致力于发展新能源产业，推动绿色能源的发展，以期实现“碳达峰”和“碳中和”的目标。在这一过程中，塔式太阳能光热发电技术作为一种创新型的电力系统技术，发挥了极为重要的作用，为中国的绿色能源事业作出了重要贡献。

塔式太阳能光热发电站（以下简称塔式电站）主要由定日镜及吸收塔组成。其中，定日镜负责将太阳光反射汇聚到安装在吸收塔顶端的集热器中，将太阳能以热能形式储存起来。在一个典型的镜场中，通常需要使用大量的定日镜才能实现能量的有效聚集。[2]在镜场的设计过程中，最关键的考虑因素之一是如何提高整个镜场对太阳光能的利用效率，也就是所谓的镜场光学效率。

镜场的布局方式是影响镜场光学效率的主要因素之一。在进行镜场的规划时，需要综合考虑多个因素，包括太阳的位置、定日镜之间的距离、镜场的排列方式、定日镜与吸热器之间的距离，以及吸热器的高度等。这些因素直接决定了镜场的光学效率。

由于太阳在一年中的不同时间段位于不同位置，也就是太阳相对于镜场内各个定日镜的位置会随着时间的推移而变化。但是各个定日镜反射光线的目标点是固定的。此外，组成镜场的多个定日镜在地面位置和反射目标点上都有所不同，这意味着即使在相同时间，每台定日镜的反射面姿态和光学效率也会不同。另外，太阳光线并非平行，而是具有一定锥角的锥形光线，这增加了高效聚焦的难度，尤其是当定日镜距离反射目标点更远时，挑战会变得更大。在这种情况下，如果将定日镜排列得过于密集以减小定日镜与目标点之间的距离，可能会增加定日镜之间的相互干扰，从而导致可接收光线的浪费。但另一方面，如果为了减小定日镜之间的相互干扰，增加彼此之间的距离，可能会导致土地的浪费。并且定日镜与吸热器之间更远的距离，也对定日镜的跟踪精度提出了更高的要求。

因而如何设计定日镜场，使更多的太阳光线能够被集热器吸收对于塔式电站来说十分重要。

1.2 问题提出

根据以上背景，需要解决以下问题：

问题 1：已知吸收塔置于圆形定日镜场中央，定日镜尺寸 $6m \times 6m$，高度 $4m$，所有定日镜位置。计算该定日镜场的年平均光学效率、年平均输出热功率以及单位镜面面积年平均输出热功率。

问题 2：在所有定日镜尺寸和安装高度都相同的情况下，设计吸收塔的位置坐标、定日镜尺寸、安装高度、定日镜数量和定日镜位置，以使得定日镜场满足额定年平均输出热功率 $60MW$ 的条件下，单位镜面面积年平均输出热功率最大化。

问题 3：在允许定日镜尺寸和安装高度不同的情况下，设计吸收塔的位置坐标、定日镜尺寸、安装高度、定日镜数量和定日镜位置，以使得定日镜场满足额定年平均输出热功率 $60MW$ 的条件下，单位镜面面积年平均输出热功率最大化。

二、 问题分析

本文要解决的是塔式太阳能光热发电站中定日镜场系统的优化问题。问题 1 要求在给定吸收塔位置、定日镜位置参数及尺寸参数等条件下，求解年均光学效率和输出热功率等数据。问题 2、三要求在第一问建立的损耗模型的基础上，考虑如何安排定日镜场中的参数，得到在满足达到额定热功率的条件下最优的方案。

2.1 对问题 1 的分析

针对问题 1 要求的定光镜场年平均光学效率，其核心在于计算各种光学损耗的几何模型的建立。我们通过单个定光镜的光学效率公式得知，需要分别去分析其阴影遮挡效率、余弦效率、大气透射率、集热器截断效率以及镜面反射率的计算公式。为此，我们建立了余弦损失模型，集热器截断损失模型，以及包括镜面阴影遮挡和塔阴影遮挡的阴影遮挡损失模型来计算定光镜的光学效率。

通过定光镜场年光学效率的求解，我们可以通过相应公式来得出年均输出热功率和单位镜面年均输出热功率。

2.2 对问题 2 的分析

问题 2 实质上是在问题 1 建立的计算损耗的几何模型的基础上的参数优化问题。其要求我们在问题 1 计算损耗方式的基础上，以最大化单位镜面的年均热输出功率为目标，考虑额定功率、场地面积等约束，对吸收塔坐标、定日镜尺寸、安装高度、以及定日镜位置等参数进行多约束目标优化。

2.3 对问题 3 的分析

问题 3 在问题 2 的基础上，增加了定日镜尺寸和高度可以不同的条件，这不仅仅是给优化模型增加了两个变量，也会给第一问的损耗模型带来影响。因此需要更多的考虑定日镜场的实际运用场景，对模型进一步扩展进行研究，给出更优的模型设计。

三、 模型假设

1. 尽管年份的不同可能会对太阳赤纬角的计算产生影响，但其影响较小。故为了简化模型，本题中不考虑闰年，假设一年均为 365 天。
2. 在集热器截断模型中考虑太阳光为锥形光线，其余模型中简化为平行光线。
3. 假设在每个监测的时间点均为晴天，忽略天气变化对定日镜场的影响。
4. 忽略定日镜之间的光学效应。
5. 假设集热器和定日镜不受其他无关外界因素影响。

四、 符号说明

符号	符号描述	单位
γ_s	太阳方位角	度°
α_s	太阳高度角	度°
α_o	定光镜方位角	度°
DNI	法相直接辐射辐照度	瓦特每平方米 W/m^2
d	定日镜镜面宽度	米 m
h_t	吸收塔高度	米 m
h_m	集热器高度	米 m
D_m	集热器直径	米 m
h_{mirror}	定日镜安装高度	米 m
L_{sb}	塔体投影长度	米 m
η	定日镜光学效率	/
η_{sb}	阴影遮挡效率	/
η_{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	/

五、 模型准备

5.1 太阳高度角

在地球上的任何给定位置，太阳高度角是指从地平线（即水平面）到太阳直射线的角度。它是一个垂直于地表的线与太阳光线之间的夹角，通常以度数为单位表示。太阳

高度角的测量通常是相对于地平线而言的，太阳位于地平线上方时高度角为正，位于地平线下方时高度角为负。

太阳高度角的值会随着时间、日期、地点以及地球上的季节而变化。太阳高度角的变化对于太阳能系统应用具有重要意义，因为它影响太阳光的强度和入射角度，从而影响能源收集、定位和气象现象等方面的问题。

5.2 太阳方位角

太阳方位角是指太阳直射线（从太阳到地球上的某一点的直线路径）与地理北方之间的水平夹角。它通常是以度数来表示的，以顺时针方向测量，其中 0 度表示太阳位于正北方，90 度表示太阳位于正东方，180 度表示太阳位于正南方，270 度表示太阳位于正西方。太阳方位角的测量与地点的纬度、太阳的时角以及地球的季节和时间有关。

太阳方位角的值用于确定太阳在天空中的位置，对于太阳能系统的朝向和定位具有关键作用。太阳方位角的计算可以根据特定地点和时间，结合太阳的时角和纬度等参数，使用几何和天文学公式来完成。

5.3 法向直接辐射辐照度 DNI

法相直接辐射辐照度（DNI）是指太阳光线以垂直于一个给定表面的方式到达该表面的能量通量。通俗地说，它是太阳光线直接射到一个表面上的能量强度，而不考虑光线在大气中的散射和透射。这种辐照度通常以单位面积（例如每平方米）的能量（通常以瓦特/平方米或瓦特/平方厘米）表示。

法相直接辐射辐照度是太阳能系统设计和性能评估的关键参数之一，因为它表示了在无遮挡情况下太阳光线的强度，可用于太阳能电池板、太阳能集热器和其他太阳能设备的能源收集计算。在太阳能系统的工程和科学研究中，了解法相直接辐射辐照度有助于确定太阳能系统的性能和效率，从而更好地规划和优化太阳能资源的利用。

5.4 定日镜长宽比假设

文献[3]曾以 Gemasolar 电站为案例，在定日镜场的采光面积保持不变的情况下，探究了定日镜的长宽比对于塔式太阳能聚光集热系统性能的影响。研究的实验结果显示，当定日镜的长宽比位于 1.3 至 1.5 之间时，系统可以实现高效的能量收集，同时保持相对较低的成本。此外，笔者还进行了市场调查，发现当前市场上已有的定日镜产品大多具有 1.2 至 1.5 之间的长宽比。因此，本文提出了一个假设，除了问题 1 已经给定 $6 \times 6(m)$ 的边长外，在确定定日镜的长宽比时，将其设置为一个介于之间 $1.2 - 1.5$ 的值是一个具有合理性的选择。

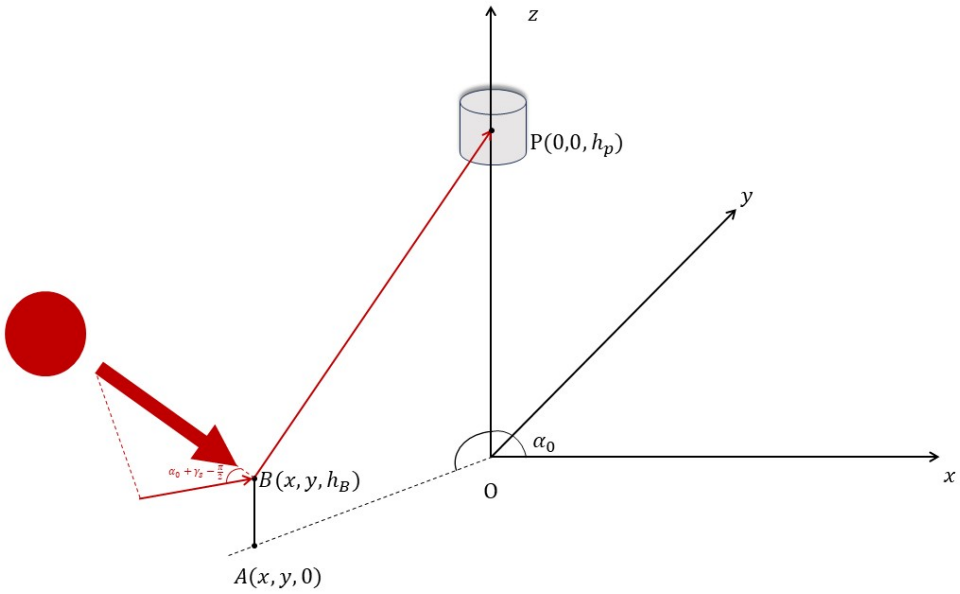


图 1　经 B、P 两点光路示意图

六、 问题 1：定日镜场系统光学效率计算

6.1 模型一: 定日镜光学损耗几何模型

6.1.1 余弦损失

对光的反射中，不可避免的会有能量损失。除了镜面材料的吸收和漫反射外，一个很重要的原因就是每一次反射过程都会产生透射现象，而只有由镜面采光口法线方向入射才可以避免这种损失。余弦效率计算公式为入射光线和法线的夹角。

如图1 所示，考虑射至定光镜中心 B 并反射至集热器中心 P 的这条光线。首先，由于太阳光会由于时间的不同而从不同方向入射，因此首先根据某个时刻的太阳方位角 γ_s 和定日镜方位角 α_0 来将太阳光投影到平面 ABPO 上作为入射光线。其光路图由2a所示。而根据确定的光路图可以确定法线和定日镜的倾斜角度，如2b所示，从而计算出余弦损失。

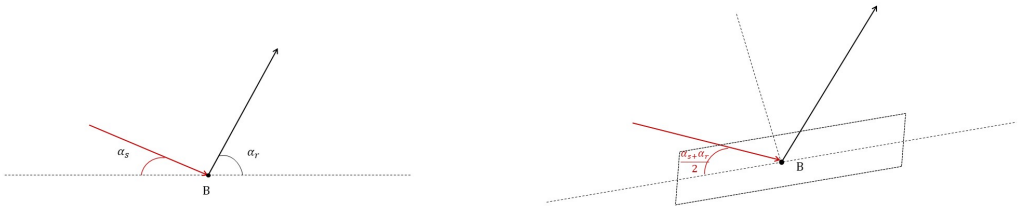
$$\tan(\alpha_o) = \frac{y}{x} \tag{1}$$

$$\tan(\alpha_r) = \frac{h_p - h_b}{\sqrt{x^2 + y^2}} \tag{2}$$

$$\tag{3}$$

而余弦效率可由太阳入射角的余弦值求得，在图2b中即为

$$\eta_{cos} = \sin(\frac{\alpha_r + \alpha_s}{2})$$



(a) **入射光路示意**　　(b) **定日镜倾斜角度示意**

图 2　余弦效率计算

6.1.2 集热器截断损失

来自太阳的入射光线是一束锥形光线，其半角展宽为 4.65 mrad(即全角展宽为 9.3 mrad)。

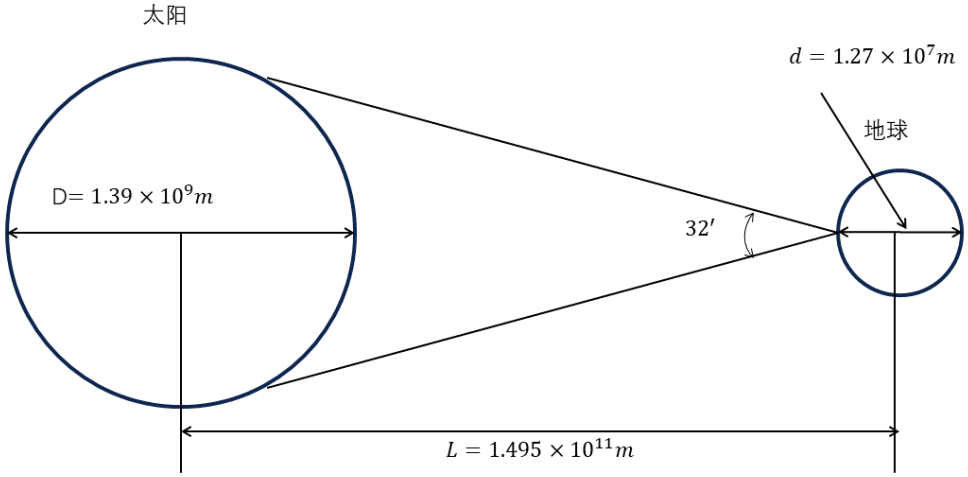
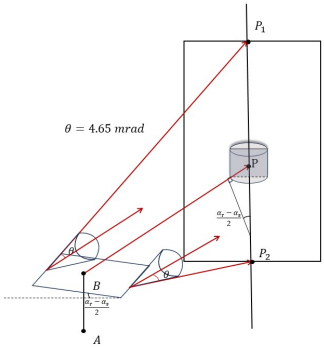


图 3　太阳张角

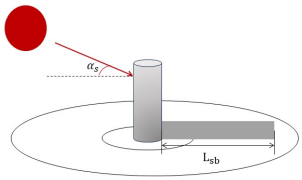
这是因为太阳轮廓相对地球表面上某一点的张角，称为太阳张角。根据太阳半径和日地距离，可以计算出太阳张角

$$\alpha_s = \arcsin(\frac{D/2}{L}) = \arcsin(\frac{1.39 \times 10^9/2}{1.495 \times 10^{11}}) = 32'$$

太阳张角表明从太阳直射到地球表面的光束并非一道平行光束，而是一道锥角为 32’(9.3 mrad) 的锥形光。根据光的反射定律，反射到集热器上的光线也呈锥形，从而有部



(a) **集热器截断模型**



(b) **吸收塔阴影示意图**

分锥形光线因为发散而无法被集热器所吸收，其造成的损失称为集热器截断损失。

如图 4a所示，首先我们通过几何关系确定出 P_1 与 P_2 的坐标。

$$h_{P_1} = h_P + \frac{\frac{d}{2}}{\cos(\frac{\alpha_r - \alpha_s}{2} + 4.65mrad)} \tag{4}$$

$$h_{P_2} = h_P - \frac{\frac{d}{2}}{\cos(\frac{\alpha_r - \alpha_s}{2} - 4.65mrad)} \tag{5}$$

再由 P_1P_2 线段长度估计定日镜在集热器平面的面，通过计算其集热器截面积的比值近似得到集热器截断效率结论，截断效率为

$$\eta_{trunc} = \begin{cases} \frac{h_m D_m}{|P_1P_2|^2} & h_{P_1} \geq 88m \ \text{并且} \ h_{P_2} \leq 80m \\ 1 & o.w. \end{cases}$$

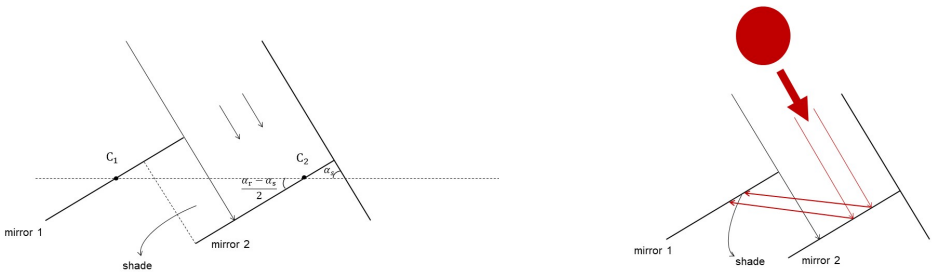
定日镜场阴影挡光损失 阴影挡光损失包括三部分: 1. 塔对镜场造成的阴影损失。2. 后排定日镜接收的太阳光被前方定日镜所阻挡被称为阴影损失。3. 后排定日镜在反射太阳光时被前方定日镜阻挡而未到达吸热器上被称为挡光损失。因此需要对“吸收塔阴影损失”和“定日镜间阴影遮挡损失”分别建模。

6.1.3 吸收塔阴影损失

太阳光从某个方向入射会使吸收塔产生一定面积的阴影，从而影响这部分区域定光镜的工作，根据有关文献[4]的计算结果，造成的效率之差多达 10%，因此排除掉这部分损失是有必要的。

题目中并未给出吸收塔的直径，但根据实物图，我们可以将其抽象为直径与集热器相同的圆柱体。

具体计算如4b所示，投影长度 L_{sb} 可由吸收塔、集热器长度之和和太阳高度角得出。再用塔体在圆环定光镜场区域的阴影占总定光镜场的面积之比来表示塔体阴影损耗。



(a) **入射阴影遮挡损失** (b) **反射遮光损失**

图 5 定日镜之间阴影遮挡示意图

$$L_{sb} = \frac{h_t + h_m}{\tan(\alpha_s)} \tag{6}$$

$$\eta_{sb\_tower} = 1 - \frac{D_m(L_{sb} - r)}{\pi(R^2 - r^2)} \tag{7}$$

6.1.4 定日镜间阴影遮挡损失

除了塔体阴影造成的损失，定日镜之间也会在接受太阳光或反射太阳光的过程中因为阴影、遮挡产生损耗，我们称之为定日镜间阴影遮挡损失。

两两考虑定日镜之间的阴影遮挡情况相当复杂，我们对模型做出一定的假设。

1. 对于某个定日镜来说，经由它的光线的阴影遮挡主要是由其前（后）方相邻的定日镜产生。（这里前（后）方相邻的判别为：在距离吸收塔的距离明显更大（小）的定日镜中选取离它最近的那面镜子）

2. 相邻定日镜的镜面理应接近平行。（误差程度可以接受）

建立在如上假设的基础上，以图5a为例，我们使用能接收到光线的镜面面积与完整镜面面积的比值来表示镜间阴影遮挡效率，并通过几何关系计算出相应的面积比值。图5b的情况可以使用类似的方式计算。

$$S_{block} = min(d', d)(\frac{1}{2}\frac{d' + d}{1.4^*} - \rho\cos(\frac{\alpha_r - \alpha_s}{2})) \tag{8}$$

$$S = \frac{d^2}{1.4} \tag{9}$$

$$\eta_{sb\_mirror} = \frac{S - S_{block}}{S} \tag{10}$$

其中 d' , d 分别是 mirror1 和 mirror2 的定日镜镜面宽度，ρ 是 C1C2 线段长度,S_{block} 是 mirror2 被遮光的面积。

综合 6.3.1 和 6.3.2 的结论，得到最后的阴影挡光损失：

$$\eta_{sb} = \eta_{sb\_tower} \times \eta_{sb\_mirror}$$

6.1.5 大气透射损失

定日镜反射聚焦到吸热器表面的太阳光线，由于当地的海拔高度、大气条件等因素的影响，在大气传播过程中存在一定的衰减，导致能量损失，称为大气衰减损失。计算公式为

$$\eta_{at} = 0.99321 - 0.0001176 d_{HR} + 1.97 \times 10^{-8} \times d_{HR}^2 \quad (d_{HR} \leq 1000)$$

其中 d_{HR} 表示镜面中心到集热器中心的距离（单位：m）

6.1.6 镜面反射率

镜面反射率用来描述定日镜反射太阳光线的能力，其与定日镜的材料、工艺和表面清洁度有关，当定日镜镜面处于清洁状态时，镜面反射率往往可达 92%-94%，本问题中设为常数 0.92。

6.2 问题 1 求解

根据定日镜的光学效率计算公式

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

当时间和定日镜参数确定后，便可以求得某时刻某块镜子的光学效率。

计算流程如下。

第一层循环：遍历月份（采取十二个特定日期）；

第二层循环：遍历定日镜；

第三层循环：遍历一天内五个时间点；

第一步：计算所需角度；

第二步：计算并记录各类光学效率；

第三步：计算并记录单位面积镜面平均输出热功率；

循环外：计算并记录所需效率均值和功率均值。

6.3 结果展示以及检验分析

求得数据见下附表2和表3。

根据输出的图表和数据分析图表可以发现，在影响定日镜光学效率的众多因素中，影响最大的是余弦损失，其余损失也均会一定程度地降低定日镜光学效率。

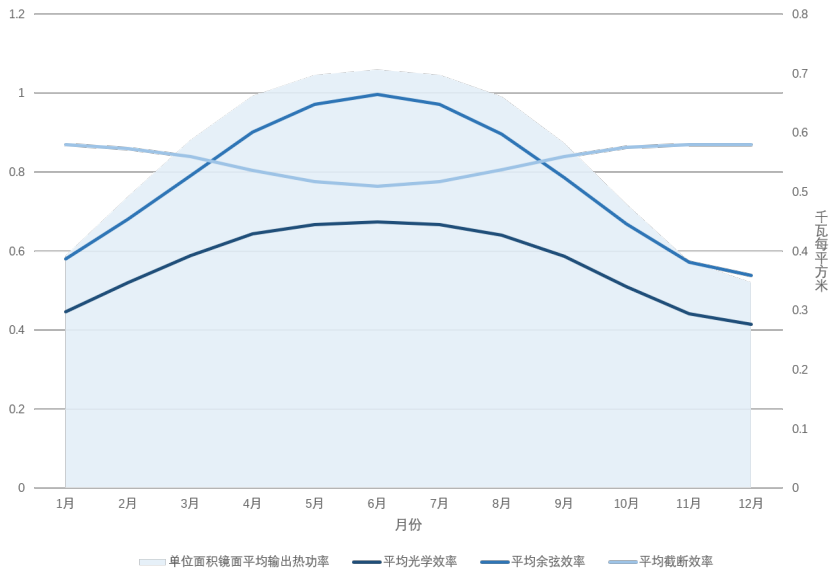


图 6　问题 1 结果分析图

七、 问题 2：定日镜场参数多目标优化设计

7.1 模型二：基于 SLSQP 的定日镜场局部调度优化模型

定日镜场调度优化问题，具有维度高，限制复杂，且限制动态变化的特征。该问题的解空间维度和反射镜数量线性相关；单面反射镜参数的范围难以设定；且反射镜参数的可行范围随其他反射镜的调整动态变化。

7.1.1 序列最小二乘法（Sequential Least Squares Programming，SLSQP）

在局部调度优化的过程中，我们使用了 SLSQP 方法。SLSQP 是一种常用的数学优化算法，通常用于非线性最小二乘问题的求解，它的基本思想是通过迭代优化的方式，逐步逼近最优解。

序列最小二乘法的优化过程，可以看做是一个序列的求解过程。在每一步迭代中，该算法会求解一个线性或非线性的最小二乘问题，并将当前的解作为下一步迭代的初始值。这样，序列最小二乘法就能够不断逼近最优解，直至达到收敛条件为止。

SLSQP 算法具有可以优化多变量函数，可以接收等式和不等式限制，收敛速度快，运算性能好等优势，适合用在本问题的局部调度优化过程中。

7.1.2 初始镜场分布

如何提升镜场的光学效率，特别是减小遮挡损失，一直以来都是镜场布局研究的热门课题。在现有的镜场中，径向交错的辐射网格排布已被广泛应用，但光学效率方面仍存在大量的遮挡损失。为了尽可能满足全年镜场中无遮挡损失的原则，业界[1]已经衍生

表 2　问题 1 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/m^2)
1 月 21 日	0.447	0.580	0.997	0.869	0.392
2 月 21 日	0.520	0.681	0.999	0.860	0.492
3 月 21 日	0.589	0.791	0.999	0.839	0.587
4 月 21 日	0.643	0.902	0.999	0.805	0.662
5 月 21 日	0.667	0.972	0.999	0.775	0.697
6 月 21 日	0.673	0.996	0.999	0.764	0.706
7 月 21 日	0.667	0.971	0.999	0.776	0.697
8 月 21 日	0.641	0.897	0.999	0.806	0.660
9 月 21 日	0.586	0.786	0.999	0.840	0.582
10 月 21 日	0.510	0.668	0.998	0.862	0.479
11 月 21 日	0.441	0.572	0.997	0.870	0.383
12 月 21 日	0.415	0.538	0.996	0.870	0.347

表 3　问题 1 年平均光学效率及输出功率表

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/m^2)
0.566	0.779	0.998	0.828	34.995	0.557

出了三种不同的布局方式，包括无遮挡 (EB) 布局、无阻塞 (No blocking-dense) 布局以及经验型 (DELSOL) 布局。

其中无遮挡布局 (EB 布局)[5] 是旨在确保太阳光线不会被任何障碍物遮挡，以最大程度地集中太阳能光线。这种布局通常需要准确的地形和太阳轨迹数据，以确保镜面可以在日照方向上自由地追踪太阳的运动，而不受任何阻碍。无阻塞布局 (No blocking-dense 布局) 是另一种定日镜场布局，旨在减少太阳光线被周围设备或结构阻挡的情况。

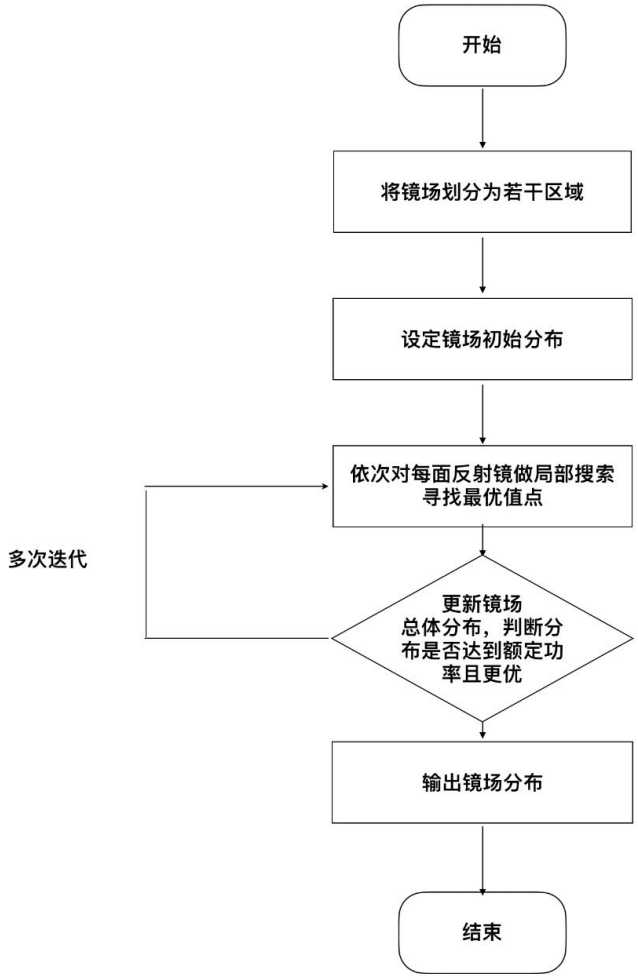
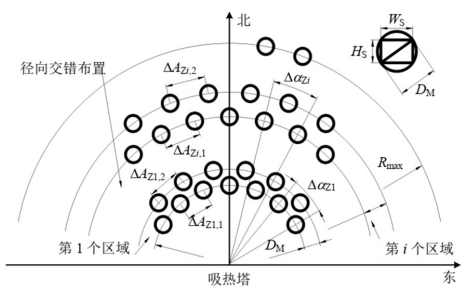


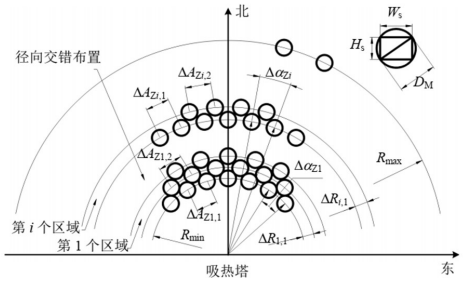
图 7　基于 SLSQP 定日镜场局部调度优化模型流程图

这种设计考虑到镜面的排列方式以及其周围的支持结构，以减少阻塞影响。目标是在实际可行性和太阳能收集效率之间找到一个平衡，以降低建设和维护成本。经验型布局 (DELSOL 布局) 是一种根据实际经验和先前项目的成功经验而设计的布局。这种布局可能不依赖于精确的地形或太阳轨迹数据，而是依赖于过去项目的经验来确定最佳镜面排列方式和系统参数。这种布局可能更适用于没有详细数据或资源受限的情况，但它通常会考虑到实际的太阳运动模式以最大程度地提高能源收集效率。

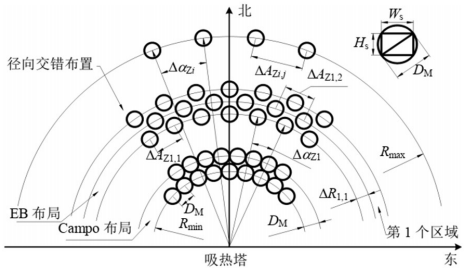
这些布局类型的选择取决于具体的项目需求、可用资源、技术水平和经济可行性。不同的定日镜场项目可能会采用不同的布局设计，以最大程度地提高太阳能的收集和利用效率。因此本文选择以这三种布局方式为背景展开研究，探索对于问题 2 最合适的布局方式。3 种镜场分布坐标等参数可以借助专业软件 SolarPILOT 来获取。



(a) DELSOL **布局示意图**



(b) EB **布局示意图**



(c) No blocking-dense **布局示意图**

图 8　三种常见布局示意图

7.1.3 模型二求解过程

解决该问题的首要目标是通过恰当的假设和预条件设定重新划分解空间。我们首先将定日镜场划分为若干区域，在此基础上确定初始镜场分布。对于每一面反射镜，我们在其所在区域内做局部搜索优化。将这个过程多次迭代，即可求得定日镜场排布。

流程图见图7。

7.2 光学效率模型应用于问题 2 的调整分析

因为吸收塔的坐标成为了需要优化的参数，而非问题 1 中的原点坐标，因此此前建立的光学损耗模型的部分细节需要作出相应调整。在五个损失模型中，镜面反射维持不变，余弦损失、大气透射损失、截断损失简单的更改公式中吸收塔的平面坐标即可得。吸收塔阴影遮挡损失经过讨论给出以下公式。

$$\eta_{sb\_tower} = \begin{cases} 1 - \frac{D_m(L_{sb}-r)}{\pi(R^2-r^2)} & \sqrt{X_t^2+Y_t^2} + L_{sb} \le R \\ 1 - \frac{D_m(R-\sqrt{X_t^2+Y_t^2}-r)}{\pi(R^2-r^2)} & \sqrt{X_t^2+Y_t^2} + L_{sb} \ge R \end{cases}$$

7.3 结果展示以及检验分析

关于问题 2 输出数据的镜场分布见图9a，可以看出，优化后仍比较符合预设的镜场分布形式。

表 4　问题 2 每月 21 日平均光学效率及输出功率

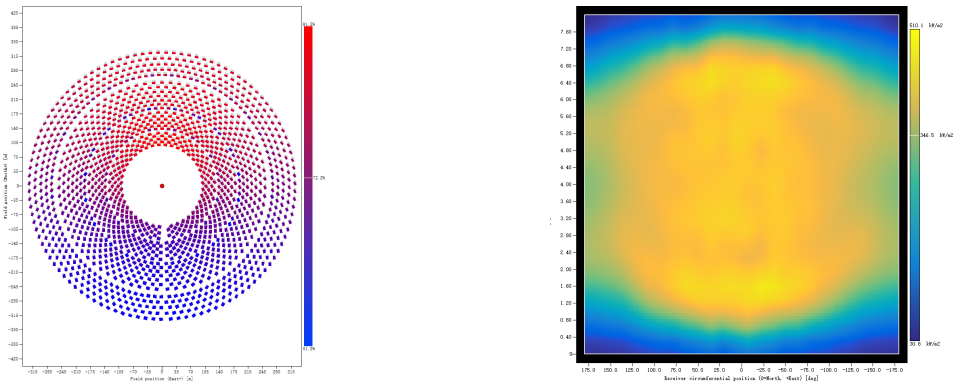
日期	平均光学效率	平均余弦效率	平均阴影遮挡效率	平均截断效率	单位面积镜面平均输出热功率 (kW/m^2)
1 月 21 日	0.452	0.586	0.997	0.869	0.753
2 月 21 日	0.523	0.683	0.998	0.862	0.941
3 月 21 日	0.591	0.789	0.999	0.844	1.120
4 月 21 日	0.645	0.894	0.999	0.812	1.262
5 月 21 日	0.669	0.961	0.999	0.784	1.328
6 月 21 日	0.674	0.984	0.999	0.773	1.345
7 月 21 日	0.668	0.960	0.999	0.784	1.327
8 月 21 日	0.643	0.889	0.999	0.814	1.257
9 月 21 日	0.588	0.783	0.999	0.845	1.111
10 月 21 日	0.514	0.670	0.998	0.864	0.918
11 月 21 日	0.446	0.578	0.997	0.870	0.736
12 月 21 日	0.421	0.546	0.996	0.870	0.669

表 5　问题 2 年平均光学效率及输出功率表

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/m^2)
0.570	0.777	0.999	0.833	60.707	1.064

表 6　问题 2 设计参数表

吸收塔位置坐标	定日镜尺寸 (width ×height)	定日镜安装高度 (m)	定日镜总面数	定日镜总面积 (m^2)
(0,0)	7.9×7.1	4	1585	88902.65



(a) **问题 2 镜场分布图**　　(b) **问题 2 集热器能流图**

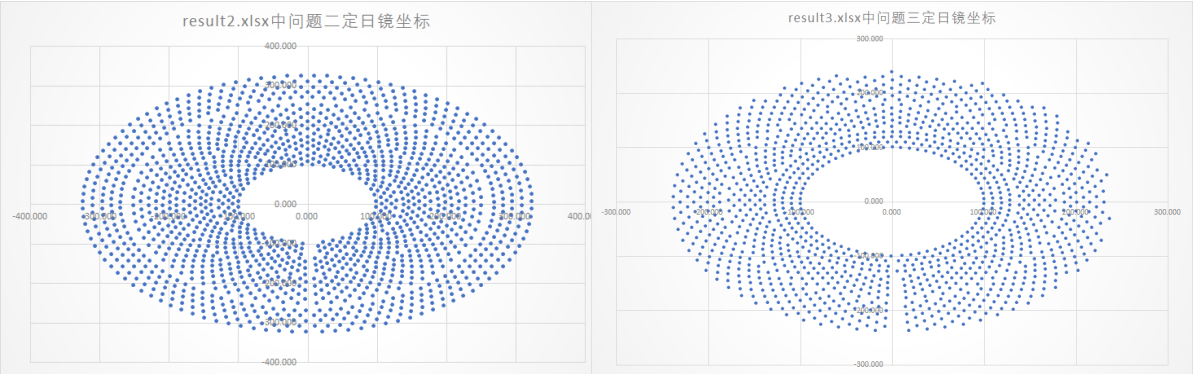
图 9　问题 2 设计示意图

八、 问题 3： 优化模型的参数增维拓展

8.1 问题 3 分析

问题 3 在问题 2 的基础上放宽了定日镜尺寸和安装高度的要求，令每个定日镜的情况可以互异。这相当于把定日镜尺寸、高度从一个外层优化的单变量，转变成每个镜子在搜索自己局部最优解时都要去优化的高维参数，即每块镜子在自身搜索解空间时需要优化的参数变成了 [坐标，尺寸，高度]，这提高了模型求解的维数，加大了求解的难度和时间。尽管增加了求解难度，降低了效率，但在本质上和问题 2 是同类的优化问题，可以在相同的算法模型的基础上改进求解，但同时也要思考如何改进使得效率、收敛性等的性能有所提高。

- 本次设计的某些模型比如阴影模型，效率比较高，且变化幅度小。因此可以考虑做一些估计，简化相应计算流程，从而优化计算。
- 恰当地选择初始值和步长可以优化解题流程，本问中可以在问题 2 得到的优化参数的基础上，进行局部的解搜索，从而尽可能提高解题效率和收敛速度。
- 应根据模型维数、问题限制等条件恰当选择基于个体的搜索方法或者基于群体的搜索方法。
- 应根据计算难度，梯度是否可计算等条件恰当选择基于梯度的搜索方法或者启发式



(a) result2.xlsx **中问题 2 定日镜坐标**　　(b) result3.xlsx **中问题 3 定日镜坐标**

图 10　定日镜坐标

局部搜索法。

- 对于不同的问题要求和性能要求，恰当选择局部最优的搜索方案或者全局最优的搜索方案。

8.2 结果展示以及检验分析

注意到，问题 2 和问题 3 还要求生成 result2.xlsx 和 result3.xlsx，因为表格维数较大，无法呈现在正文以及附录中，这两个文件将附在支撑文件中。此外，将 result2.xlsx 和 result3.xlsx 中的部分重要数据（这里指生成的定日镜坐标）用散点图的形式呈现在10

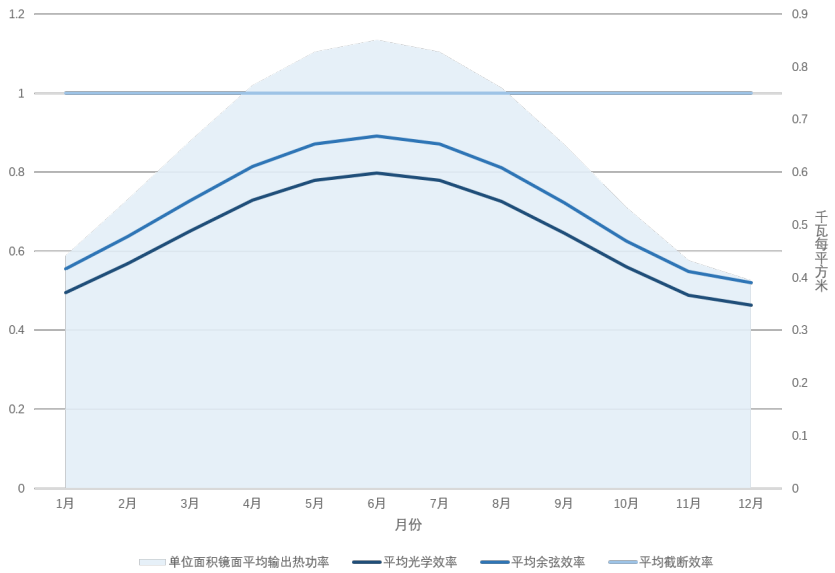


图 11　问题 2 表 1 数据分析

表 7　问题 3 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/m^2)
1 月 21 日	0.495	0.555	0.998	1.000	0.441
2 月 21 日	0.569	0.637	0.999	1.000	0.548
3 月 21 日	0.650	0.727	1.000	1.000	0.659
4 月 21 日	0.729	0.815	1.000	1.000	0.764
5 月 21 日	0.779	0.872	1.000	1.000	0.829
6 月 21 日	0.797	0.891	1.000	1.000	0.851
7 月 21 日	0.779	0.871	1.000	1.000	0.829
8 月 21 日	0.726	0.812	1.000	1.000	0.760
9 月 21 日	0.645	0.722	1.000	1.000	0.653
10 月 21 日	0.560	0.626	0.999	1.000	0.534
11 月 21 日	0.489	0.548	0.998	1.000	0.432
12 月 21 日	0.464	0.520	0.997	1.000	0.394

表 8　问题 3 年平均光学效率及输出功率表

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/m^2)
0.640	0.716	0.999	1.000	61.117	0.641

九、 模型总结与评价

本文综合考虑定日镜工作过程中的各种损耗，结合几何关系确定了定日镜光学效率的数计算公式，建立了基于 SLSQP 的定日镜场局部调度优化模型，并应用于多目标优化的塔式电站定日镜场的优化设计中。本文所建立模型和求解过程具有以下特点：

问题 1

表 9　问题 3 设计参数表

吸收塔位置坐标	定日镜尺寸 (width ×height)	定日镜安装高度 (m)	定日镜总面数	定日镜总面积 (m^2)
(0,0)	/	/	1299	95322.97

（1）该问的核心在于对光学效率损耗建立的模型精度是否足够高。总体观察而言，计算出的各个效率都比较符合基本逻辑。当然为了简便计算，在建模过程中做出了不少假设，会带来一定的误差。比如在定日镜间阴影遮蔽模型建立时，便只考虑了相邻镜子的影响，这建立在以同心圆为主要排布趋势且排列较密的调度方式中。而结果来看假设的合理性得到一定验证，比如前文提到的 no-blocking dense 分布和 EB 分布的高阴影效率，便可证明模型有一定的合理性。

（2）当然，本问建模为了简化计算而做出了几个假设。倘若使用坐标变换法和光线追迹法来解析光线经过的位置，应该能够进一步提升解答的准确性和精度。

问题 2

（1）该问在第一问建立的损耗模型的基础上，建立了随吸收塔坐标变动而改变的更精细的模型，减少了一定误差。

（2）针对该问的多目标高维数的优化问题，本文借助经典镜场分布模型先大致确定排布，再采用局部最优搜索算法进行局部优化，减轻了原问题计算压力。同时，搜索算法选择 SLSQP，适用于非线性连续问题，速度快，精度高，收敛性能较好。

问题 3

(1) 该问是第二问的深化，增加了优化的变量，放宽了约束，相应也增加了维数。本问中合理选择初始值使得收敛性能变好，合理简化损耗模型减轻计算压力，合理选择局部搜索算法使其恰好符合本问题的应用场景。

参考文献

[1] 孙浩, 高博, 刘建兴. 塔式太阳能电站定日镜场布局研究[J]. 发电技术, 2021, 42 (690-698).

[2] 张平, 奚正稳, 华文瀚, 王娟娟, 孙登科. 太阳能塔式光热镜场光学效率计算方法[J]. 技术与市场, 2021, 28(6): 4.

[3] 吕彩霞. 定日镜参数对塔式太阳能聚光集热系统性能的影响[J]. 节能, 2023, 42(5): 34-37.

[4] 郭苏, 刘德有. 塔式太阳能热发电厂的定日镜有效利用率计算[C]//上海材料研究所《能源技术》编辑部. 长三角清洁能源论坛论文专辑. 上海材料研究所《能源技术》编辑部, 2005: 100-103.

[5] 何杰, 杜行, 奚正稳, 徐文奇, 华文瀚. 塔式太阳能定日镜安置位的研究[J]. 技术与市场, 2020, 27(1-5).

附录

附录 1		
支撑文件清单		
文件夹名	文件名	含义
数据	result2.xlsx	问题 2 结果
	result3.xlsx	问题 3 结果
代码	to1.py	塔阴影计算
	tr.py	截断效率计算
	bs1.py	阴影挡光计算
	co.py	余弦效率计算
	ef1.py	角度和效率计算
	nc.py	角度计算和数据处理
	DNI.py	法向直接辐射辐照度计算
	ef2.py	角度和效率计算
	ps.py	局部搜索调度优化

附录 2

to1.py 塔阴影计算

```
"""Find tower shading efficiency"""

import numpy as np

def calc(alpha_s): # Measured in radians
   """Calculate tower shading efficiency"""

   a = 1 - (7 * ((88 / np.tan(alpha_s)) - 100)) / (np.pi *
       (250) * (450))
   eta = np.where(a > 1, 1, a)
   return eta
```

附录 3

tr.py 截断效率计算

```
"""Find truncation efficiency"""

import numpy as np

def calc(i):
    """Calculate truncation efficiency"""

    from nc import mirror
    import ef1

    bound = 4.65 / 1000 # 4.65mrad
    h1 = 84 + mirror[i, 3] / (2 * np.cos((ef1.alpha_r -
        ef1.alpha_s) / 2 + bound))
    h2 = 84 - mirror[i, 3] / (2 * np.cos((ef1.alpha_r -
        ef1.alpha_s) / 2 - bound))
    a = 56 / np.square(h1 - h2)
    eta_tr = np.where(h1 >= 88 and h2 <= 80, a, 1)
    return eta_tr
```

附录 4

bs1.py 阴影挡光效率计算

```
"""Find blocking and shading efficiency"""

import numpy as np

def calc(i, alpha_r, alpha_s): # Greek letters measured in
    radians
   """Calculate bs efficiency (excluding tower shading)"""

   from nc import mirror, my_mirror

   j = my_mirror[i]
   if (j == -1):
      return 1
   hi = mirror[i, 3]
   hj = mirror[j, 3]
   di = mirror[i, 4]
   dj = mirror[j, 4]
   a = np.minimum(di, dj) * ((hi + hj) * 0.5 -
       np.linalg.norm(mirror[i, 0:3] - mirror[j, 0:3]) *
       np.cos((alpha_r - alpha_s) * 0.5))
   return np.where(a <= 0, 1, 1 - a / (hi * hj))
```

附录 5

bs1.py 余弦效率计算

```
"""Find cosine efficiency"""

import numpy as np

def calc(alpha_o, alpha_r, alpha_s, gamma_s): # Measured in
    radians
   """Calculate cosine efficiency"""
   return np.sin((alpha_s + alpha_r) / 2)
```

附录 6

ef1.py 角度和效率计算

```
"""Find the efficiency of a mirror at a certain time"""

alpha_o, alpha_r, omega, delta, alpha_s, gamma_s, eta_sb,
    eta_cos, eta_at, eta_trunc, eta_ref, eta = 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0


import numpy as np

def calc(i, month, hour):
   """Calculate the said efficiency

   This function has side effects."""

   import nc, bs1, to1, co, at, tr

   global alpha_o, alpha_r, omega, delta, alpha_s, gamma_s,
       eta_sb, eta_cos, eta_at, eta_trunc, eta_ref, eta

   global eta_bs1, eta_to1


   na = np.array
   alpha_o = np.arctan2(nc.mirror[i, 1], nc.mirror[i, 0])
   x = 3.5 * np.cos(alpha_o)
   y = 3.5 * np.sin(alpha_o)
   alpha_r = nc.Angle(nc.mirror[i, 0:3], na([x, y,
       nc.mirror[i, 2]]), na([x, y, 84]))
   phi = nc.phi
   omega = nc.Omega(nc.ST[hour])
   delta = nc.Delta(nc.D[month])
   alpha_s = nc.Alpha_s(delta, phi, omega)
   gamma_s = nc.Gamma_s(delta, phi, alpha_s)
```

```
    eta_bs1 = bs1.calc(i, alpha_r, alpha_s)
    eta_to1 = to1.calc(alpha_s)
    eta_sb = eta_bs1 * eta_to1
    eta_cos = co.calc(alpha_o, alpha_r, alpha_s, gamma_s)
    eta_at = at.calc(nc.mirror[i][5])
    eta_trunc = tr.calc(i)
    eta_ref = 0.92
    eta = eta_sb * eta_cos * eta_at * eta_trunc * eta_ref
    return eta
```

附录 7

nc.py 角度计算和数据处理

```
"""Number crunching"""

import numpy as np

G_0 = 1.366 # kW/m2
H = 3 # km
phi = np.radians(39.4)
D = np.array([-59, -28, 0, 31, 61, 92, 122, 153, 184, 214,
    245, 275]) # Year: 2023; index starts from 0
ST = np.array([9, 10.5, 12, 13.5, 15])

num = 0
area = 0
mirror = np.ndarray(shape=(0, 6))
my_mirror = np.ndarray(shape=(0))

def Delta(D):
   """Calculate solar declination

   This function takes a value from the array D as input."""

   a = np.sin(2 * np.pi * D / 365) * np.sin(np.pi * 23.45 /
       180)
   a = np.where(a > 1, 1, a)
   a = np.where(a < -1, -1, a)
   return np.arcsin(a)

def Omega(hour):
   """Calculate solar hour angle

   This function takes a value from the array ST as input."""

   return np.pi / 12 * (hour - 12)
```

```
def Alpha_s(delta, phi, omega):
    """Calculate solar altitude angle"""

    a = (np.cos(delta) * np.cos(phi) * np.cos(omega)) +
        (np.sin(delta) * np.sin(phi))
    a = np.where(a > 1, 1, a)
    a = np.where(a < -1, -1, a)
    return np.arcsin(a)

def Gamma_s(delta, phi, alpha_s):
    """Calculate solar azimuth angle"""
    a = (np.sin(delta) - np.sin(alpha_s) * np.sin(phi)) /
        (np.cos(alpha_s) * np.cos(phi))
    a = np.where(a > 1, 1, a)
    a = np.where(a < -1, -1, a)
    ans = np.arccos(a)
    return ans

def Angle(a, b, c): # A is vertex
    """Calculate angle BAC, return value in radian

    a, b, c should be numpy arrays containing coordinates (x,
        y, z).
    """

    ab = b - a
    ac = c - a
    cosine_angle = np.dot(ab, ac) / (np.linalg.norm(ab) *
        np.linalg.norm(ac))
    cosine_angle = np.where(cosine_angle > 1, 1, cosine_angle)
    cosine_angle = np.where(cosine_angle < -1, -1, cosine_angle)
    return np.arccos(cosine_angle)
```

```
def find_my_mirror(mirror):
   """For each mirror find the closest mirror that has a
       larger d_hr"""

   global my_mirror
   my_mirror = np.ndarray(shape=(num), dtype=int)
   meerror = mirror[:, 5].argsort()
   for i in range(num):
      myne = -1
      my_dist = 100000
      for j in range(i + 1, num):
         new_dist = np.linalg.norm(mirror[meerror[i], 0:3] -
             mirror[meerror[j], 0:3])
         if (new_dist < my_dist):
            myne = j
            my_dist = new_dist
      my_mirror[meerror[i]] = meerror[myne]
```

附录 8

DNI.py 法向直接辐射辐照度计算

```
"""Find DNI"""

import numpy as np

def Dni(G_0, H, alpha_s): # alpha_s measured in radians
   """Calculate DNI"""

   a = 0.4237 - 0.00821 * np.square(6 - H)
   b = 0.5055 + 0.00595 * np.square(6.5 - H)
   c = 0.2711 + 0.01858 * np.square(2.5 - H)
   dni = G_0 * (a + b * np.exp(-c / np.sin(alpha_s)))
   return dni
```

附录 9

ef2.py 角度和效率计算

```
"""Find the efficiency of a mirror at a certain time"""

alpha_o, alpha_r, omega, delta, alpha_s, gamma_s, eta_sb,
    eta_cos, eta_at, eta_trunc, eta_ref, eta = 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0


import numpy as np

def calc(i, month, hour):
   """Calculate the said efficiency

   This function has side effects."""

   import nc, co, at, tr
   import ps, p2

   global alpha_o, alpha_r, omega, delta, alpha_s, gamma_s,
       eta_sb, eta_cos, eta_at, eta_trunc, eta_ref, eta

   na = np.array
   x, y = ps.pol2cart(p2.particle[i, 0], p2.particle[i, 1])
   alpha_o = np.arctan2(y, x)
   alpha_r = nc.Angle(na([x, y, p2.particle[i, 2]]), na([0, 0,
       p2.particle[i, 2]]), na([0, 0, 84]))
   phi = nc.phi
   omega = nc.Omega(nc.ST[hour])
   delta = nc.Delta(nc.D[month])
   alpha_s = nc.Alpha_s(delta, phi, omega)
   gamma_s = nc.Gamma_s(delta, phi, alpha_s)
   eta_sb = 0.999
   eta_cos = co.calc(alpha_o, alpha_r, alpha_s, gamma_s)
   eta_at = at.calc(at.D_hr(i))
   eta_trunc = tr.calc(i)
   eta_ref = 0.92
   eta = eta_sb * eta_cos * eta_at * eta_trunc * eta_ref
   return eta
```

附录 10

ps.py 局部搜索调度优化

```
"""Incorporate global and local search"""

import numpy as np
import p2

shuffle_count = 0
shuffle_result = np.array([])
adjust_ret = None

def place(rou, theta, z, length, width):
   """Five parameters are set: coordinates(rou, theta, z),
       length, and width."""

   newone = np.array([[rou, theta, z, length, width]])
   if (len(p2.particle) <= p2.num):
      p2.particle = np.append(p2.particle, newone, axis=0)
   else:
      p2.particle[p2.num] = newone[0]
   p2.num += 1

def remove():
   """Remove (pop) the last particle"""

   p2.num -= 1

def gridpos(i):
   """Find the grid containing the particle"""

   rou, theta = p2.particle[i, 0:2]
   nt = int(theta // (2 * np.pi / p2.maxt))
   nr = int((rou - 100) // (600 / p2.maxr))
   return (nr, nt)
```

```
def nbhd(i):
   """Find the four neighbors of a particle

   Return in shape=(4, 2), invalid result set to [-1, -1]"""

   nr, nt = gridpos(i)
   a = np.array([[nr - 1, nt - 1], [nr - 1, nt + 1], [nr + 1,
       nt - 1], [nr + 1, nt + 1]], dtype=int)

   for i in range(4):
      if (a[i, 0] < 0) or (a[i, 0] >= p2.maxr):
         a[i] = np.array([-1, -1], dtype=int)
      if (a[i, 1] == -1):
         a[i, 1] = p2.maxt - 1
      if (a[i, 1] == p2.maxt):
         a[i, 1] = 0
   return a

def cart2pol(x, y):
   """Cart to pol, return (rho, theta)

   This function takes two scalars as input."""

   rho = np.sqrt(x**2 + y**2)
   theta = np.arctan2(y, x)
   return (rho, theta)

def pol2cart(rho, theta):
   """Pol to cart, return (x, y)

   This function takes two scalars as input."""

   x = rho * np.cos(theta)
   y = rho * np.sin(theta)
   return (x, y)
```

```
def dist2p(i, j):
    """Calculate the distance between two particles in
        3d-space"""

    if (i == -1) or (j == -1):
        return 100
    x1, y1 = pol2cart(p2.particle[i, 0], p2.particle[i, 1])
    x2, y2 = pol2cart(p2.particle[j, 0], p2.particle[j, 1])
    return np.sqrt((x1 - x2)**2 + (y1 - y2)**2)


def dkp(i):
    """Return True if proper Distance is Kept between Particle
        i and its neighbors"""

    neii = nbhd(i)
    for i in range(4):
        if (neii[i, 0] == -1):
            continue
        rou, theta = neii[i]
        if (p2.occupied[rou, theta] == -1):
            continue
        j = p2.occupied[rou, theta]
        dij = dist2p(i, j)
        width = np.maximum(p2.particle[i, 4], p2.particle[j, 4])
        if (dij < width + 5):
            return False
    return True

def infield(rou, theta, opgt, oprt):
    """Return True if (rou, theta) is in the heliostat field"""

    x1, y1 = pol2cart(rou, theta)
    x2, y2 = pol2cart(oprt, opgt - np.pi)
    return np.sqrt((x1 - x2)**2 + (y1 - y2)**2) < 350
```

```
def run(opgt, oprt, opt, opr, opl, opw, opz, maxnum,
    testround):
   """Run with gamma_t, rou_t ..."""

   p2.maxnum = maxnum
   p2.rou_t = oprt
   preset(opgt, oprt, opt, opr, opl, opw, opz)
   for i in range(testround):
      shuffle()
   print(shuffle_result)

def preset(opgt, oprt, opt, opr, opl, opw, opz):
   """Preset initial conditions"""

   lt = p2.maxt
   lr = p2.maxr
   for i in range(lr):
      for j in range(lt):
         if (p2.otrange[j] <= i):
            continue
         if (not infield(i, j, opgt, oprt)):
            p2.otrange[j] = i
            continue
   for i in range(lr - 1):
      for j in range(lt - 1):
         if (p2.num >= p2.maxnum):
            break
         if (np.minimum(p2.otrange[j], p2.otrange[j + 1]) <=
             lr + 1):
            continue
         rou = (250 + oprt) / p2.maxr * (i * 2 + 1) / 2
         theta = 2 * np.pi / p2.maxt * (j * 2 + 1) / 2
         place(rou, theta, opz, opl, opw)
         if (not dkp(p2.num - 1)):
            remove()
            continue
         p2.occupied[i, j] = int(p2.num - 1)
```

```
def adjust(i):
   """Adjust parameters of a mirror locally

   Maximize DNI * eta.
   This function has side effects."""

   from scipy.optimize import minimize

   power = 0

   def target(x):
      """The target function to be minimized"""

      import ef2
      import DNI, nc

      nonlocal i, power

      power = 0
      for month in range(12):
         for hour in range(5):
            place(x[0], x[1], p2.particle[i, 2],
                p2.particle[i, 3], p2.particle[i, 4])
            eta = ef2.calc(p2.num - 1, month, hour)
            dni = DNI.Dni(nc.G_0, nc.H, ef2.alpha_s)
            remove()
            power += dni * eta / 60
      return -power

   def consdkp(x):
      """dkp as a contraint function"""

      nonlocal i

      place(x[0], x[1], p2.particle[i, 2], p2.particle[i, 3],
          p2.particle[i, 4])
      a = np.where(dkp(p2.num - 1), 1, -1)
      remove()
      return a
```

```
   global adjust_ret

   nr, nt = gridpos(i)
   sr = (250 + p2.rou_t) / p2.maxr
   st = 2 * np.pi / p2.maxt
   bnds = ((nr * sr, (nr + 1) * sr), (nt * st, (nt + 1) * st))
   cons = ({'type':'ineq', 'fun':consdkp})
   res = minimize(fun=target, x0=p2.particle[i, 0:2],
       method='SLSQP', bounds=bnds, constraints=cons)
   adjust_ret = res
   return power

def shuffle():
   """Shuffle mirrors globally, one at a time

   This function has side effects."""

   global adjust_ret, shuffle_count, shuffle_result

   power = 0
   for i in range(p2.num):
      a = adjust(i)
      b = adjust_ret.x
      power += a / p2.num
      p2.particle[i, 0:2] = b
   shuffle_count += 1
   shuffle_result = np.append(shuffle_result, power)
   print(power * p2.num * 35)
   return power
```