定日镜场优化设计模型

摘要

定日镜场的优化设计对实现高效的太阳能光热转换至关重要。本文基于**蒙特卡洛光线追踪法**，建立了阴影遮挡效率和截断效率的计算模型模型，求解出了定日镜场的年均光学效率和年均热输出功率。此外，通过建立**双层规划**模型，在满足额定功率的前提下，对定日镜场的各个参数进行优化，以最大化单位镜面面积的年均热输出功率。

针对问题一：本文建立了一个**基于蒙特卡洛光线追迹法的统计模型**。首先，为描述入射光的非平行性，采用了二维正态分布来模拟实际的光锥光束。其次，根据反射定律和光线追迹法确定了单根光线的运动轨迹，并判断光线和其他定日镜或吸热器的相交情况。最后，通过蒙特卡洛方法，对大量光线的轨迹进行统计，计算出阴影遮挡效率和截断效率，最终计算出定日镜场的年均光学效率 **61.58%**、年均热输出功率 **37.629MW**，以及单位镜面面积的年均热输出功率 **0.599kW/m^2**。

针对问题二：本文构建了**同心圆密排布局方式和双层规划模型**。通过同心圆密排布局方式建立了定日镜尺寸和定日镜位置、数量的关系，极大地减少了计算量。通过双层规划，在上层规划中确定了吸收塔的具体位置，并将其传递给下层。下层在上层规划的基础上，对定日镜的参数进行优化，并将优化结果反馈给上层，以进一步调整吸收塔的位置。通过控制变量法进行遍历，反复迭代后得到最大的单位镜面面积的年均热输出功率为 **0.591kW/m^2**。

针对问题三：本文在第二问的基础上，参考实际定日镜场布局，建立了**定日镜尺寸和安装高度与同心圆半径的函数关系**，对问题二的**双层规划模型**进行了修改，通过控制变量法进行遍历，得到了最大化单位镜面面积年平均输出热功率为 **0.594kW/m^2**。

关键词：蒙特卡洛光线追迹法　双层规划　定日镜场布局优化　光学功率

1 问题重述

1.1 背景分析

在塔式太阳能光热发电站中，定日镜场扮演着至关重要的角色，它由众多定日镜组成。这些定日镜随着太阳的运动而自动调整位置，以精确地将太阳光线汇聚到吸收塔上，从而产生大量的热能，最终经过转化成电能。通过建立数学模型，我们可以调整定日镜的位置和尺寸，以最大程度地提高光热发电的效率，这个过程对于实现高效的可再生能源产生至关重要。

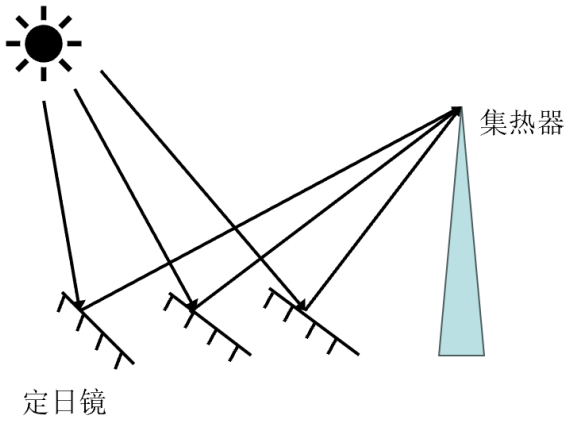

图 1 定日镜场示意图

1.2 问题提出

问题一要求在一个圆形定日镜场内，在已知吸收塔的位置、定日镜尺寸和安装高度、所有定日镜中心的坐标的情况下，建立数学模型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，以评估该光热发电站的性能。

问题二旨在在所有定日镜具有相同尺寸和安装高度的情况下，优化设计圆形定日镜场，以满足年均 60 兆瓦热功率输出要求。优化目标是确定吸收塔最佳的位置坐标和定日镜的尺寸、安装高度、位置坐标，以最大化单位镜面面积的年均输出热功率，实现高效光热发电。

问题三要求重新设计定日镜场的各个参数，包括定日镜尺寸和安装高度可以

不同，但额定功率仍为问题 2 中的 60 兆瓦（MW）。设计目标是找到吸收塔的位置坐标、各定日镜尺寸、安装高度、定日镜数目以及定日镜的位置坐标，以使得定日镜场能够在达到额定功率的条件下，尽可能地提高单位镜面面积年平均输出热功率。

2 问题分析

2.1 问题一的分析

问题一要求计算给定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。需要建立不同光学效率的模型，平均阴影遮挡效率和平均截断效率是问题一的关键。通过蒙特卡洛光线追迹法，对单根光线进行追踪。由于太阳光并非平行光线，而是具有一定锥形角的光锥，通过建立光锥能流密度模型，用二维正态分布去刻画不同的光线的入射角度。结合光线的分布符合二维正态分布的特点，通过统计大量光线的路径，判断光线和其他定日镜的接触情况，计算照射到集热器的光线总数，以此得到阴影遮挡效率和平均截断效率。在完成所有效率计算后，可以得出年均效率和年均功率的结果。

2.2 问题二的分析

问题二要求定日镜场在达到额定功率的条件下，优化定日镜位置、尺寸、数量、安装高度以及吸收塔位置等参数，尽可能地增大单位镜面面积年平均输出热功率。本文建立同心圆密排布局方式和双层规划模型来优化最大单位面积年平均输出热功率，将复杂的多参数优化问题进行拆分。上层规划确定了吸收塔的具体位置，并传递给下层；下层在上层的规划基础上，对定日镜的参数进行优化，并将优化结果反馈给上层，上层规划再根据反馈调整吸收塔的位置。通过控制变量法遍历参数，多次循环迭代可以得到局部最优结果。

2.3 问题三的分析

问题三要求在问题二的基础上，令定日镜尺寸和安装高度也可以各不相同，

在达到额定功率的条件下，尽可能地增大单位镜面面积年平均输出热功率。根据真实的定日镜场布局方式，建立同心圆半径和定日镜安装高度、尺寸的函数关系，极大地减少了计算量。在问题二中建立的同心圆密排布局方式和双层规划模型的基础上进行修改，用控制变量法遍历得到了局部最优的单位镜面面积年平均输出热功率。

3 模型假设

为了简化模型，本文做出以下合理的假设或条件约束：

1. 假设光锥在圆盘面上的能流密度满足二维正态分布。
2. 假设光锥半角宽度为 4.65mrad。
3. 假设在同一时间下，每个定日镜在无损失的情况下接受到的太阳光线是一样多的。
4. 假设光锥中的太阳中心点发出的太阳主光线经定日镜中心反射后指向集热器中心。
5. 假设定日镜围绕着吸收塔呈同心圆状分布，且同一个同心圆上定日镜的所有间距是均匀的。
6. 假设在同一个同心圆上的所有定日镜的参数是相同的。

4 符号约定

参数名称	说明	单位
N	定日镜的总数	面
$\eta_i^{(jk)}$	第 i 个定日镜在第 j 个月的 21 日的第 k 个时间段的光学效率	
n_i	第 i 个定日镜的法向向量	
I_{sun}	太阳主光线入射光线单位向量	
R_{sun}	太阳主光线反射光线单位向量	
I_{offset}	非平行太阳光入射光线单位向量	
N_g	每个镜子接收到的太阳光线总条数	个
N_I	被遮挡的入射光线总条数	个
N_R	被遮挡的出射光线总条数	个
α_s	太阳高度角	度
γ_s	太阳方位角	度
δ	太阳赤纬角	度
$E_{\text{field}}^{(jk)}$	第 j 个月的 21 日的第 k 个时间段定日镜场的输出热功率	
(x_i, y_i, z_i)	第 i 个定日镜的坐标和安装高度	米
(X_t, Y_t, H_t)	吸收塔的坐标和吸收塔高度	米

参数名称	说明	单位
w_i	定日镜的镜面宽度	米
l_i	定日镜的镜面高度	米
$\Delta R_{\min}$	最小安全距离	米
$N_n = \dfrac{2\pi r_n}{\Delta R_{\min}}$	第 n 个同心圆上定日镜的数量	个
n	同心圆的总数量	个

注：在具体的模型中，将对符号进行分别说明

5 模型的建立与求解

5.1 问题一模型的建立与求解

5.1.1 定日镜法向向量模型的建立

定义ST为当地时间，D为以春分作为第0天起算的天数，太阳时角以及太阳赤纬角的表达式[1]为

$$\omega=\frac{\pi}{12}(ST-12) \tag{5-1}$$

$$\sin\delta=\sin\frac{2\pi D}{365}\sin\left(\frac{2\pi}{360}23.45\right) \tag{5-2}$$

以地面为基准，以圆形区域中心为原点，正东方向为x轴正向，正北方向为y轴正向，垂直于地面向上方向为z轴正向建立镜场坐标系。太阳的高度角α_s和方位角γ_s的表达式为

$$\sin\alpha_s=\cos\delta\cos\varphi\cos\omega+\sin\delta\sin\varphi \tag{5-3}$$

$$\cos\gamma_s=\frac{\sin\delta-\sin\alpha_s\sin\varphi}{\cos\alpha_s\cos\varphi} \tag{5-4}$$

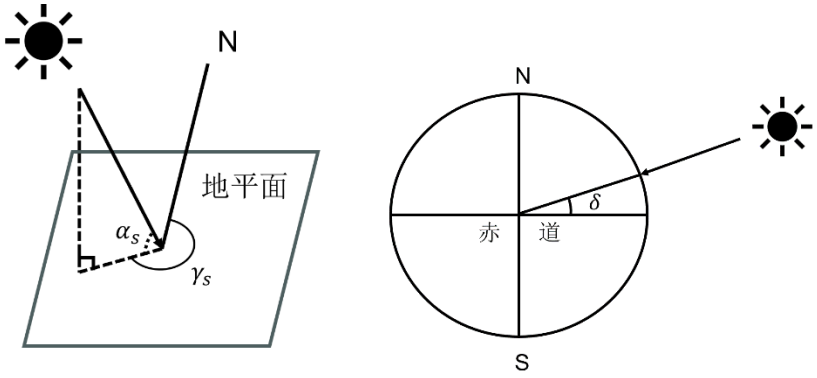

图 2 太阳高度角、方位角、赤纬角示意图

在镜场坐标系中，太阳主光线入射光线的单位向量为：

$$I_{sun}=\left(\cos\alpha_S\sin\gamma_S,\cos\alpha_S\cos\gamma_S,\sin\alpha_S\right) \tag{5-5}$$

在问题一中，吸收塔的位置坐标以及所有定日镜的位置和安装高度是确定的，

且控制系统会使得太阳中心点发出的光线经定日镜中心反射后指向集热器中心。综上所述，镜场坐标系中太阳主光线反射光线的单位向量为：

$$R_{sun}=\left(\frac{-x_i}{\sqrt{x_i^2+y_i^2+(H_t-z_i)^2}},\frac{-y_i}{\sqrt{x_i^2+y_i^2+(H_t-z_i)^2}},\frac{H_t-z_i}{\sqrt{x_i^2+y_i^2+(H_t-z_i)^2}}\right) \quad (5\text{-}6)$$

其中，H_t 为吸收塔的高度，(x_i,y_i,z_i) 为第 i 个定日镜的坐标和安装高度。

根据入射单位向量和反射单位向量，可以导出第 i 个定日镜的法向向量为

$$n_i=\vec{I}_{sun}+\vec{R}_{sun} \quad (5\text{-}7)$$

5.1.2 余弦效率与大气透射率模型的建立

余弦损失是指由于太阳光入射方向与镜面采光口法线方向不平行引起的接收能量损失，在反射时镜面必然无法与入射光线垂直，只有当镜面与光线垂直时，实际反射面积才等同于镜面面积。其损失大小与入射方向和法向方向的夹角余弦值相关，具体表达式为：

$$\eta_{\cos}=\cos[\frac{1}{2}\cdot\arccos(-I_{sun}\cdot R_{sun})] \quad (5\text{-}8)$$

太阳光在空气传播的过程中，由于空气中的杂质会对光线造成一定程度的削弱，导致一部分能量损失。根据题目信息，其损失大小与定日镜和吸收塔的距离有关，其具体表达式如下[2]。

$$\eta_{\text{at}}=0.99321-0.0001176d_{\text{HR}}+1.97\times10^{-8}\times d_{\text{HR}}^2 \quad (5\text{-}9)$$

其中，d_{HR} 表示镜面中心到集热器中心的距离。

5.1.3 光锥能流密度模型的建立

由于光线的发散性，太阳光的入射光线是一束锥形光束，假设其半角展宽为4.65mrad[3]，并以主光线的单位向量为垂直参考。在入射光线的光锥圆盘中，能量的分布并不均匀，由内向外逐渐减少。本文采用常用的二维正态分布模型[4]去描述光锥圆盘面的能流密度,从而能够模拟出光线入射的非平行性。以圆盘平面与主光线的交点为原点，主光线为 z 轴，建立光锥直角坐标系。

假设其在 x-y 方向上满足二维正态分布，且相互独立、方差均为σ，得到二

维正态分布函数如下：

$$f(x,y)=\frac{1}{2\pi\sigma^2}\exp[-\frac{1}{2}\frac{\left(x^2+y^2\right)}{\sigma^2}] \tag{5-10}$$

其中，以光锥圆盘面的半径(对应半角展宽为 4.65mrad)作为3σ。

根据二维正态分布的结果，越靠近圆心的位置，点数越密集，相应的能量高，光线的数目多，光线朝该方向的概率就越大；靠近圆盘边缘则点数就越稀少，相应的能量低，光线的数目少，光线朝该方向的概率就越小。

在二维的正态分布结果中随机地选取一个点，将其和原点链接形成偏移向量t，用于描述非平行光线和主光线之间的偏移。将主光线的入射单位向量与其做矢量叠加，能够得到在光锥中偏移过后的非平行光线的单位向量[5]。

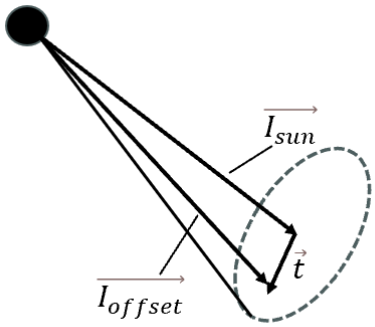

图 3 非平行光线示意图

$$I_{offset}=\frac{\vec{t}+\vec{I}_{sun}}{\left|\vec{t}+\vec{I}_{sun}\right|} \tag{5-11}$$

其中，I_{sun}为太阳主光线的单位向量，详见式(5-5)。

5.1.4 阴影遮挡效率模型的建立

阴影挡光损失主要包括三部分[3]:

①吸收塔的阴影遮挡定日镜造成的阴影损失。

②后排定日镜接收的入射光线被前方定日镜所阻挡造成的阴影损失。

③后排定日镜的反射光线被前方定日镜阻挡而未到达吸热器所造成的挡光损失。

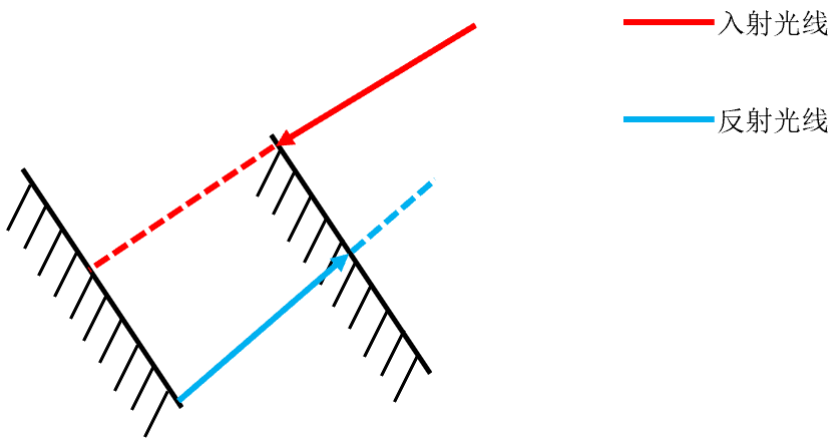

图 4 阴影遮挡示意图

本模型采用光线追迹法去刻画阴影遮挡带来的能量损失。通过追迹每一个太阳光入射光线的运动轨迹，计算经镜面反射后最终到达集热器中心的光线数目来得到阴影遮挡效率。

通过入射光线(或反射光线)向量以及定日镜上点的坐标可以得到一条唯一的直线方程。由于定日镜为平面矩形且上下两边始终和地面平行，通过定日镜的法向向量和定日镜中心点可以确定一个唯一平面。将直线方程和平面方程联立得到点的坐标，即可判断光线是否被遮挡。相似地，将直线和吸收塔的边界方程联立，即可判断入射光线是否被遮挡。

根据 5.1.3 节中建立的模型，采用蒙特卡洛方法随机地选取大量的点，可以引入不同方向的非平行光线。在定日镜中均匀地选取若干个点，由任意一点和不同方向的非平行光线组成若干个入射光线。计算所有入射光线所对应的反射光线，判断入射光线和反射光线是否被其他定日镜或者吸收塔所遮挡，若不被遮挡，则求取入射光线和反射光线最终与集热器所在平面的交点，判断该交点是否在集热器内，即可计算出单个定日镜的阴影遮挡效率。

$$\eta_{\mathrm{sb}} = 1 - \frac{N_I + N_R}{N_g} \tag{5-12}$$

其中，N_I 为该定日镜被遮挡的入射光线的数目，N_R 为该定日镜被遮挡的反射光线的数目，N_g 为总光线的数目。

5.1.5 集热器截断效率模型的建立

截断损失主要由于集热器的尺寸有限，导致部分反射光线照射在集热器之外，造成了能量的损失。

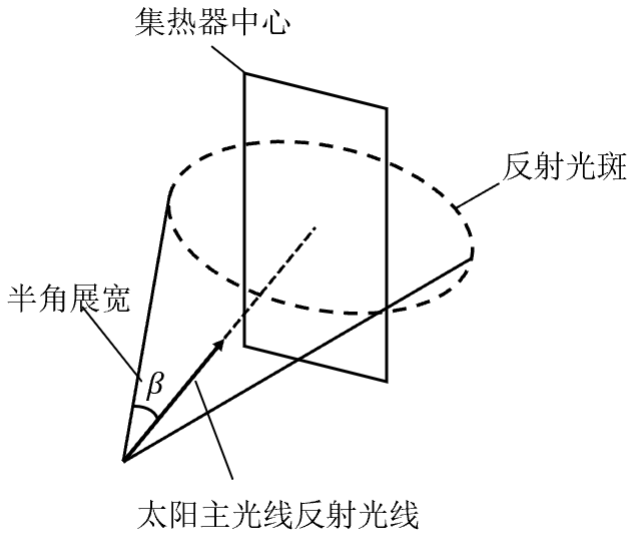

图 5 截断效率示意图

根据 5.1.4 节建立的模型，继续沿用光线追迹法的思路，计算经镜面反射后最终到达集热器中心的光线数目来得到截断效率。通过反射光线的单位向量和定日镜上点的坐标可以确定一条直线，将直线与集热器的方程联立得到交点坐标(不能从下方射入底面)，即可判断反射光线是否照射在集热器上。

$$\eta_{trunc}=1-\frac{N_{\text{int}}}{N_g-N_I-N_R} \tag{5-13}$$

其中，N_{int} 为照射在集热器上的光线总数。

5.1.6 年平均光学效率、年平均输出热功率、单位镜面面积年平均输出热功率模型的建立

根据前几节建立的模型，我们可以得到定日镜的光学效率的公式为：

$$\begin{aligned}\eta_i^{(jk)} &= \eta_{\text{sb}}\eta_{\cos}\eta_{\text{at}}\eta_{\text{trunc}}\eta_{\text{ref}} \\ &= 0.92\times\cos[\frac{1}{2}\cdot\arccos(-I_{sun}\cdot R_{sun})]\times(1-\frac{N_{\text{int}}}{N-N_I-N_R}) \\ &\times(0.99321-0.0001176d_{\text{HR}}+1.97\times10^{-8}\times d_{\text{HR}}^2)\times(1-\frac{N_I+N_R}{N})\end{aligned} \tag{5-14}$$

其中，$\eta_i^{(jk)}$为第 i 个定日镜在第 j 个月的 21 日的第 k 个时间段的光学效率，η_{ref}为常数，取为 0.92，其他效率公式详见 5.1.2-5.1.5 节。

可以得到年平均光学效率为：

$$\overline{\eta} = \frac{\sum_{i}^{N}\sum_{j=1}^{12}\sum_{k=1}^{5}\eta_i^{(jk)}}{60\times N} \tag{5-15}$$

其中，N 为定日镜的总数量，5 代表每日的 5 个时间，12 代表 12 个月份。根据式(5-15)可以推导不同效率的年平均值。

本模型用法向直接辐射辐照度 DNI 表示地球上垂直于太阳光线的平面单位面积上、单位时间内接收到的太阳辐射能量，其表达式为[6]：

$$\begin{aligned}&\text{DNI} = G_0\left[a+b\exp\left(-\frac{c}{\sin\alpha_s}\right)\right] \\ &a = 0.4237-0.00821(6-H)^2 \\ &b = 0.5055+0.00595(6.5-H)^2 \\ &c = 0.2711+0.01858(2.5-H)^2\end{aligned} \tag{5-16}$$

其中，G_0为太阳常数，其值取为 1.366 kW/m^2，H为海拔高度为 3km。

因此，我们可以得到第 j 个月的 21 日的第 k 个时间段定日镜场的输出热功率为：

$$E_{\text{field}}^{(jk)} = \text{DNI}\cdot\sum_{i}^{N}A_i\eta_i^{(jk)} \tag{5-17}$$

其中，DNI 为法向直接辐射辐照度；N为定日镜总数（单位：面）；A_i为第i面定日镜采光面积（单位：m^2）；$\eta_i^{(jk)}$为第 i 个定日镜在第 j 个月的 21 日的第 k 个时间段的光学效率。

由此得到年平均输出热功率为：

$$\bar{E}_{\text{field}} = \frac{\sum_{j=1}^{12}\sum_{k=1}^{5}\sum_{i}^{N}\text{DNI}\cdot A_i\eta_i^{(jk)}}{60} \tag{5-18}$$

单位镜面面积年平均输出热功率为：

$$\bar{E}_{\text{unit}} = \frac{\sum_{j=1}^{12}\sum_{k=1}^{5}\sum_{i}^{N}\text{DNI}\cdot A_i\eta_i^{(jk)}}{60\times\sum_{i}^{N}A_i} \tag{5-19}$$

5.1.7 问题一模型的求解与分析

基于上述建立的模型，采用蒙特卡洛方法生成光锥中的非平行线，通过联立方程组的方法判断遮挡关系，最终的计算结果如下表所示。

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/m^2)
1 月 21 日	57.25%	71.95%	91.25%	99.89%	0.500
2 月 21 日	60.39%	74.00%	93.05%	99.81%	0.570
3 月 21 日	62.45%	76.06%	93.39%	99.72%	0.622
4 月 21 日	64.24%	77.86%	93.56%	99.76%	0.661
5 月 21 日	65.23%	78.84%	93.60%	99.79%	0.682
6 月 21 日	65.53%	79.13%	93.61%	99.82%	0.688
7 月 21 日	65.20%	78.81%	93.60%	99.79%	0.681
8 月 21 日	64.11%	77.74%	93.54%	99.71%	0.659
9 月 21 日	62.28%	75.89%	93.37%	99.73%	0.618
10 月 21 日	60.00%	73.67%	92.91%	99.84%	0.561
11 月 21 日	56.84%	71.73%	90.96%	99.92%	0.491
12 月 21 日	55.39%	71.05%	89.82%	99.91%	0.461

表 2 问题一 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)	单位面积镜面平均输出热功率(kW/m^2)
61.58%	75.56%	92.72%	99.80%	37.629	0.599

根据求解结果，年平均光学效率为 61.58%，其中各个效率均在 6 月 21 日达到最大值这与太阳高度角和太阳方位角相关。截断效率较高，几乎接近于 1，说明反射光学的光锥基本在集热器范围内。余弦效率较低，是导致年平均光学效率低下的主要的影响因素。由于吸收塔位于场的中心，部分定日镜的太阳光入射的方向和反射方向的夹角较大，导致平均过后整体效率降低。此外，年平均光学效率也受大气折射率影响，在离吸收塔较远处的定日镜大气折射率较低，提供的能量少。

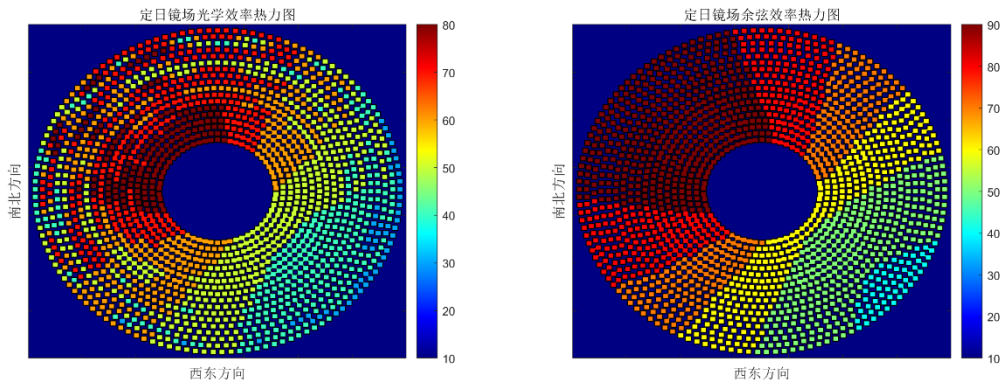

图 6 问题一定日镜场光学效率热力图

图 6 为定日镜场的光学效率热力图，其中以吸热塔为原点，横轴为 x 轴代表正东方向，纵轴为 y 轴代表正北方向。由于问题一中每个定日镜的安装高度和尺寸均为一致的，则光学效率可以反映定日镜输出功率之间的关系。由图 5 可知，在西北向的定日镜光学效率普遍高于东南向的定日镜，且二者之间有较明显的分界线。余弦效率和总光学效率的分布情况基本一致，且余弦效率的分布更加规整，分界线更加明显，在西北向的定日镜法向和太阳入射光线的夹角小，对应的余弦效率大，相应的总光学效率大。因此，余弦效率是影响镜场光学效率的重要因素。

5.2 问题二模型的建立与求解

5.2.1 确定约束条件

根据问题二的要求，在定日镜场在达到额定功率的条件下，尽可能地增大单位镜面面积年平均输出热功率。令(X_t,Y_t)表示吸收塔的位置坐标，l_i表示定日镜的镜面高度，w_i表示定日镜的镜面宽度，N为定日镜的总数，(x_i,y_i,z_i)为第 i 个定日镜的位置和安装高度，通过调整以上的参数来提升平均输出热功率。根据题目的要求，有以下几个约束条件：

(1)平均输出热功率要达到额定功率

$$\bar{E}_{\text{field}} \geq 60\text{MW} \tag{5-20}$$

(2)吸收塔周围 100m 内不安装定日镜，且定日镜和吸收塔均在定日镜场内

$$\begin{cases} \sqrt{(x_i-X_t)^2+(y_i-Y_t)^2} \geq 100\text{m} \\ \sqrt{x_i^2+y_i^2}, \sqrt{X_t^2+Y_t^2} \leq 350\text{m} \end{cases} \tag{5-21}$$

(3)通常镜面宽度不小于镜面高度，镜面边长在 2 m 至 8 m 之间，相邻定日镜底座中心之间的距离比镜面宽度多 5 m 以上，即

$$\begin{cases} w_i \geq l_i \\ 2\text{m} \leq w_i, l_i \leq 8\text{m} \\ \sqrt{(x_i-x_j)^2+(y_i-y_j)^2} - \max(w_i,w_j) \geq 5 \end{cases} \tag{5-22}$$

其中，*i*和*j*表示相邻的两个定日镜，$\max(w_i,w_j)$表示两个相邻的定日镜中最大的镜面宽度。

(4)安装高度在 2 m 至 6 m 之间，安装高度必须保证镜面在绕水平转轴旋转时不会触及地面

$$\begin{cases} 2\text{m} \leq z_i \leq 6\text{m} \\ \dfrac{l_i}{2} \leq z_i \end{cases} \tag{5-23}$$

5.2.2 同心圆密排布局方式

为了尽可能提高年平均输出热功率，参考 Campo 布置方法[7]，提出一种同心圆密排布局方式。假设定日镜围绕着吸收塔呈同心圆状的分布，且同一个同心圆上定日镜的间距是均匀的。根据约束条件：

$$\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} - \max(w_i, w_j) \geq 5 \tag{5-24}$$

可以得到相邻两个定日镜的最小安全距离为：

$$\Delta R_{\min} = \max(w_i, w_j) + 5 \tag{5-25}$$

其中，i和j表示相邻的两个定日镜，$\max(w_i, w_j)$表示两个相邻的定日镜中最大的镜面宽度。

为了尽可能地提升平均输出热功率，在同心圆密排布局中，令同一圆上的定日镜间距和不同同心圆之间的间距均为最小安全距离，并在不同的同心圆之间引入一定的偏移角度使其交错排列。通过该布局，能够建立定日镜尺寸和定日镜位置、数目的关系，改变尺寸即可改变定日镜的位置和数目，减少参数的数量。

建立定日镜尺寸和第n个同心圆上定日镜数量N_n的关系如下：

$$N_n = \frac{2\pi r_n}{\Delta R_{\min}} \tag{5-26}$$

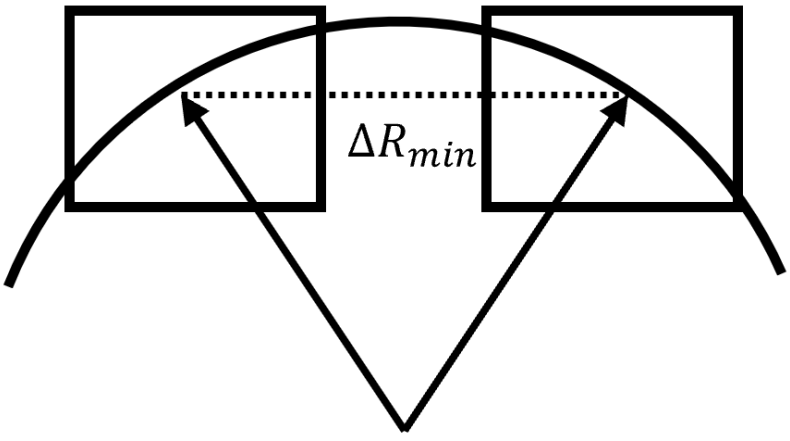

图 7 最小安全距离示意图

5.2.3 单位镜面面积年平均输出热功率优化模型的建立

根据 5.2.1 节确定的约束条件，建立单位面积年平均输出热功率优化模型如下：

$$\max(\bar{E}_{unit})$$

$$s.t.\begin{cases}\bar{E}_{\text{field}} \geq 60\text{MW} \\ \sqrt{(x_i - X_t)^2 + (y_i - Y_t)^2} \geq 100\text{m} \\ \sqrt{x_i^2 + y_i^2}, \sqrt{X_t^2 + Y_t^2} \leq 350\text{m} \\ \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} - \max(w_i, w_j) \geq 5 \\ w_i \geq l_i \\ \dfrac{l_i}{2} \leq z_i \\ 2\text{m} \leq w_i, l_i \leq 8\text{m} \\ 2\text{m} \leq z_i \leq 6\text{m}\end{cases} \tag{5-27}$$

其中，i和j表示相邻的两个定日镜，$\max(w_i, w_j)$表示两个相邻的定日镜中最大的镜面宽度。

5.2.4 双层规划模型的建立

在求最大的单位面积年平均热输出功率时，首先要提升年平均输出热功率。由于本问题涉及的参数较多，普通的模型和算法根本无法给出结果和方案，因此本文建立双层规划模型，分步骤去得到最大单位面积年平均输出热功率。

由式(5-17)：

$$E_{\text{field}}^{(jk)} = \text{DNI} \cdot \sum_{i}^{N} A_i \eta_i^{(jk)}$$

在定日镜场的输出热功率不变的情况下，光学效率越高，镜子的总面积就越小。为了得到最大的单位面积年平均热输出功率，一定程度上要提高光学效率。在参数中，吸收塔的位置尤为关键，它对整个定日镜场的平均光学效率有重要影响。

(1)吸收塔规划模型的建立

由于吸收塔只通过影响光学效率进而影响最后的年平均输出热功率。故建立吸收塔规划模型即上层规划，以优化最大的年平均光学效率为目标，以吸收塔的位置作为决策变量，最终确定吸收塔的位置，并将规划方案传递给下层模型。其具体的优化函数如下：

$$\max(\overline{\eta})$$

$$s.t.\begin{cases}\sqrt{(x_i-X_t)^2+(y_i-Y_t)^2}\geq 100\text{m}\\ \sqrt{X_t^2+Y_t^2}\leq 350\text{m}\end{cases} \tag{5-28}$$

其中，$\overline{\eta}$ 为年平均光学效率，详见式(5-15)，(X_t,Y_t) 为吸收塔的位置，是本上层规划中的决策变量。

(2)定日镜规划模型的建立：

在上层规划给出吸收塔的位置后，下层规划以优化最大年均输出热功率为目标，以定日镜的位置、数量、尺寸、安装高度等参数为决策变量，用控制变量法进行求解。并将规划后的结果反馈给上层，使上层的模型根据更优的定日镜的尺寸、位置、数量、安装高度继续规划，反复迭代后找到最优解。

其具体的优化函数如下：

$$\max(\overline{E}_{unit})$$

$$s.t.\begin{cases}\overline{E}_{\text{field}}\geq 60\text{MW}\\ \sqrt{(x_i-X_t)^2+(y_i-Y_t)^2}\geq 100\text{m}\\ \sqrt{x_i^2+y_i^2},\sqrt{X_t^2+Y_t^2}\leq 350\text{m}\\ \sqrt{(x_i-x_j)^2+(y_i-y_j)^2}-\max(w_i,w_j)\geq 5\\ w_i\geq l_i\\ \dfrac{l_i}{2}\leq z_i\\ 2\text{m}\leq w_i,l_i\leq 8\text{m}\\ 2\text{m}\leq z_i\leq 6\text{m}\end{cases} \tag{5-29}$$

其中，$\overline{E}_{\text{field}}$ 为年均输出热功率，详见式(5-18).

在双层规划中，上层规划确定了吸收塔的具体位置，并传递给下层；下层在上层的规划基础上，对定日镜的参数进行优化，并将优化结果反馈给上层，上层

规划再根据反馈调整吸收塔的位置。多次循环迭代可以让年平均热输出功率不断提高，直到其达到额定功率。

具体的规划流程如下图所示：

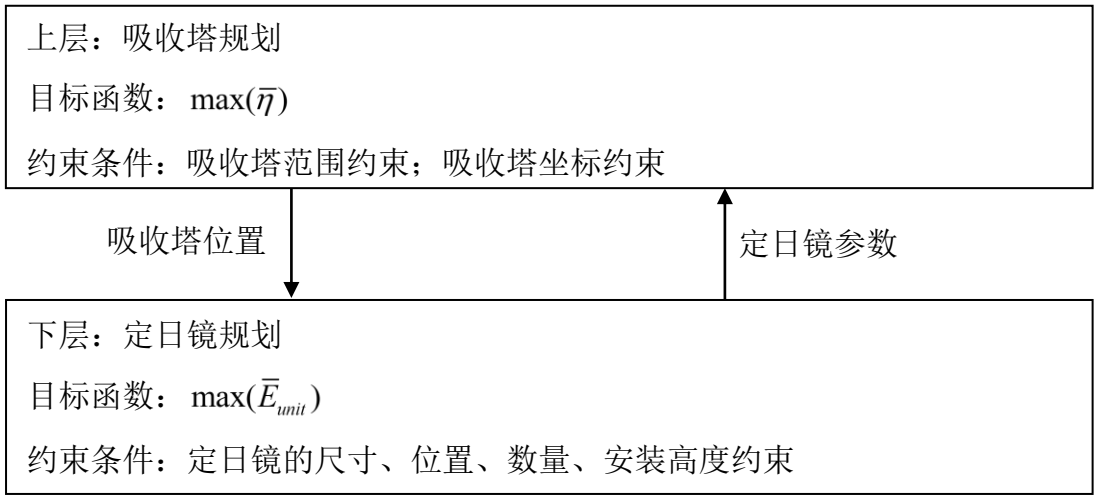

图 8 双层规划框架图

5.2.5 问题二模型的求解与分析

在问题二中，我们采用同心圆密排的方式进行布局，令同一圆上的定日镜间距和不同同心圆之间的间距均为最小安全距离，并在不同的同心圆之间引入一定的偏移角度使其交错排列。能够有效地减少参数数量，提高计算效率。

Step1:确定吸收塔位置

在求解上层模型的过程中，假设定日镜围绕着吸收塔呈同心圆状的分布，且同一个同心圆上定日镜的间距是均匀的。以吸收塔为圆心，以不同的同心圆半径 r_n 做 n 个同心圆。按照同心圆密排的方式排列定日镜。通过遍历算法，在镜场坐标系上均匀取点，计算不同吸收塔位置下的年平均光学效率，找到最优位置。

Step2:遍历所有参数

在得到上层模型提供的吸收塔位置后，下层模型采用控制变量法的思想，按顺序遍历所有参数

$$z_i \to l_i \to w_i \to l_i \to z_i \tag{5-30}$$

其中， z_i 表示安装高度， l_i 表示镜面高度， w_i 表示镜面宽度。

由于安装高度和镜面尺寸相互耦合影响，按以上顺序使用遍历算法，能够实现更好的优化效果。把信息反馈给上层，回到 Step1.

Step3:求解单位镜面面积年平均输出热功率优化模型

重复上述过程，直到求出最大的单位镜面面积年平均热输出功率。得到的结果如下所示。

表 3 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/m^2)
1 月 21 日	58.45%	83.98%	80.28%	99.90%	0.510
2 月 21 日	61.09%	84.45%	83.05%	99.97%	0.576
3 月 21 日	61.95%	84.51%	83.84%	99.97%	0.616
4 月 21 日	62.41%	83.96%	84.65%	100%	0.642
5 月 21 日	61.09%	83.15%	85.21%	100%	0.651
6 月 21 日	61.95%	82.76%	85.41%	100%	0.653
7 月 21 日	62.41%	83.19%	85.18%	100%	0.651
8 月 21 日	62.42%	84.02%	84.62%	100%	0.641
9 月 21 日	61.90%	84.53%	83.78%	99.98%	0.613
10 月 21 日	60.79%	84.40%	82.75%	99.98%	0.567
11 月 21 日	58.01%	83.91%	79.79%	99.90%	0.501
12 月 21 日	56.23%	83.67%	77.77%	99.93%	0.468

表 4 问题二 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)	单位面积镜面平均输出热功率(kW/m^2)
60.87%	83.88%	83.03%	99.97 %	60.023	0.591

表 5 问题二 设计参数表

吸收塔位置坐标	定日镜尺寸（宽×高）	定日镜安装高度 (m)	定日镜总面数	定日镜总面积 (m^2)
(48.2,-126)	5.75×5.75	2.875	3070	101501.8

最终得到的单位面积镜面平均输出热功率为 0.591kW/m^2，得到的布局结果如下。

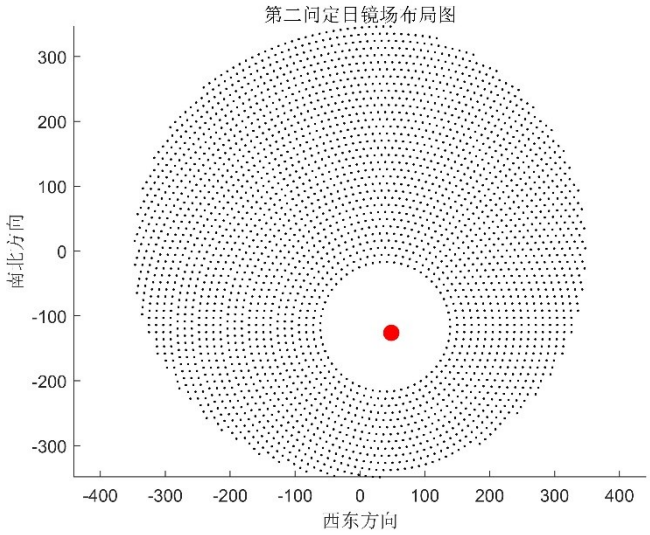

图 9 定日镜场布局图

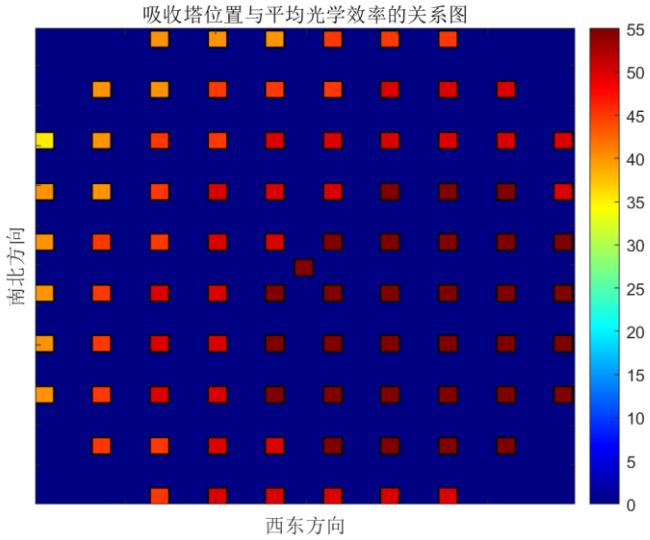

图 10 吸热塔位置与平均光学效率关系图

图中 x 轴表示正东方向，y 轴表示正北方向。由图可知当吸热塔放在东南方向时，年平均光学效率较高，这是由余弦效率造成的。

在问题二的求解过程中，我们采用不同的初始点，最后定日镜的尺寸和安装高度均收敛至 5.75×5.75 和 2.875m 附近，一定程度上反映了模型的稳定性，如果取更小的步长，能够得到更加精确的结果，但计算效率会显著下降。根据结果，我们在光学效率和单位面积镜面平均输出热功率和问题一基本保持的情况下，到达了额定功率 60MW。

5.3 问题三模型的建立与求解

5.3.1 问题三模型的建立

问题三在问题二的基础上添加了新的条件，不同的定日镜尺寸和安装高度可以各不相同。在定日镜场在达到额定功率的条件下，尽可能地增大单位镜面面积年平均输出热功率。继续沿用在问题二中提出的模型，把同心圆半径 r_n 把定日镜进行分组。假设在同一个同心圆上的所有定日镜的尺寸(宽×高)和安装高度是相同的，记为 $w_i^{(n)}, l_i^{(n)}$ 和 $z_i^{(n)}$ 。

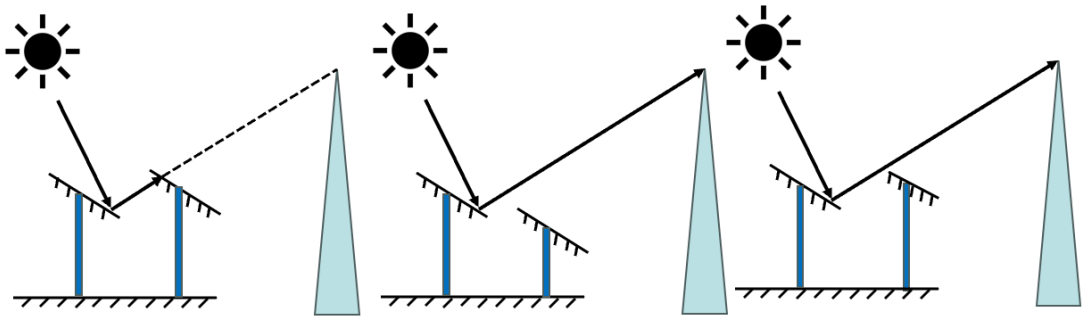

图 11 定日镜安装高度、尺寸变化示意图

由图可知，通过调整不同定日镜的安装高度和尺寸可以提高光学效率和热输出功率。若离吸收塔较远的定日镜安装高度高，离吸收塔较近的定日镜尺寸较小，吸收塔获得的能量是最高的。在真实的定日镜场的布局中，近塔区的定日镜密集，远塔区的定日镜稀疏[8]，根据问题二得到的结果和以上分析，建立 $w_i^{(n)}, l_i^{(n)}$ 和 $z_i^{(n)}$ 与同心圆半径的函数关系：

$$\begin{cases} w_i^{(n)} = A \cdot (r_n - 100) + 5.75 \\ l_i^{(n)} = B \cdot (r_n - 100) + 5.75 \\ z_i^{(n)} = C \cdot (r_n - 100) + 2.875 \end{cases} \tag{5-31}$$

其中， A,B,C 为斜率，是模型需要优化的参数，均为正数；常数为截距，表示 $w_i^{(n)},l_i^{(n)}$和$z_i^{(n)}$在第二问的结果。

仍采用问题二建立的同心圆密排布局和双层规划模型，得到修改后的模型如下：

$$\max(\bar{E}_{unit})$$

$$s.t.\begin{cases}\bar{E}_{\text{field}} \geq 60\text{MW} \\ \sqrt{(x_i^{(n)}-X_t)^2+(y_i^{(n)}-Y_t)^2} \geq 100\text{m} \\ \sqrt{(x_i^{(n)})^2+(y_i^{(n)})^2},\sqrt{X_t^{\,2}+Y_t^2} \leq 350\text{m} \\ \sqrt{(x_i^{(n)}-x_j^{(n)})^2+(y_i^{(n)}-y_j^{(n)})^2}-\max(w_i^{(n)},w_j^{(n)}) \geq 5 \\ w_i^{(n)} \geq l_i^{(n)} \\ \dfrac{l_i^{(n)}}{2} \leq z_i^{(n)} \\ 2\text{m} \leq w_i^{(n)},l_i^{(n)} \leq 8\text{m} \\ 2\text{m} \leq z_i^{(n)} \leq 6\text{m}\end{cases} \tag{5-32}$$

其中，上标n表示该定日镜位于第n组同心圆，公式中其他变量的具体含义详见式(5-27).

5.3.2 问题三模型的求解与分析

根据问题二和问题三中建立的模型，继续采用双层规划和遍历算法进行求解，得到的结果如下所示。

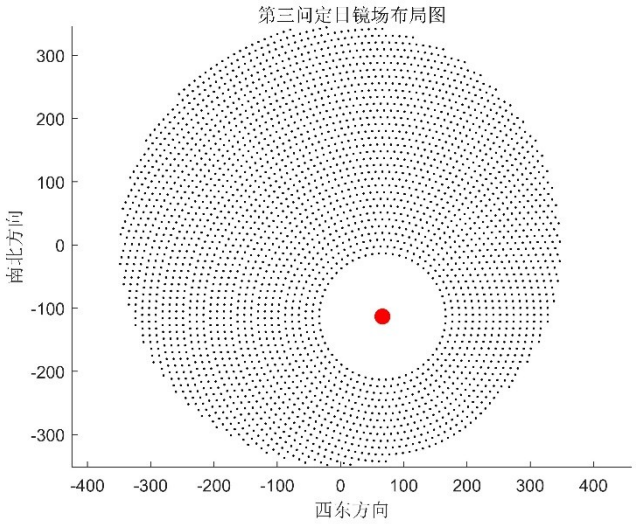

图 12 问题三定日镜场布局图

表 6 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/m^2)
1 月 21 日	58.53%	83.71%	80.68%	99.88%	0.511
2 月 21 日	61.25%	84.35%	83.38%	99.96%	0.578
3 月 21 日	62.24%	84.60%	84.16%	99.99%	0.620
4 月 21 日	62.79%	84.24%	84.89%	100.00%	0.646
5 月 21 日	62.83%	83.57%	85.41%	100.00%	0.657
6 月 21 日	62.78%	83.23%	85.60%	100.00%	0.659
7 月 21 日	62.84%	83.60%	85.39%	100.00%	0.656
8 月 21 日	62.77%	84.29%	84.83%	100.00%	0.645
9 月 21 日	62.17%	84.60%	84.09%	99.99%	0.616
10 月 21 日	60.96%	84.27%	83.12%	99.99%	0.569
11 月 21 日	58.11%	83.62%	80.23%	99.92%	0.502
12 月 21 日	56.32%	83.33%	78.26%	99.95%	0.469

表 7 问题三 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)	单位面积镜面平均输出热功率(kW/m^2)
61.13%	83.95%	83.34%	99.97%	60.01	0.594

表 8 问题三 设计参数表

吸收塔位置坐标	定日镜尺寸（宽×高）	定日镜安装高度 (m)	定日镜总面数	定日镜总面积 (m^2)
(66.25,-113.23)			3075	101016.98

通过第三问的模型，成功把单位面积镜面的平均输出热功率提高到了 0.594kW/m^2.具体的尺寸和高度详见支撑材料。

6 模型的评价

6.1 模型的优点

1. 采用蒙特卡洛光线追迹法，基于物理原理准确地模拟光线的传播过程，适用于复杂的定日镜效率计算。
2. 通过二维正态分布体现出太阳光的非平行性，更加贴合实际情况。
3. 通过双层规划模型来提升年均输出热功率，将复杂的多参数问题进行拆分，降低了计算的复杂度。
4. 参考 Campo 定日镜场的布置方法，提出了同心圆密排布局方式，建立同心圆半径和定日镜位置、数量的关系，能够极大程度地简化计算的复杂性。

6.2 模型的缺点

1. 蒙特卡洛光线追踪方法和遍历算法对计算资源的要求高，计算成本大。若在蒙特卡洛方法中取得光线数目较少，会导致光学效率有较大的误差。
2. 本文假设同一个同心圆上定日镜的分布是均匀的，定日镜的所有参数是相同的，得到的解为局部最优解，实际情况中不同的定日镜会存在一定的偏差。
3. 本文在问题三中用线性关系去描述定日镜尺寸和安装高度与同心圆半径的关系，只能得到局部最优解，需要更加精确的函数。

6.3 模型的改进

1. 本文采用的等步长遍历算法去求解模型，在求解过程中观察到变量对输出热功率的影响均为线性变化，可以采用二分法遍历算法，能够极大地提升模型的计算效率。

7 参考文献

[1] 蔡志杰，太阳影子定位[J]，数学建模及其应用，2015，4(4):25-33.

[2] O. Farges, J.J. Bezian, M. El Hafi, Global optimization of solar power tower systems

using a Monte Carlo algorithm: Application to a redesign of the PS10 solar thermal power plant [J], Renewable Energy, 2018, 119:345-353.

[3] 张平等，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):5-8.

[4] 张宏丽,王志峰.塔式电站定日镜场布置范围的理论分析[J].太阳能学报,2011,32(01):89-94.

[5] 周艺艺,田军,陈将,赵豫红.基于 GPU 的塔式太阳能热电系统吸热功率计算[J].控制工程,2015,22(02):282-286.

[6] 杜宇航等，塔式光热电站定日镜不同聚焦策略的影响分析[J]，动力工程学报，2020， 40(5):426-432.

[7] 高博,刘建兴,孙浩,刘二林.基于自适应引力搜索算法的定日镜场优化布置[J].太阳能学报,2022,43(10):119-125.

[8] 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(06):690-698

8 附录

附录 1：支持材料列表
1.问题一源程序 2.问题二源程序 3.问题三源程序 4.结果

附录 2：Matlab 代码

8.1 问题一用到的程序：

程序一：问题一主程序

a_q1.m

```
clear;
clc;

excelFileName = 'data1.xlsx';

% 集热塔位置信息
collector_tower = [48.2369 -126.0147   80.0000];
% 定日镜位置信息
positions(:, 1:2) = concentric_circles(collector_tower, 100, 100,
5+5.75);
data_len = size(positions, 1);
% 定日镜安装高度
positions(:, 3) = 2.8750 .* ones(data_len, 1);
% 定日镜高度
heights = 5.75 .* ones(data_len, 1);
% 定日镜宽度
widths = 5.75 .* ones(data_len, 1);



% 绘制散点图
scatter3(positions(:, 1), positions(:, 2), positions(:, 3), 10, 'b',
'filled');
hold on;
scatter3(0, 0, 80, 50, 'red', 'filled');

dates = {'2023-1-21', '2023-2-21', '2023-3-21', '2023-4-21', '2023-5-21',
'2023-6-21', ...
```

```
        '2023-7-21', '2023-8-21', '2023-9-21', '2023-10-21', '2023-11-
21', '2023-12-21'};
times = {'9:00:00', '10:30:00', '12:00:00', '13:30:00', '15:00:00'};
datetimeMatrix = repmat(datetime('now'), 12, 5);
for i = 1:length(dates)
    for j = 1:length(times)
        % 合并日期和时间，然后转换为 datetime 对象
        datetimeString = [dates{i} ' ' times{j}];
        datetimeMatrix(i, j) = datetime(datetimeString, 'InputFormat',
'yyyy-MM-dd HH:mm:ss');
    end
end

energy = cell(12, 5);
eta = cell(12, 5);
eta_sb = cell(12, 5);
eta_cos = cell(12, 5);
eta_trunc = cell(12, 5);
eta_at = cell(12, 5);
for j = 1:12
    for i = 1:5
        [~, energy{j, i}, eta{j, i}, eta_sb{j, i}, eta_trunc{j, i},
eta_cos{j, i}, eta_at{j, i}] = ...
            caculate(datetimeMatrix(j, i), positions, heights, widths,
collector_tower);
        fprintf('%d %d\n', j, i);
    end
end

sum_energy = 0;
energy_month = zeros(12, 1);
avg_eta = zeros(12, 1);
avg_eta_at = zeros(12, 1);
avg_eta_cos = zeros(12, 1);
avg_eta_sb = zeros(12, 1);
avg_eta_trunc = zeros(12, 1);
for j = 1:12
    for i = 1:5
        avg_eta(j) = avg_eta(j) + mean(eta{j, i});
        avg_eta_at(j) = avg_eta_at(j) + mean(eta_at{j, i});
        avg_eta_cos(j) = avg_eta_cos(j) + mean(eta_cos{j, i});
        avg_eta_sb(j) = avg_eta_sb(j) + mean(eta_sb{j, i});
        avg_eta_trunc(j) = avg_eta_trunc(j) + mean(eta_trunc{j, i});
        sum_energy = sum_energy + energy{j, i};
```

```
        energy_month(j) = energy_month(j) + energy{j, i};
        if ~isreal(energy{j, i})
            fprintf('%d %d\n', j, i);
        end
    end
    avg_eta(j) = avg_eta(j) / 5;
    avg_eta_at(j) = avg_eta_at(j) / 5;
    avg_eta_cos(j) = avg_eta_cos(j) / 5;
    avg_eta_sb(j) = avg_eta_sb(j) / 5;
    avg_eta_trunc(j) = avg_eta_trunc(j) / 5;
    energy_month(j) = energy_month(j) / 5;
end
disp(sum_energy / 60 * 1000)

% figure;
% % 绘制散点图
% scatter3(positions(:, 1), positions(:, 2), 2 * eta_sb, 10, 'b',
'filled');
% hold on;
% scatter3(0, 0, 3, 50, 'red', 'filled');
% quiver3(0, 0, 3, sun_direction(1) * 100, sun_direction(2) * 100,
sun_direction(3) * 1);
% xlabel('X');
% ylabel('Y');
% zlabel('Z');
% axis([-400, 400, -400, 400, 0, 4])
```

程序二:角度计算函数

```
function [unit_energy, energy, eta, eta_sb, eta_trunc, eta_cos, eta_at] =
caculate(date, positions, heights, widths, collector_tower)

    data_len = size(positions, 1);
    % 计算太阳高度角，太阳方位角，太阳主光线方向
    [solar_elevation_deg, sun_azimuth_deg, sun_direction] =
sun_angle(98.5, 39.4, date);
    sun_direction = real(sun_direction);

    % 定日镜法向量（每次都要重新计算）
    norm_directions = zeros(data_len, 3);
    % 定日镜四个角的坐标（每次都要重新计算）
    vertex_data = zeros(4, 3, data_len);
```

```
    % 计算每个定日镜的法向向量和四角坐标
    for i = 1:data_len
        norm_directions(i, :) = normal_direction(positions(i, :),
sun_direction, collector_tower);
        [vertex_data(1, :, i), vertex_data(2, :, i), vertex_data(3, :, i),
vertex_data(4, :, i),] = ...
            vertex(positions(i, :), heights(i), widths(i),
norm_directions(i, :));
    end

    % 计算每个定日镜的 阴影遮挡效率 和 集热器截断效率 和 余弦效率 和 大气透射率
    eta_sb = zeros(data_len, 1);
    eta_trunc = zeros(data_len, 1);
    eta_cos = zeros(data_len, 1);
    eta_at = zeros(data_len, 1);
    for i = 1:data_len
        [eta_sb(i), eta_trunc(i), eta_cos(i), eta_at(i)] =
eta_caculate(positions, i, sun_direction, norm_directions(i, :),
collector_tower, vertex_data);
    end


    % 镜面反射率
    eta_ref = 0.92;

    % 计算 DNI
    G0 = 1.366; % 太阳常数
    H = 3; % 海拔高度(km)
    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 * exp(-c / sind(solar_elevation_deg)) );

    % 计算 光学效率 和 输出热功率
    energy = 0;
    unit_energy = 0;
    eta = zeros(data_len, 1);
    for i = 1:data_len
        eta(i) = eta_sb(i) * eta_trunc(i) * eta_cos(i) * eta_at(i) *
eta_ref;
        energy = energy + DNI * heights(i) * widths(i) * eta(i);
    end
    unit_energy = energy / sum(heights .* widths);
```

```
end
```

程序三:相交函数

```
function [isIntersected] = check_intersect_cylinder(rayPoint, rayDir,
collector_tower, h, r)

    % 检查从 rayPoint 发出的，朝着 rayDir 方向的射线，与圆柱体是否相交
    % rayPoint: 射线的起点
    % rayDir: 射线的方向向量
    % collector_tower: 圆柱体中心坐标
    % h 圆柱体高度
    % r 圆柱体半径
    % isIntersect: 是否相交
    % intersectionPoint: 如果相交，返回交点坐标，否则返回 []

    isIntersected = false;
    point = [NaN, NaN, NaN];

    % 计算射线与圆柱体顶部和底部的交点
    t1 = (collector_tower(3) - rayPoint(3)) / rayDir(3);
    t2 = (collector_tower(3) + h - rayPoint(3)) / rayDir(3);

    % 计算交点坐标
    p1 = rayPoint + t1 .* rayDir;
    p2 = rayPoint + t2 .* rayDir;

    % 检查交点是否在圆柱体的顶部或底部
    % 这里对底部的设定是为了代码复用，适用于计算阴影遮挡和截断效率
    % 阴影遮挡部分，入射光的反方向都是向上的，不会与吸收塔底部产生交点
    % 截断效率部分，反射光的方向向上，如果计算出反射光与集热器"底部"有交点，则这
条光线被吸收塔阻挡，导致最终与集热器没有交点
    if t1 > 0 && (norm(p1(1:2) - collector_tower(1:2)) <= r)
        isIntersected = false;
        point = [];
        return;
    elseif t2 > 0 && (norm(p2(1:2) - collector_tower(1:2)) <= r)
        isIntersected = true;
        point = p2;
```

```
        return;
    end

    % 计算射线与圆柱体侧面的交点
    d = rayPoint(1:2) - collector_tower(1:2);
    A = rayDir(1)^2 + rayDir(2)^2;
    B = 2 * (rayDir(1)*d(1) + rayDir(2)*d(2));
    C = d(1)^2 + d(2)^2 - r^2;

    discriminant = B^2 - 4*A*C;

    if discriminant >= 0
        t3 = (-B + sqrt(discriminant)) / (2*A);
        t4 = (-B - sqrt(discriminant)) / (2*A);

        p3 = rayPoint + t3 .* rayDir;
        p4 = rayPoint + t4 .* rayDir;

        % 检查交点是否在圆柱体的有效高度内
        if t3 > 0 && (collector_tower(3) <= p3(3) && p3(3) <=
collector_tower(3) + h)
            isIntersected = true;
            point = p3;
        elseif t4 > 0 && (collector_tower(3) <= p4(3) && p4(3) <=
collector_tower(3) + h)
            isIntersected = true;
            point = p4;
        end
    end
end
```

程序四:圆内生成点

concentric_circles.m

```
function [points, nums, circle_r, point_num] = concentric_circles(center,
circle_num, min_r, dist)
    % 输入参数:
    % center: 圆心坐标 [x, y]
    % n: 同心圆的数量
    % r: 最小的圆半径
    % d: 两点之间的最小距离
    % 输出参数:
```

```
    % points: 所有生成的点的坐标

    points = [];
    point_num = zeros(1, circle_num);
    circle_r = min_r:dist:min_r+(circle_num-1)*dist;
    prev_num_points = 0; % 记录前一个圆上的点的数量

    for i = 1:circle_num
        R = min_r + (i-1)*dist;  % 当前圆的半径
        circumference = 2 * pi * R;  % 当前圆的周长
        num_points = floor(circumference / dist);  % 当前圆上可以放置的点的
数量
        delta_theta = 2*pi / num_points;  % 每个点之间的角度间隔

        % 基于前一个圆的点数计算偏移
        if i > 1
            offset = pi / (prev_num_points + num_points);
        else
            offset = 0;
        end

        for j = 1:num_points
            theta = (j-1) * delta_theta + offset;  % 加上偏移量
            x = center(1) + R * cos(theta);
            y = center(2) + R * sin(theta);
            points = [points; x y];
        end

        prev_num_points = num_points; % 更新前一个圆的点数
    end

    % 删除距离原点超过 350 的点
    distances = sqrt(sum((points - [0, 0]).^2, 2));
    points(distances > 350, :) = [];
    nums = size(points, 1);

%     scatter(points(:, 1), points(:, 2), 'k.');
%     hold on;
%     banjing = 350; % 半径
%     a = 0; % 圆心横坐标
%     b = 0; % 圆心纵坐标
%     theta = 0:pi/20:2*pi; %角度[0,2*pi]
%     x = a+banjing*cos(theta);
%     y = b+banjing*sin(theta);
```

```
%     plot(x,y,'-')
%     axis equal
%     fprintf('生成了%d 个点\n', size(points, 1));

End
```

程序五:效率计算

eta_caculate.m

```
function [eta_sb, eta_trunc, eta_cos, eta_at] = eta_caculate(positions,
num, sun_direction, normal_direction, collector_tower, vertex_data)

    % 计算单个定日镜的阴影遮挡效率
    % sun_direction 是太阳主光线的方向
    % normal_direction 是目标定日镜的法向向量
    % position 是定日镜位置数据
    % num 是目标定日镜编号
    % collector_tower 是集热塔坐标

    % 取点个数
    num_points = 100;
    % 距离
    dist_nearby = 20;
    % 轮数
    epoch = 1;

    % 目标定日镜数据
    target_heliostat = positions(num, 1:3);

    % 计算目标点与所有点的距离
    distances = sqrt(sum((target_heliostat - positions).^2, 2));

    % 找出距离不超过 n 的点的索引
    nearby_indices = distances <= dist_nearby & distances > 0;

    % 根据索引提取符合条件的坐标点
    nearby_heliostat_vertex = vertex_data(:, :, nearby_indices);
%     disp(size(nearby_heliostat_vertex, 3))

    points = generate_point(vertex_data(:, :, num), num_points);
```

```
    % 没被挡住的光线数目
    counter_not_blocked = 0;
    % 到达集热器的光线数目
    counter_reach_collector = 0;
    % 对每一轮
    for e = 1:epoch
        % 对每个点
        for i = 1:size(points, 1)
            % 标识是否被挡住，初始设置为 false
            blocked = false;
            % 太阳光方向（光锥模型，进行一定偏移）
            new_direction = offset_direction(sun_direction, 4.65e-3);
            % 反射方向
            reflection_direction = reflect(-sun_direction,
normal_direction);
%             disp(sun_direction)
%             disp(reflection_direction)
            % 检查入射光是否被集热塔阻挡
            blocked = check_intersect_cylinder(points(i, :),
new_direction, [collector_tower(1), collector_tower(2), 0], 84, 3.5);
%             if blocked
%                 disp(num)
%             end
            for j = 1:size(nearby_heliostat_vertex, 3)
                if blocked
                    break
                end
                % 检查入射是否被遮挡
                blocked = blocked | check_intersection(points(i, :),
new_direction, nearby_heliostat_vertex(:, :, j));
                % 检查反射是否被遮挡
                blocked = blocked | check_intersection(points(i, :),
reflection_direction, nearby_heliostat_vertex(:, :, j));
            end
            if ~blocked
                counter_not_blocked = counter_not_blocked + 1;
                if check_intersect_cylinder(points(i, :),
reflection_direction, [collector_tower(1), collector_tower(2), 0], 84,
3.5)
                    counter_reach_collector = counter_reach_collector + 1;
                end
            end
        end
    end
```

```
    % 阴影遮挡效率
    eta_sb = counter_not_blocked/ (epoch * size(points, 1));
    % 集热器截断效率
    if counter_not_blocked == 0
        eta_trunc = 0;
    else
        eta_trunc = counter_reach_collector / counter_not_blocked;
    end

    % 余弦效率
    eta_cos = cos(0.5 * acos(dot(real(sun_direction),
real(reflection_direction))));
%     eta_cos = sqrt((dot(real(sun_direction),
real(reflection_direction)) + 1) / 2);
    assert(isreal(eta_cos));

    % 镜面中心到集热器中心的距离
    d_HR = norm(collector_tower - target_heliostat);
    % 大气透射率
    eta_at = 0.99321 - 0.0001176 * d_HR + 1.97*10^-8 * d_HR^2;

end
```

程序六:定日镜生成点

generate_point.m

```
function points_inside_rectangle = generate_point(vertices, num_points)

    % 在一个定日镜上生成均匀的点阵
    % 输入四个顶点坐标矩阵和要生成的点的数量
    % 输出在长方形内均匀取点的坐标

    % 从输入矩阵中提取四个顶点
    v1 = vertices(1, :);
    v2 = vertices(2, :);
    v3 = vertices(3, :);
    v4 = vertices(4, :);

    % 计算长方形的边向量和面积
```

```
    edge1 = v2 - v1;
    edge2 = v4 - v1;
    area = norm(cross(edge1, edge2));

    % 计算每个点的面积增量
    increment = sqrt(area / num_points);

    % 计算长方形的边长
    length1 = norm(edge1);
    length2 = norm(edge2);

    % 计算在每个边上的点数
    num_points_edge1 = round(length1 / increment);
    num_points_edge2 = round(length2 / increment);

    % 初始化存储点坐标的数组
    points_inside_rectangle = zeros(num_points_edge1 * num_points_edge2,
3);

    % 生成均匀分布的点坐标
    k = 1;
    for i = 1:num_points_edge1
        for j = 1:num_points_edge2
            u = (i - 0.5) * increment / length1;
            v = (j - 0.5) * increment / length2;
            point = v1 + u * edge1 + v * edge2;
            points_inside_rectangle(k, :) = point;
            k = k + 1;
        end
    end

end
```

程序七:法线计算

Normal_direction.m

```
function [normal_direction] = normal_direction(target_heliostat,
sun_direction, collector_tower)

    % 计算定日镜的法向方向向量
    % target_heliostat 是目标定日镜数据
    % sun_direction 是太阳主光线的方向
```

```
    % collector_tower 是集热塔坐标

    % 反射光线方向
    reflect_direction = collector_tower - target_heliostat;
    assert (reflect_direction(3) * sun_direction(3) >= 0)
    % 归一化
    sun_direction = sun_direction ./ norm(sun_direction);
    reflect_direction = reflect_direction ./ norm(reflect_direction);
    % 计算定日镜法线方向
    normal_direction = sun_direction + reflect_direction;
    normal_direction = normal_direction ./ norm(normal_direction);

end
```

8.2 问题二用到的程序:

程序一:问题二主程序

```
clear;
clc;

% 第二问主程序的第一部分
% 遍历吸收塔的可能位置，给双层规划赋予初始状态
% 吸收塔可能出现的坐标
x_ = linspace(-350, 350, 10);
y_ = linspace(-350, 350, 10);
p_tower = zeros(length(x_), 2);
for i = 1:length(x_)
    for j = 1:length(y_)
        p_tower(i * length(x_) + j, :) = [x_(i) y_(j)];
    end
end
% 删除距离原点超过 350 的点
distances = sqrt(sum((p_tower - [0, 0]).^2, 2));
p_tower(distances > 350, :) = [];
p_tower = unique(p_tower, 'rows');
% scatter(p_tower(:, 1), p_tower(:, 2))
% axis equal
```

```
% p_tower = [116.6667 -194.4444]; % 结果

fprintf('遍历 %d 个点，初始化吸收塔坐标\n', size(p_tower, 1));



% 集热塔位置信息（需要优化 x 和 y）
collector_tower = [0, 0, 80];
% 定日镜安装高度（需要优化）
install_height = 5;
% 定日镜高度（需要优化）
height = 6;
% 定日镜宽度（需要优化）
width = 6;
% 最小同心圆的半径
min_r = 100;
% 相邻两点最小距离
dist = width + 5;



% 遍历的记录
record = zeros(size(p_tower, 1), 3);

for e = 1:size(p_tower, 1)

    init_circle_num = 75; % 初始圆圈数量(设置大一点，以铺满整个场，超出部分会被去除)
    [positions, data_len, circle_r, point_num] = concentric_circles([p_tower(e, :), 80], init_circle_num, min_r, dist);
    assert(isvalid(install_height, height, width, circle_r, point_num));
    % 定日镜法向量（每次都要重新计算）
    norm_directions = zeros(data_len, 3);
    % 定日镜四个角的坐标（每次都要重新计算）
    vertex_data = zeros(4, 3, data_len);
    [record(e, 1), record(e, 2), record(e, 3)] = val_func(repmat(install_height, data_len, 1), repmat(height, data_len, 1), ...
        repmat(width, data_len, 1), init_circle_num, min_r, dist, [p_tower(e, :), 80]);
    disp(e)
    disp(record(e, :))

end
```

程序二:问题二主程序

a_q2_2.m

```
clear;
clc;
% 第二问主程序的第二部分（根据第一部分的结果进行双层规划
% 初始化结果
% 集热塔位置信息（需要优化 x 和 y）
collector_tower = [95.4545 -159.0909, 80]; % 采用初始化的结果
% 定日镜安装高度（需要优化）
install_height = 5;
% 定日镜高度（需要优化）
height = 6;
% 定日镜宽度（需要优化）
width = 6;

% % 检查点结果
% % 集热塔位置信息（需要优化 x 和 y）
% collector_tower = [78.836753, -148.319177   80.0000]; % 采用初始化的结果
% % 定日镜安装高度（需要优化）
% install_height = 2.8750;
% % 定日镜高度（需要优化）
% height = 5.75;
% % 定日镜宽度（需要优化）
% width = 5.75;

% 初始圆圈数量(设置大一点，以铺满整个场，超出部分会被去除)
init_circle_num = 75;
% 最小同心圆的半径
min_r = 100;
% 相邻两点最小距离
dist = width + 5;

[~, data_len, circle_r, point_num] = concentric_circles(collector_tower,
init_circle_num, min_r, dist);
assert(isvalid(install_height, height, width, circle_r, point_num));
% [~, data_len] = concentricCircles(collector_tower, circle_r, point_num,
circle_angle);
```

```
[init_energy, init_unit_energy, init_eta] =
val_func(repmat(install_height, data_len, 1), repmat(height, data_len,
1), ...
    repmat(width, data_len, 1), init_circle_num, min_r, dist,
collector_tower);

fprintf("功率：%4f, 单位功率：%4f, 光学效率：%4f\n", init_energy,
init_unit_energy, init_eta);

% 循环直到收敛
flag = true;
epoch = 0;
while flag
    epoch = epoch + 1;
    if epoch ~= 1
        % 上层规划，目标是平均光学效率最高
        % 吸收塔移动步长
        step = 2.5 + 15 * rand();
        % 步长数组
        step_sizes = [-step, 0, step];

        % 初始化存储周围点的数组
        neighbor_points = [];
        % 生成周围的点
        for dx = step_sizes
            for dy = step_sizes
                % 计算新点的坐标
                new_x = collector_tower(1) + dx;
                new_y = collector_tower(2) + dy;
                % 将新点添加到数组中
                neighbor_points = [neighbor_points; [new_x, new_y]];
            end
        end

        % 遍历邻居点，选择光学效率最高的点作为吸收塔的坐标
        unit_energy_up_layer = zeros(1, length(step_sizes)^2);
        energy_up_layer = zeros(1, length(step_sizes)^2);
        eta_up_layer = zeros(1, length(step_sizes)^2);
        for i = 1:length(step_sizes)^2
            % 计算定日镜坐标
            [~, data_len] = concentric_circles([neighbor_points(i, :),
80], init_circle_num, min_r, dist);
            % 计算光学效率
```

```
            [energy_up_layer(i), unit_energy_up_layer(i), eta_up_layer(i)]
= val_func(repmat(install_height, data_len, 1), repmat(height, data_len,
1), ...
                repmat(width, data_len, 1), init_circle_num, min_r, dist,
[neighbor_points(i, :), 80]);
            fprintf('邻居%d 的光学效率：%4f\n', i, eta_up_layer(i));
        end
        % 选择最大光学效率的点
        [~, index] = max(eta_up_layer);
        collector_tower = [neighbor_points(index, :), 80];
        fprintf('collector_tower 更新为：%4f, %4f\n', collector_tower(1),
collector_tower(2));
    end


    init_install_height = install_height;
    init_height = height;
    init_width = width;
    % 下层规划，目标是单位面积输出功率最大，要满足各种约束条件
    step = 0.25;
    for inner_loop = 1:2
        % 调整安装高度
        best_install_h = install_height;
        best_unit_energy = -inf;
        for install_h = max([2, height/2]):step:6
            % 计算定日镜坐标
            [~, data_len] = concentric_circles(collector_tower,
init_circle_num, min_r, dist);
            [res_energy, res_unit_energy, ~] = val_func(repmat(install_h,
data_len, 1), repmat(height, data_len, 1), ...
                repmat(width, data_len, 1), init_circle_num, min_r, dist,
collector_tower);
            if res_energy > 60 && res_unit_energy > best_unit_energy
                best_unit_energy = res_unit_energy;
                best_install_h = install_h;
            end
            fprintf('暂定安装高度为：%4f，总功率：%4f，单位功率：%4f\n',
install_h, res_energy, res_unit_energy);
        end
        % 取单位效率最高的参数
        install_height = best_install_h;
        fprintf('install_height 更新为：%4f\n', install_height);
```

```
        % 调整高度
        best_h = height;
        best_unit_energy = -inf;
        for h = max(2, 2/3*width):step:min([2*install_height, 8, width])
            % 计算定日镜坐标
            [~, data_len] = concentric_circles(collector_tower,
init_circle_num, min_r, dist);
            [res_energy, res_unit_energy, ~] =
val_func(repmat(install_height, data_len, 1), repmat(h, data_len, 1), ...
                repmat(width, data_len, 1), init_circle_num, min_r, dist,
collector_tower);
            if res_energy > 60 && res_unit_energy > best_unit_energy
                best_unit_energy = res_unit_energy;
                best_h = h;
            end
            fprintf('暂定高度为：%4f，总功率：%4f，单位功率：%4f\n', h,
res_energy, res_unit_energy);
        end
        % 取单位效率最高的参数
        height = best_h;
        fprintf('height 更新为：%4f\n', height);


        % 调整宽度
        best_w = width;
        best_unit_energy = -inf;
        for w = max([height, 2]):step:8
            % 更新相邻两定日镜最小距离
            dist = w + 5;
            % 计算定日镜坐标
            [~, data_len] = concentric_circles(collector_tower,
init_circle_num, min_r, dist);
            [res_energy, res_unit_energy, ~] =
val_func(repmat(install_height, data_len, 1), repmat(height, data_len,
1), ...
                repmat(w, data_len, 1), init_circle_num, min_r, dist,
collector_tower);
            if res_energy > 60 && res_unit_energy > best_unit_energy
                best_unit_energy = res_unit_energy;
                best_w = w;
            end
            fprintf('暂定宽度为：%4f，总功率：%4f，单位功率：%4f\n', w,
res_energy, res_unit_energy);
        end
```

```
        % 取单位效率最高的参数
        width = best_w;
        fprintf('width 更新为：%4f\n', width);
        % 更新相邻两定日镜最小距离
        dist = width + 5;

        % 调整高度
        best_h = height;
        best_unit_energy = -inf;
        for h = 2:step:min([2*install_height, 8, width])
            % 计算定日镜坐标
            [~, data_len] = concentric_circles(collector_tower,
init_circle_num, min_r, dist);
            [res_energy, res_unit_energy, ~] =
val_func(repmat(install_height, data_len, 1), repmat(h, data_len, 1), ...
                repmat(width, data_len, 1), init_circle_num, min_r, dist,
collector_tower);
            if res_energy > 60 && res_unit_energy > best_unit_energy
                best_unit_energy = res_unit_energy;
                best_h = h;
            end
            fprintf('暂定高度为：%4f，总功率：%4f，单位功率：%4f\n', h,
res_energy, res_unit_energy);
        end
        % 取单位效率最高的参数
        height = best_h;
        fprintf('height 更新为：%4f\n', height);
    end

    % 下层规划没有变动，则收敛，退出循环
    if init_install_height == install_height && init_height == height &&
init_width == width
        flag = false;
    end


    [positions, data_len] = concentric_circles(collector_tower,
init_circle_num, min_r, dist);
    [res_energy, res_unit_energy, res_eta] =
val_func(repmat(install_height, data_len, 1), repmat(height, data_len,
1), ...
        repmat(width, data_len, 1), init_circle_num, min_r, dist,
collector_tower);
```

```
    fprintf("功率：%4f，单位功率：%4f，光学效率：%4f\n", init_energy,
init_unit_energy, init_eta);

end
```

程序三:判断是否合法

```
function [valid, flag] = isvalid(install_height, height, width, circle_r,
point_num)
    % 检查状态是否合法
    if width < height || height > 2 * install_height
        valid = false;
        flag = 1;
        return
    end
    if width < 2 || width > 8 || height < 2 || height > 8 ||
install_height < 2 || install_height > 6
        valid = false;
        flag = 2;
        return
    end
    if ~all(circle_r >= 100)
        valid = false;
        flag = 3;
        return
    end
    for i = 2:length(circle_r)
        if circle_r(i) - circle_r(i-1) < width + 5
            % 如果相邻元素的差值不大于 width，将 flag 设置为假并退出循环
            valid = false;
            flag = 4;
            return;
        end
    end
    for i = 1:length(circle_r)
        if 2 * sin(pi / point_num(i)) * circle_r(i) < width + 5
            valid = false;
            flag = 100 + i;
            return;
        end
    end
    valid = true;
```

```
    flag = 0;
end
```

程序四:偏移角度

```
function new_direction = offset_direction(original_direction,
max_offset_angle)

    % original_direction : 输入射线的方向向量
    % max_offset_angle : 最大偏移角度
    % new_direction : 新射线的方向

    % 计算 sigma，3*sigma 对应半角展宽
    sigma = 1/3 * atan(max_offset_angle);

    % 确保 original_direction 是单位向量
    original_direction = original_direction / norm(original_direction);

    % 找到与 original_direction 不平行的任意向量
    if original_direction(1) ~= 0 || original_direction(2) ~= 0
        temp_vector = [0; 0; 1];
    else
        temp_vector = [1; 0; 0];
    end

    % 通过外积找到第一个基向量
    basis1 = cross(original_direction, temp_vector);
    basis1 = basis1 / norm(basis1);

    % 再次使用外积找到第二个基向量
    basis2 = cross(original_direction, basis1);
    basis2 = basis2 / norm(basis2);

    % 在两个基向量上生成二维正态分布的随机点
    rand_point = sigma * randn(2, 1);

    % 使用基向量将这些随机点转换为平面上的三维点
    point = (basis1 * rand_point(1, :) + basis2 * rand_point(2, :))';
    new_direction = original_direction + point';
    new_direction = new_direction ./ norm(new_direction);
end
```

程序五:光线反射

```
function reflection = reflect(input, norm_mirror)

    % 根据入射光线和镜面法向向量，计算出射光线
    % input: 入射光线的方向向量 (单位向量)
    % norm_mirror: 镜面的法向量 (单位向量)

    % 输出:
    % R: 出射光线的方向向量

    reflection = input - 2 * dot(input, norm_mirror) * norm_mirror;
    % 将 R 归一化
    reflection = reflection / norm(reflection);

end
```

程序六:太阳角度

```
function [solar_elevation_deg, sun_azimuth_deg, direction] =
sun_angle(lon_deg, lat_deg, date_time)

    % 计算太阳高度角的函数
    % 输入观测地的地理经纬度，地方时（日期时间）
    % 输出太阳高度角的 sin 值，太阳高度角

    % 观测地的地理经度
    % lon_deg = 98.5;
    lon_rad = deg2rad(lon_deg); % 将度数转换为弧度

    % 观测地的地理纬度
    % lat_deg = 39.4;
    lat_rad = deg2rad(lat_deg); % 将度数转换为弧度

    % 观测地的日期
    % date_time = datetime('2023-9-21 12:00');

    % 计算距离春分的天数（以春分作为第 0 天起算）
    spring_equinox = datetime(year(date_time), 3, 21, 0, 0, 0); % 假设春分
日期为 3 月 21 日
    D = days(date_time - spring_equinox);

    % 太阳赤纬
    sin_delta = sin(2*pi*D / 365) * sin(2*pi*23.45 / 360);
    delta_rad = asin(sin_delta);
```

```
    % 地方时（以时为单位）
    local_time_hours = hour(date_time) + minute(date_time) / 60;

    % 计算太阳高度角（以弧度为单位）
    w = (local_time_hours - 12) * (pi / 12); % 将地方时转换为弧度
    sin_as = sin(lat_rad) * sin(delta_rad) + cos(lat_rad) *
cos(delta_rad) * cos(w);
    solar_elevation_rad = asin(sin_as); % 太阳高度角的弧度值

    % 太阳方位角
    cos_sun_azimuth = (sin_delta - sin_as * sin(lat_rad)) /
(cos(solar_elevation_rad) * cos(lat_rad));
    sun_azimuth_rad = acos(cos_sun_azimuth);

    % 计算太阳光的方向向量
    x = cos(solar_elevation_rad) * sin(sun_azimuth_rad);
    y = cos(solar_elevation_rad) * cos(sun_azimuth_rad);
    z = sin(solar_elevation_rad);
    direction = [x, y, z];


    % 将太阳高度角转换为度数
    solar_elevation_deg = rad2deg(solar_elevation_rad);
    sun_azimuth_deg = rad2deg(sun_azimuth_rad);

    % 展示结果
%     fprintf('太阳高度角（度）：%f\n', solar_elevation_deg);
%     fprintf('太阳方位角（度）：%f\n', sun_azimuth_deg);
%     fprintf('太阳光的方向向量为：[%.4f, %.4f, %.4f]\n', x, y, z);

end
```

程序七:评价函数

val_func.m

```
function [sum_energy, unit_energy, avg_eta] = val_func(install_heights,
heights, widths, circle_num, min_r, dist, collector_tower)

    % 评价函数
    positions = concentric_circles(collector_tower, circle_num, min_r,
dist);
```

```
    positions = [positions, install_heights];

    dates = {'2023-1-21', '2023-2-21', '2023-3-21', '2023-4-21', '2023-5-
21', '2023-6-21', '2023-7-21', '2023-8-21','2023-9-21', '2023-10-
21','2023-11-21', '2023-12-21'};
    times = {'9:00:00', '10:30:00', '12:00:00', '13:30:00', '15:00:00'};
    datetimeMatrix = repmat(datetime('now'), 12, 5);
    for i = 1:length(dates)
        for j = 1:length(times)
            % 合并日期和时间，然后转换为 datetime 对象
            datetimeString = [dates{i} ' ' times{j}];
            datetimeMatrix(i, j) = datetime(datetimeString, 'InputFormat',
'yyyy-MM-dd HH:mm:ss');
        end
    end

    energy = cell(length(dates), length(times));
    eta = cell(length(dates), length(times));

    % 求年均热功率
    for j = 1:length(dates)
        for i = 1:length(times)
            [~, energy{j, i}, eta{j, i}] = ...
                caculate(datetimeMatrix(j, i), positions, heights, widths,
collector_tower);
        end
    end

    sum_energy = 0;
    avg_eta = zeros(length(dates), 1);
    for j = 1:length(dates)
        for i = 1:length(times)
            avg_eta(j) = avg_eta(j) + mean(eta{j, i});
            sum_energy = sum_energy + energy{j, i};
        end
        avg_eta(j) = avg_eta(j) / length(times);
    end
    sum_energy = sum_energy / length(dates) / length(times) / 1000;
    avg_eta = mean(avg_eta);
    sum_area = 0;
    for i = 1:size(positions, 1)
        sum_area = sum_area + widths(i) * heights(i);
    end
    unit_energy = sum_energy / sum_area;
```

```
end
```

程序八:计算顶点坐标

Vertex.m

```
function [vertex1, vertex2, vertex3, vertex4] = vertex(target_heliostat,
height, width, normal_direction)

    % 计算定日镜四个角的坐标
    % target_heliostat 是目标定日镜数据
    % normal_direction 是定日镜法向方向

    % 计算长方形的四个顶点坐标
    % 首先，计算两个与地面平行的单位向量，用于确定矩形的边方向
    up_vector = [0, 0, 1]; % 地面的法向向量
    side_vector1 = cross(normal_direction, up_vector);
    side_vector1 = side_vector1 / norm(side_vector1);
    side_vector2 = cross(normal_direction, side_vector1);
    side_vector2 = side_vector2 / norm(side_vector2);

    % 计算四个顶点坐标
    vertex1 = target_heliostat - (width / 2) * side_vector1 - (height /
2) * side_vector2;
    vertex2 = target_heliostat + (width / 2) * side_vector1 - (height /
2) * side_vector2;
    vertex3 = target_heliostat + (width / 2) * side_vector1 + (height /
2) * side_vector2;
    vertex4 = target_heliostat - (width / 2) * side_vector1 + (height /
2) * side_vector2;

    % disp('长方形的四个顶点坐标：');
    % disp(['顶点 1: ', num2str(vertex1)]);
    % disp(['顶点 2: ', num2str(vertex2)]);
    % disp(['顶点 3: ', num2str(vertex3)]);
    % disp(['顶点 4: ', num2str(vertex4)]);


end
```

8.3 问题三用到的程序:

程序一:主程序

a_q3.m

```
clear;
clc;

% 第三问主程序，双层规划优化
% 集热塔位置信息（需要优化 x 和 y）
collector_tower = [48.2369, -126.0147, 80]; % 采用第二问的结果
% 最小同心圆的半径
min_r = 100;
% 安装高度函数的斜率
k_install_h = 0;
% 高度函数的斜率
k_h = 0;
% 宽度函数的斜率
k_w = 0;
% 基准点
a1 = 2.875;
a0 = 5.75;
init_circle_num = 75;
% 第二问的
dist = 5.75 + 5;

[~, ~, circle_r, ~] = concentric_circles(collector_tower,
init_circle_num, min_r, dist);
[valid, install_heights, heights, widths, dist, circle_r] =
update_(k_install_h, k_h, k_w, circle_r, collector_tower, a0, a1);
assert(valid)
disp('初始解合法')


[init_energy, init_unit_energy, init_eta] = val_func_2(install_heights,
heights, widths, init_circle_num, min_r, dist, collector_tower);
fprintf("功率：%4f, 单位功率：%4f, 光学效率：%4f\n", init_energy,
init_unit_energy, init_eta);


% 循环直到收敛
```

```
flag = true;
epoch = 0;
while flag
    epoch = epoch + 1;
    if epoch ~= 1
        % 上层规划，目标是平均光学效率最高
        % 吸收塔移动步长
        step = 2.5 + 15 * rand();
        % 步长数组
        step_sizes = [-step, 0, step];

        % 初始化存储周围点的数组
        neighbor_points = [];
        % 生成周围的点
        for dx = step_sizes
            for dy = step_sizes
                % 计算新点的坐标
                new_x = collector_tower(1) + dx;
                new_y = collector_tower(2) + dy;
                % 将新点添加到数组中
                neighbor_points = [neighbor_points; [new_x, new_y]];
            end
        end

        % 遍历邻居点，选择光学效率最高的点作为吸收塔的坐标
        unit_energy_up_layer = zeros(1, length(step_sizes)^2);
        energy_up_layer = zeros(1, length(step_sizes)^2);
        eta_up_layer = zeros(1, length(step_sizes)^2);
        for i = 1:length(step_sizes)^2
            % 计算定日镜坐标
            [~, install_heights, heights, widths, dist] =
update_(new_k_install_h, k_h, k_w, circle_r, [neighbor_points(i, :), 80],
a0, a1);
            [~, data_len] = concentric_circles_2([neighbor_points(i, :),
80], init_circle_num, min_r, dist);

            % 计算光学效率
            [energy_up_layer(i), unit_energy_up_layer(i), eta_up_layer(i)]
= val_func_2(install_heights, heights, ...
                widths, init_circle_num, min_r, dist,
[neighbor_points(i, :), 80]);
            fprintf('邻居%d 的光学效率：%4f\n', i, eta_up_layer(i));
        end
        % 选择最大光学效率的点
```

```
        [~, index] = max(eta_up_layer);
        collector_tower = [neighbor_points(index, :), 80];
        fprintf('collector_tower 更新为：%4f, %4f\n', collector_tower(1),
collector_tower(2));
    end


    init_k_install_h = k_install_h;
    init_k_h = k_h;
    init_k_w = k_w;
    % 下层规划，目标是单位面积输出功率最大，要满足各种约束条件
    step = 0.00025;
    range = 0.001;
    for inner_loop = 1:2
        % 调整安装高度的函数斜率
        best_k_install_h = k_install_h;
        best_unit_energy = -inf;
        best_circle_r = circle_r;
        for new_k_install_h = k_install_h-range:step:k_install_h+range
            [valid, install_heights, heights, widths, dist, tmp_circle_r]
= update_(new_k_install_h, k_h, k_w, circle_r, collector_tower, a0, a1);
            if ~valid
%                 disp('斜率不合法，进行下次计算');
                continue;
            end
            [res_energy, res_unit_energy, ~] = val_func_2(install_heights,
heights, widths, init_circle_num, min_r, dist, collector_tower);
            if res_energy > 60 && res_unit_energy > best_unit_energy
                best_unit_energy = res_unit_energy;
                best_k_install_h = new_k_install_h;
                best_circle_r = tmp_circle_r;
            end
            fprintf('暂定安装高度斜率为：%4f，总功率：%4f，单位功率：%4f\n',
new_k_install_h, res_energy, res_unit_energy);
        end
        % 取单位效率最高的参数
        k_install_h = best_k_install_h;
        circle_r = best_circle_r;
        fprintf('k_install_h 更新为：%4f\n', k_install_h);

        % 调整高度的函数斜率
        best_k_h = k_h;
        best_unit_energy = -inf;
        best_circle_r = circle_r;
```

```
        for new_k_h = k_h-range:step:k_h+range
            [valid, install_heights, heights, widths, dist, tmp_circle_r]
= update_(k_install_h, new_k_h, k_w, circle_r, collector_tower, a0, a1);
            if ~valid
%                 disp('斜率不合法，进行下次计算');
                continue;
            end
            [res_energy, res_unit_energy, ~] = val_func_2(install_heights,
heights, widths, init_circle_num, min_r, dist, collector_tower);
            if res_energy > 60 && res_unit_energy > best_unit_energy
                best_unit_energy = res_unit_energy;
                best_k_h = new_k_h;
                best_circle_r = tmp_circle_r;
            end
            fprintf('暂定高度斜率为：%4f，总功率：%4f，单位功率：%4f\n',
new_k_h, res_energy, res_unit_energy);
        end
        % 取单位效率最高的参数
        k_h = best_k_h;
        circle_r = best_circle_r;
        fprintf('k_h 更新为：%4f\n', k_h);

        % 调整宽度的函数斜率
        best_k_w = k_w;
        best_unit_energy = -inf;
        best_circle_r = circle_r;
        for new_k_w = k_w-range:step:k_w+range
            [valid, install_heights, heights, widths, dist, tmp_circle_r]
= update_(k_install_h, k_h, new_k_w, circle_r, collector_tower, a0, a1);
            if ~valid
%                 disp('斜率不合法，进行下次计算');
                continue;
            end
            [res_energy, res_unit_energy, ~] = val_func_2(install_heights,
heights, widths, init_circle_num, min_r, dist, collector_tower);
            if res_energy > 60 && res_unit_energy > best_unit_energy
                best_unit_energy = res_unit_energy;
                best_k_h = new_k_h;
                best_circle_r = tmp_circle_r;
            end
            fprintf('暂定宽度斜率为：%4f，总功率：%4f，单位功率：%4f\n',
new_k_h, res_energy, res_unit_energy);
        end
        % 取单位效率最高的参数
```

```
        k_w = best_k_w;
        circle_r = best_circle_r;
        fprintf('k_w 更新为：%4f\n', k_w);

        % 调整高度的函数斜率
        best_k_h = k_h;
        best_unit_energy = -inf;
        best_circle_r = circle_r;
        for new_k_h = k_h-range:step:k_h+range
            [valid, install_heights, heights, widths, dist, tmp_circle_r]
= update_(k_install_h, new_k_h, k_w, circle_r, collector_tower, a0, a1);
            if ~valid
%                 disp('斜率不合法，进行下次计算');
                continue;
            end
            [res_energy, res_unit_energy, ~] = val_func_2(install_heights,
heights, widths, init_circle_num, min_r, dist, collector_tower);
            if res_energy > 60 && res_unit_energy > best_unit_energy
                best_unit_energy = res_unit_energy;
                best_k_h = new_k_h;
                best_circle_r = tmp_circle_r;
            end
            fprintf('暂定高度斜率为：%4f，总功率：%4f，单位功率：%4f\n',
new_k_h, res_energy, res_unit_energy);
        end
        % 取单位效率最高的参数
        k_h = best_k_h;
        circle_r = best_circle_r;
        fprintf('k_h 更新为：%4f\n', k_h);

    end


    % 下层规划没有变动，则收敛，退出循环
    if init_k_install_h == k_install_h && init_k_h == k_h && init_k_w ==
k_w
        flag = false;
    end

    [valid, install_heights, heights, widths, dist, circle_r] =
update_(k_install_h, k_h, new_k_w, circle_r, collector_tower, a0, a1);
    [init_energy, init_unit_energy, init_eta] =
val_func_2(install_heights, heights, widths, init_circle_num, min_r,
dist, collector_tower);
```

```
    fprintf("功率：%4f，单位功率：%4f，光学效率：%4f\n", init_energy,
init_unit_energy, init_eta);


end


function [valid, flag] = validation_test(group_install_heights,
group_heights, group_widths, circlr_r, group_num)
    for i = 1:group_num
        if group_widths(i) < group_heights(i) || group_heights(i) > 2 *
group_install_heights(i)
            valid = false;
            flag = 1;
            return
        end
        if group_widths(i) < 2 || group_widths(i) > 8 || group_heights(i)
< 2 || group_heights(i) > 8 || group_install_heights(i) < 2 ||
group_install_heights(i) > 6
            valid = false;
            flag = 2;
            return
        end
        if i ~= 1 && circlr_r(i) - circlr_r(i-1) < max([group_widths(i),
group_widths(i-1)]) + 5
            valid = false;
            flag = 3;
            return
        end
    end
    valid = true;
    flag = 0;
    return;
end

function [valid, install_heights, heights, widths, dist, circle_r] =
update_(k_install_h, k_h, k_w, circle_r, collector_tower, a0, a1)

    % 初始圆圈数量(设置大一点，以铺满整个场，超出部分会被去除)
    min_r = 100;
    init_circle_num = 75;
    group_install_height = zeros(init_circle_num, 1);
    group_height = zeros(init_circle_num, 1);
    group_width = zeros(init_circle_num, 1);
```

```
    dist = zeros(init_circle_num, 1);
    % 每组定日镜的信息
    for i = 1:init_circle_num
        group_install_height(i) = k_install_h * (circle_r(i)-min_r) + a1;
        group_height(i) = k_h * (circle_r(i)-min_r) + a0;
        group_width(i) = k_w * (circle_r(i)-min_r) + a0;
        dist(i) = group_width(i) + 5.01;
    end


    [~, data_len, circle_r, ~, group] =
concentric_circles_2(collector_tower, init_circle_num, min_r, dist);
    group_nums = max(group);
    group_install_height = group_install_height(1:group_nums);
    group_width = group_width(1:group_nums);
    group_height = group_height(1:group_nums);
    [valid, flag] = validation_test(group_install_height, group_height,
group_width, circle_r, group_nums);
%     if ~valid
%         disp(group_install_height')
%         disp(group_width')
%         disp(group_height')
%         disp(flag);
%     end
    % 每个定日镜的信息
    install_heights = zeros(data_len, 1);
    heights = zeros(data_len, 1);
    widths = zeros(data_len, 1);
    for i = 1:data_len
        install_heights(i) = group_install_height(group(i));
        heights(i) = group_height(group(i));
        widths(i) = group_width(group(i));
    end
end
```

程序二:评价函数二

```
function [sum_energy, unit_energy, avg_eta] = val_func_2(install_heights,
heights, widths, circle_num, min_r, dist, collector_tower)

    % 评价函数
```

```
    positions = concentric_circles_2(collector_tower, circle_num, min_r,
dist);
    positions = [positions, install_heights];

    dates = {'2023-1-21', '2023-2-21', '2023-3-21', '2023-4-21', '2023-5-
21', '2023-6-21', '2023-7-21', '2023-8-21','2023-9-21', '2023-10-
21','2023-11-21', '2023-12-21'};
    times = {'9:00:00', '10:30:00', '12:00:00', '13:30:00', '15:00:00'};
    datetimeMatrix = repmat(datetime('now'), 12, 5);
    for i = 1:length(dates)
        for j = 1:length(times)
            % 合并日期和时间，然后转换为 datetime 对象
            datetimeString = [dates{i} ' ' times{j}];
            datetimeMatrix(i, j) = datetime(datetimeString, 'InputFormat',
'yyyy-MM-dd HH:mm:ss');
        end
    end

    energy = cell(length(dates), length(times));
    eta = cell(length(dates), length(times));

    % 求年均热功率
    for j = 1:length(dates)
        for i = 1:length(times)
            [~, energy{j, i}, eta{j, i}] = ...
                caculate(datetimeMatrix(j, i), positions, heights, widths,
collector_tower);
        end
    end

    sum_energy = 0;
    avg_eta = zeros(length(dates), 1);
    for j = 1:length(dates)
        for i = 1:length(times)
            avg_eta(j) = avg_eta(j) + mean(eta{j, i});
            sum_energy = sum_energy + energy{j, i};
        end
        avg_eta(j) = avg_eta(j) / length(times);
    end
    sum_energy = sum_energy / length(dates) / length(times) / 1000;
    avg_eta = mean(avg_eta);
    sum_area = 0;
    for i = 1:size(positions, 1)
        sum_area = sum_area + widths(i) * heights(i);
```

```
    end
    unit_energy = sum_energy / sum_area;
end
```

程序三:生成圆上点

Concentric_circle2.m

```
function [points, nums, circle_r, point_num, group] =
concentric_circles_2(center, circle_num, min_r, dist)
    % 按最密排列的方法，以 center 为中心生成同心圆，生成圆上的点的坐标
    % 输入参数:
    % center: 圆心坐标 [x, y]
    % n: 同心圆的数量
    % r: 最小的圆半径
    % dist: 向量，代表每个同心圆相邻点的最小距离
    % 输出参数:
    % points: 所有生成的点的坐标

    points = [];
    point_num = zeros(1, circle_num);
    circle_r = zeros(1, circle_num);
    circle_r(1) = min_r;
    for i = 2:circle_num
        circle_r(i) = circle_r(i-1) + max(dist(i-1), dist(i));
    end
    prev_num_points = 0; % 记录前一个圆上的点的数量
    group = [];

    for i = 1:circle_num
        R = circle_r(i);  % 当前圆的半径
        circumference = 2 * pi * R;  % 当前圆的周长
        num_points = floor(circumference / dist(i));  % 当前圆上可以放置的点
的数量
        delta_theta = 2*pi / num_points;  % 每个点之间的角度间隔

        % 基于前一个圆的点数计算偏移
        if i > 1
            offset = pi / (prev_num_points + num_points);
        else
            offset = 0;
        end
```

```
        for j = 1:num_points
            theta = (j-1) * delta_theta + offset;  % 加上偏移量
            x = center(1) + R * cos(theta);
            y = center(2) + R * sin(theta);
            points = [points; x y];
            group = [group, i];
        end

        prev_num_points = num_points; % 更新前一个圆的点数
    end

    % 删除距离原点超过 350 的点
    distances = sqrt(sum((points - [0, 0]).^2, 2));
    points(distances > 350, :) = [];
    nums = size(points, 1);
    group = group(1:nums);
%
%     scatter(points(:, 1), points(:, 2), 'k.');
%     hold on;
%     banjing = 350; % 半径
%     a = 0; % 圆心横坐标
%     b = 0; % 圆心纵坐标
%     theta = 0:pi/20:2*pi; %角度[0,2*pi]
%     x = a+banjing*cos(theta);
%     y = b+banjing*sin(theta);
%     plot(x,y,'-')
%     axis equal
%     fprintf('生成了%d 个点\n', size(points, 1));

end
```