基于非对称学习因子粒子群的定日镜场优化设计

摘　要

塔式太阳能光热发电站是一种经典的太阳能光热发电装置，通过传感器检测太阳位置，调整定日镜的角度，将太阳光线反射至吸收塔的集热器中心，集热器升温后推动汽轮机转动发电。经过合理排布定日镜位置与配置定日镜和吸收塔的参数，能显著提高发电站的发电容量与经济效益。

问题一中，我们通过向量计算太阳光线的入射角，并带入公式得出所需的太阳高度角和方位角；利用微分的思想，将定日镜视细化为多个均匀分布的点阵，验证每个点是否被遮挡以及反射光线；通过矩阵变换，计算反射光线的向量，验证其是否投影在吸收塔集热器的范围内，通过统计符合条件的点的数量，得到遮挡效率和截断效率。最后计算各效率的乘积，即光学效率；利用附件公式计算，得到该镜面的平均输出热功率。

问题二中，我们分析了该问题的未知值，先是基于问题一的定日镜的规格参数，减少了关于求解最优解函数的维度，其次关于 60MW 的处理我们将 60MW 转移到方程的等号右侧，此时方程为一个零等式，由于计算的能量在于物理实际上并没有负值的，以及事实上并没有赋值能量的存在，也就是说转变为一个求最小值为 0 的方程，那么在问题二我们将会在方程减去 60 的基础之上无限接近于 0，我们同时还引入了粒子群算法，和模拟退火算法，基于两种算法模型，进行对方程最小值的无限接近，从接近值的大小以及维度判定哪一个算法模型将会更为适合本次解决问题二的重要算法，维度决定着函数方程的不确定性引入可动的学习因子，能够更快的求解，以及处理不确定的变量数目多的问题.

问题三中，在问题二模型的基础上可见增加了对自变量的更加不确定度，也就是说在问题二的基础之上，方程维度变得更高，方程的约束变得愈发的少，通过在模型二引入粒子群算法，和模拟退火算法。我们发现在更高不确定性的维度上，粒子群对比模拟退火在问题三的层面上融洽度更高、更适合，基于对 0 接近值的分析，粒子群算法不仅如此，还能更好的寻找关于局部最优解问题跳出的解决。将高度与问题一所用题目附录公式来看，改变学习因子，建立非对称学习因子粒子群算法建造了关于定日镜高度，坐标，数量， 等因素的函数，利用相同于问题二的方法，解出问题三所需要的解

最后本文解释了通过坐标向量的换算，解出关于第一问不同坐标之间参照角度的问题，第二问基于两种算法，对问题二进行求解，通过分析建立的方程与零的接近值大小关系，得出算法在对应方程函数选取的优劣性：模拟退火算法，与粒子群算法在对待不同变量个数具有优劣之分，当变量个数的增多，模拟退火会产生更大的误差值，搜索全局最优解将会陷于局部当中难以重新跳出，而粒子群算法则可以在变量个数增加的同时保持着自身低误差的特性，同样的迭代时间粒子群，在寻找全局最优对比着模拟退火算法，是运算更加快的体现，其次改变粒子的学习因子将会在不确定的问题之上得到不同于基本粒子群算法更好的改善。

关键词：微分思想 光线追踪 模拟退火算法 粒子群算法 非对称学习因子

一、 问题重述

1.1 问题一

假设吸收塔位于该圆形定日镜场中心，定日镜尺寸均为 6 m×6 m，安装高度均为 4 m，附件已给定所有定日镜的中心位置，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，将数据填入表中。

1.2 问题二

设计太阳能塔式电站镜场的额定年平均输出热功率为 60 MW，为了使得定日镜场在达到额定功率的条件下，单位镜面面积年平均输出热功率尽量大，请设计定日镜场的以下参数：吸收塔的位置坐标、定日镜尺寸、安装高度、定日镜数目、定日镜位置，将数据填入表中。

1.3 问题三

允许定日镜尺寸和安装高度各不相同，请重新设计定日镜场的各个参数，使得定日镜场在达到额定功率的条件下，单位镜面面积年平均输出热功率尽量大。并记录吸吸收塔位置坐标、定日镜尺寸、安装高度和总面数。

二、 问题分析

2.1 问题一的分析

首先，分析附件公式，确定需要计算的参数；我们通过计算定日镜中心与太阳连线向量和定日镜中心与集热器中心连线向量之间的夹角，以获取入射角，并同步运算太阳方位角。

考虑到样本容量过大，通过研究太阳运动轨迹与定日镜分布，利用等效近似与旋转思想，选定若干个具代表性的定日镜分析其阴影遮挡效率；同时，我们将每个发热板分成若干个点，通过分析该点是否被临近的定日镜遮挡，通过统计未被遮挡的点的数量，得出当前定日镜的阴影遮挡效率；计算定日镜所处的圆环面积和吸收塔在该圆环上的投影面积，近似运算二者的商为吸收塔的对该发电厂的阴影遮挡效率，两个平均遮挡效率的积为实际平均遮挡效率。

与阴影遮挡效率思路近似，将定日镜每个点视为每个太阳光线的反射点，通过模拟每个光线从太阳射线反射到集热器表面的全过程，验证当前光线是否经过集热器所在平面，通过统计经过集热器的光线数量，得出当前定日镜的截断效率。

通过查询相关资料，获取年平均余弦效率等参数的计算公式，通过如上对数据的计算，确定光学效率与当前镜面平均输出热功率，对二者乘积以获取年平均输出热功率。

2.2 问题二的分析

考虑到问题二基于多个不确定变量所建立的最值问题，本文在解决第二题方面，综合所有变量，如吸收塔位置、定日镜尺寸、安装高度、定日镜数目，归结为问题二的所有变量，通读题目背景，建立的定日镜场为圆形区域，以圆形区域的半径为安装定日镜，鉴于每个定日镜的长宽尺寸以及对应每个时间点的角度来分析，在圆形区域的半径上放置多少个定日镜则实际取决于定日镜板的长与宽，即

定日镜的数量与定日镜的尺寸存在相关性。

利用粒子群算法和模拟退火算法各自寻找最优解，按照最靠内的定日镜安装环以此也并再次确立了吸收塔的位置坐标，而坐标确立则实际吸收塔至每个定日镜的距离和问题一所求的各效率亦可知。综合下来问题二的中定日镜的位置坐标即为其余变量的子自变量，也为整个问题二的绝对自变量。再以此基础上建立最大热功率与定日镜位置自变量的最值问题。

考虑到问题二基于多个不确定变量所建立的最值问题，本文在解决第二题方面，基于第一问的高度指标，对四个未知的元素，进行上下界规划，从题目镜面边长 2~8 之间，高度 2~6 之间，其次数量的选取尤为重要，考虑到每一面定日镜时具体会绕地面而旋转以及调节角度，所以可以近似将定日镜看成为一个在球域内的运动，结合最小的以及最大的差距，可以得出一般定日镜越大其分布及数量会有所降低，同理可以得知，实际则为取决于规范的五个变量以及额定功率的计算，通过控制其范围以及正负相关性，可以对这一道题进行相关的求解，再根据已发生的坐标进行关于如问题一的求解，问题二则更看重优化与物理模型的结合。

2.3 问题三的分析

基于问题二的思想，由于未控制的量在问题二的量度上有所增加，依然是由问题二的思路，对更多的自变量进行求解，通过在右边增加一个因变量，欲使其中最有效的接近于额定功率即因变量要实际的越发于接近于 0,考虑到此时每一个定日镜的规格以及参数将会发生由不同而转变为相同，高度进行上下界划分，建立基于高度的又一个自变量，而与高度相关的公式为 DNI 的计算有关于真是法向辐射量的微小存量，介入 DNI 的计算我们与输入功率进行相关联，加之以控制有题目规定的规格，以此来进行有关第三问问题的求解。而关于每一个平均效率、量来说，基于第一问的物理建模以及求出之坐标以此来完成整一道问题的求解.

三、 模型假设

1. 通过查询发电厂的经纬坐标，发现该发电厂位于甘肃酒泉，查询中国天气网（http://www.weather.com.cn/cityintro/101160801.shtml），我们发现其降水量极少，年平均日照时数高达 3000 小时，我们假设当地天气对发电厂影响极小，可以省略。
2. 各个定日镜全年正常工作，没有出现无故损坏。
3. 定日镜维护时间相较工作时间占比极小或在非工作时间维护，可以忽略。
4. 定日镜能够及时清洗，避免表面杂质影响反射
5. 吸收塔周围 100 m 的厂房高度较小，不会影响定日镜的正常工作。

四、 符号说明

符号	说明	单位
φ	纬度北纬为正	度
θ	入射角	度
α_s	太阳高度角	度
γ_s	太阳方位角	度

δ	赤纬角	度
ω	太阳时角	度
D	春分作为第 0 天起算的天数	天
H	海拔	千米
ST	当地时间	时
$\vec{a}$	太阳至定日镜中心向量	
$\vec{b}$	集热器至定日镜中心向量	
η	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_{turnc}	集热器截断效率	
η_{ref}	镜面反射率	
dHR	镜面中心到集热器中心的距离	米
Pn	单位面积镜面平均输出热功率	千瓦/平方米
c_1	个体学习因子	
c_2	社会学习因子	

五、 模型的建立与求解

5.1 问题一模型的建立与求解

5.1.1 计算太阳入射角

考虑到太阳光线反射至吸收塔为镜面反射，即入射角等于反射角，因此太阳入射与镜面中心至吸收塔采热器中心连线的夹角为 2 倍入射角θ。

为了计算该夹角，我们需要计算太阳至定日镜中心连线的方向向量$\vec{a}$以及集热器中心至定日镜中心的向量$\vec{b}$，并利用如下公式计算反射角。

$$cos2\theta = \frac{\vec{a}\cdot\vec{b}}{|\vec{a}|\times|\vec{b}|} \tag{1}$$

首先计算$\vec{a}$,，正东方向为 x 轴，正北方向为 y 轴，竖直方向为 z 轴，以定日镜中心为原点建立直角坐标系；查询太阳方位角γ_s的定义[1]，即太阳光线在地面的投影与正北方向的夹角，因此可以确定$\vec{a}$在地表的投影为$(sin\gamma_s, cos\gamma_s)$，太阳高度角$\alpha_s$为太阳光线与地表的夹角，此时$\vec{a}$地表的投影长度为 1，可以得出$\vec{a} = (sin\gamma_s, cos\gamma_s, tan\alpha_s)$。

下面计算太阳高度角α_s和太阳方位角γ_s，将使用如下公式

$$sin\alpha_s = cos\delta\cos\varphi\cos\omega + \sin\delta\sin\varphi \tag{2}$$

$$cos\gamma_s = \frac{\sin\delta - \sin\alpha_s\sin\varphi}{\cos\alpha_s\cos\varphi} \tag{3}$$

φ为为当地纬度，北纬为正即 39.4°，ω为太阳时角，其公式为

$$\omega = \frac{\pi}{12}(\mathrm{ST} - 12) \tag{4}$$

ST为当地时间，δ为太阳赤纬角，其公式为

$$\sin\delta = \sin\frac{2\pi D}{365}\sin\frac{46.9\pi}{360} \tag{5}$$

D 为春分作为第 0 天起算的天数。

下面计算$\vec{b}$，吸收塔高度 80 m，集热器高度 8 m，定日镜高度 4m，这二者中心相对高度为 72m，由附件已知每个定日镜与吸收塔的相对位置（x，y），二者中心连线向量的地表投影即定日镜与吸收塔相对位置的相反数，得出$\vec{b}$= (-x，-y，72)。

利用 matlab 编程（见附录 1）计算入射角，其具体的流程如下：

Step1：将表格中的每个时间与春分的距离时间、每个定日镜坐标、5 个 ST 填入变量表；

Step2：利用循环计算 12 个月的太阳赤纬角、5 个时间的的太阳时角；

Step3：用户填入需要获取数据的月份，提取需要的赤纬角和与春分的距离时间；

Step4：将数据带入公式，得出当天 5 个时间点的太阳高度角α_s和太阳方位角γ_s，即$\vec{a}$；

Step5：利用每个定日镜的坐标及已知的定日镜与集热器相对高度，得出$\vec{b}$；

Step6：将两向量带入公式，得出入射角θ。

但是在计算过程中，我们发现部分太阳方位角γ_s出现复数的情况，由于计算方式是利用反函数，当参数绝对值大于 1 时，γ_s无解；同时，分析γ_s为虚数出现的位置，其只出现在中午 12 点，通过分析太阳的运动轨迹与当地经纬度，其正午 12 点位于正南，此时γ_s为 180°，而复数的γ_s其实部均为π，因此我们将复数部分γ_s的虚部剔除。

5.1.2 计算平均遮挡效率和截断效率

在计算阴影遮挡效率时，考虑到有以下三种阴影遮挡损失[2]：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。考虑到样本容量过大，通过研究太阳运动轨迹与定日镜分布，利用等效近似与旋转思想，可以将定日镜场划分成一个定日镜均匀排列而形成的圆环，通过选取圆环上任意一个最小矩形区域的定日镜进行分析计算，分组研究相邻定日镜的阴影遮挡损失，这里我们选取位于或靠近东西轴向分布的定日镜作为研究对象，可以减少判断选取相邻定日镜的误差。

运用微分法和光线追踪法，我们将每个发热板分成若干个点，通过判断该点射出的反射光线是否会被相邻定日镜遮挡，来统计未被遮挡的点的数量，得出定日镜的阴影遮挡效率；计算定日镜所处的圆环面积和吸收塔在该圆环上的投影面积，近似运算二者的商为吸收塔的对该发电厂的阴影遮挡效率，两个平均遮挡效率的积为实际平均遮挡效率。

首先计算单个定日镜在地面的阴影投影，对此将$\vec{a}$和$\vec{\mathrm{b}}$转换为单位向量，将二者相加的结果即定日镜法向向量，计算阴影定日镜在入射坐标系中的坐标，其表示为

$$X_i = -Xsin\gamma_s + Ycos\gamma_s \tag{6}$$

$$Y_i = -\sin\alpha_s(Xcos\gamma_s + Ysin\gamma_s) + Zcos\alpha_s \tag{7}$$

$$Z_i = cos\alpha_s(Ysin\gamma_s + Xcos\gamma_s) + Zsin\alpha_s - \frac{Z_m}{cos\theta} \tag{8}$$

X,Y,Z 为原始坐标，Z_m为该定日镜法向向量至地表的投影长度。

定日镜的镜面中心沿入射光方向的镜面投影坐标为

$$X_{mi} = X_i \cos(\alpha_H - \alpha_s) + Y_i sin\alpha_s sin(\alpha_H - \alpha_s) - Z_i cos\alpha_s sin(\alpha_H - \alpha_s) \tag{9}$$

$$\begin{aligned} Y_{mi} = -X_i \cos\gamma_H sin(\alpha_H - \alpha_s) + Y_i[cos\gamma_s sin\alpha_s cos(\alpha_H - \alpha_s) + sin\gamma_s cos\alpha_s] \\ - Z_i[cos\gamma_H cos\alpha_s cos\alpha_s sin(\alpha_H - \alpha_s) - sin\gamma_H sin\alpha_s] \end{aligned} \tag{10}$$

γ_H，α_H分别表示定日镜法向向量的俯仰角和太阳方位角.

微分定日镜，建立 1000×1000 的矩阵，每个赋值为 1，假定第 i 个点的位置为(k,l)，由于定日镜尺寸为 6m×6m,，制定如下判定：

1、当Y_{mi}<0，X_{mi}<0 时，同时满足

$$0.5 - \frac{1000 \times Y_{mi}}{6} < k < 1000.5$$

$$0.5 < l < 1000.5 + \frac{1000 \times X_{mi}}{6}$$

2、当Y_{mi}>0，X_{mi}<0 时，同时满足

$$0.5 < k < 1000.5 - \frac{1000 \times Y_{mi}}{6}$$

$$0.5 < l < 1000.5 + \frac{1000 \times X_{mi}}{6}$$

3、当Y_{mi}<0，X_{mi}>0 时，同时满足

$$0.5 - \frac{1000 \times Y_{mi}}{6} < k < 1000.5$$

$$0.5 + \frac{1000 \times X_{mi}}{6} < l < 1000.5$$

4、当Y_{mi}>0，X_{mi}>0 时，同时满足

$$0.5 < k < 1000.5 - \frac{1000 \times Y_{mi}}{6}$$

$$0.5 + \frac{1000 \times X_{mi}}{6} < l < 1000.5$$

满足如上条件时，说明该点被遮挡，统计矩阵 1 的个数并除以矩阵大小，即当前定日镜的遮挡效率。考虑真实情况，定日镜被遮挡时，被遮挡的点应该连成片，对此我们随机输出遮挡效率不为零的矩阵三维图。

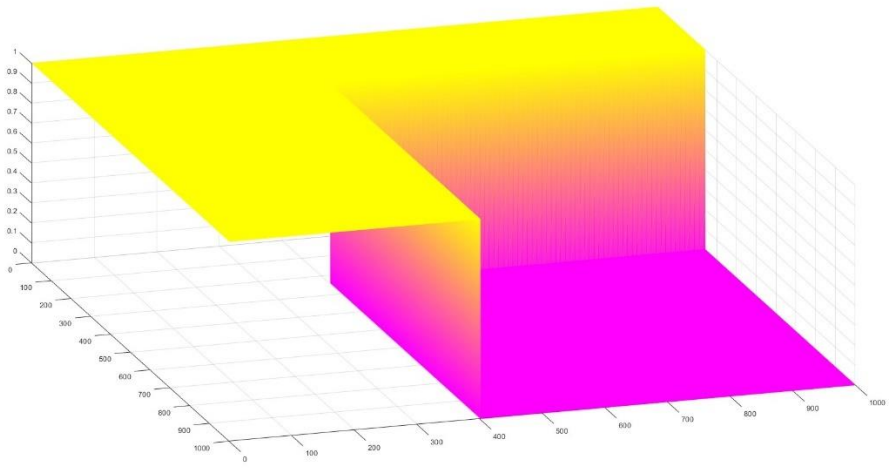


图 1、12 月 31 日 13 时 30 分某一定日镜的矩阵

可以发现，数值为 0 即被遮挡的部分是连续的，符合实际情况。

为减小误差，我们将日镜由远到近排序，利用多项式拟合研究的定日镜当日平均阴影遮挡效率的曲线，并将曲线积分，除以积分长度，得出所有定日镜当日平均阴影遮挡效率。利用 matlab 编程，我们发现利用多项式拟合且最高次为 6 次时，相关系数高达 0.95，具有较高的置信水平。

下面利用光线追踪法[3]，模拟每个光线从太阳射线反射到集热器表面的全过程，验证当前光线是否经过集热器所在平面，通过统计经过集热器的光线数量，得出当前定日镜的截断效率。

根据线性代数[4]中的运算法则，向量在不同的坐标系之间的转换关系有规定的公式。在本文的计算中用到的线性空间，笛卡尔直角坐标系之间的转换只需要进行平移和旋转。

建立地面坐标系 XS_0，以 x_0 轴朝正南，z_0 朝正西，y_0 垂直向上的直角坐标系。建立定日镜场坐标系 XS_1，以吸收塔为坐标原点，以矩形定日镜为例，令局部坐标系的 x_1 轴平行于水平转轴，y_1 轴在镜平面内垂直于 x_1 轴，z_1 轴为镜面法线，垂直镜面向上，镜中心点 H_1 为原点坐标为零[3]。

通过建立坐标系，进行坐标系空间转换，可以得到局部坐标系转换为地面坐标系的转换矩阵：

$$T10 = \begin{bmatrix} x_{N0} & z_{N0} & m_{z10}n_{x10} \\ y_{N0} & 0 & m_{z10}n_{x10} - l_{z10}n_{x10} \\ z_{N0} & -x_{N0} & -m_{z10}l_{x10} \end{bmatrix} \tag{11}$$

建立吸热塔坐标系 XS_2，以 x_2 轴朝正南，y_2 朝正西，z_2 垂直向下的直角坐标系。由于吸热塔的位置已定，且俯仰角为零，则地面坐标系转换到吸热塔坐标系的转换矩阵：

$$T30=\begin{bmatrix} 0 & -1 & 0 \\ 0 & 0 & -1 \\ 1 & 0 & 0 \end{bmatrix} \tag{12}$$

所以 XS1 上的点 $P_1(x_1, y_1, 0)$ 点变换到吸热塔局部坐标系 XS_3 内的坐标 P_3 可以表示为

$$P_3 = T30^T \cdot T10 \cdot P_1 \tag{13}$$

通过计算入射光线的向量，在已知镜面法线向量的情况下，求解反射光线。太阳入射光线方向向量即$\vec{a}$，转换$\vec{a}$在定日镜场坐标系上，定日镜微分，建立 100×100 的矩阵，每个赋值为 0，利用反射定律

$$\vec{a}+\vec{r}=2\times\vec{n}\cdot\cos\theta \tag{14}$$

$\vec{r}$分别为反射光，$\vec{n}$为平面镜法向向量。

计算每个点的反射光线向量在集热器平面的投影，其投影若满足-3.5<=x<=3.5 和-4<=y<=4，则说明其投影在集热器上，赋值为 1；最后统计 1 的个数并除以矩阵大小，即截断效率。同理，考虑真实情况，被截断的点应该连成片，对此我们随机截断遮挡效率不为零的矩阵三维图。

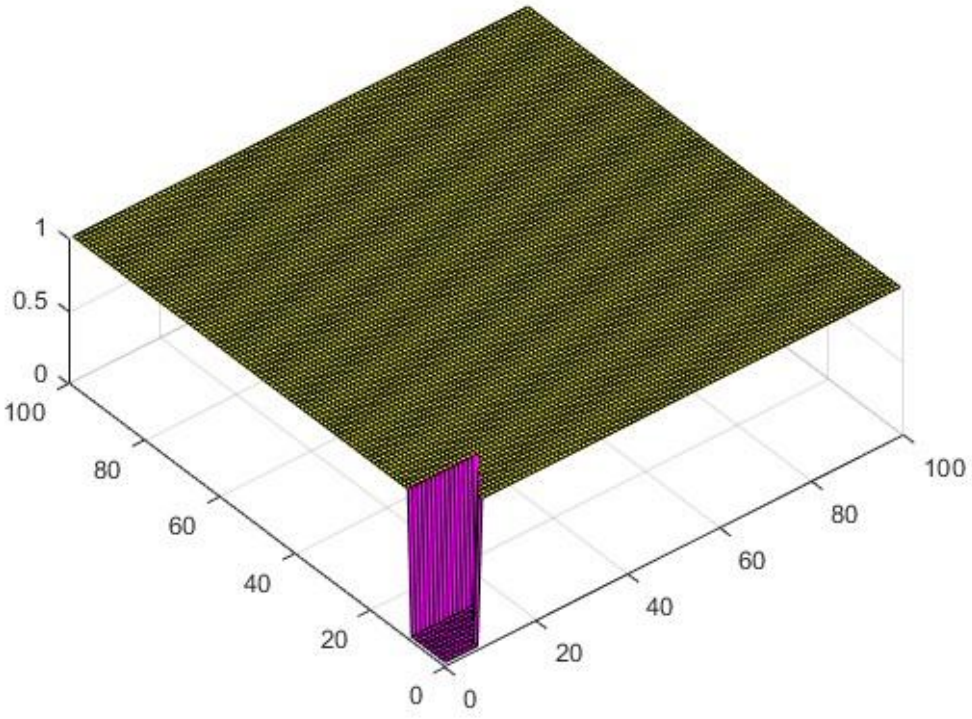


图 2、11 月 31 日 15 时某一定日镜的矩阵

可以发现，数值为 0 即被截断的部分是连续的，符合实际情况。

5.1.3 模型求解

光学效率η的计算公式为

$$\eta=\eta_{sb}\eta_{\cos}\eta_{at}\eta_{turnc}\eta_{ref} \tag{15}$$

大气透射率η_{at}公式为

$$\eta_{\mathrm{at}}=0.99321-0.0001176\mathrm{dHR}+1.97\times10^{-8}\times\mathrm{dHR} \tag{16}$$

余弦效率$\eta_{\cos}$的公式[4]1 为

$$\eta_{\cos}=\cos\theta \tag{17}$$

已知平均遮挡效率η_{sb}，镜面反射效率η_{ref} =0.92，利用 matlab 编程(见附录 1)，计算当前日期选取的 9 组定日镜的 5 个时间点的光学效率η，然后计算平均光学热效率，得出结果。

下面计算单位面积镜面平均输出热功率 Pn，首先近似计算法向直接辐射辐照度 DNI，即当地单位面积接受太阳辐射功率，DNI 与该镜面的光学效率乘积即当前镜面的单位面积镜面平均输出热功率，近似运算公式为

$$DNI=G_0[a+be^{(-\frac{c}{\sin\alpha_S})}] \tag{18}$$

$$a=0.4237-0.00821(6-H)^2 \tag{19}$$

$$b = 0.5055 + 0.00595(6.5 - H)^2 \quad (20)$$

$$c = 0.2711 + 0.001858(2.5 - H)^2 \quad (21)$$

求出当天 5 个时间点的 DNI，求平均后并乘以该定日镜的光学效率，即当前镜面平均输出热功率 Pn，随后统计所有定日镜的镜面平均输出热功率，求平均后即当天的单位面积镜面平均输出热功率。

计算结果如下：

日期	平均光学效率	平均余弦效率	平均阴影遮挡效率	平均截断效率	单位面积镜面平均输出热功率 (kW/m2)
1 月 21 日	0.6182	0.7531	0.9184	0.9999	0.5709
2 月 21 日	0.6322	0.7634	0.9259	0.9999	0.6258
3 月 21 日	0.6337	0.7711	0.9186	0.9998	0.6573
4 月 21 日	0.6240	0.7746	0.9010	0.9999	0.6670
5 月 21 日	0.6259	0.7735	0.9150	0.9999	0.6779
6 月 21 日	0.6264	0.7721	0.9176	0.9999	0.6811
7 月 21 日	0.6258	0.7735	0.9148	0.9998	0.6778
8 月 21 日	0.6238	0.7745	0.9005	0.9999	0.6661
9 月 21 日	0.6341	0.7708	0.9196	0.9999	0.6564
10 月 21 日	0.6307	0.7622	0.9251	0.9998	0.6197
11 月 21 日	0.6167	0.7521	0.9144	0.9999	0.5652
12 月 21 日	0.6109	0.7479	0.9144	0.9999	0.5413

年平均光学效率	年平均余弦效率	年平均阴影遮挡效率	年平均截断效率	年平均输出热功率 (MW)	单位面积镜面年平均输出热功率 (kW/m2)
0.6252	0.7657	0.9154	0.9999	39.82	0.633875

5.2 问题二模型的建立与求解

考虑到多个参数计算最优解的庞大运算量，下面将基于模拟退火算法和粒子群同时求解，并取用最优解。

5.2.1 基于模拟退火的模型求解

模拟退火的核心思想迭代求解的过程寻找最优解，基于物理中固体物质的退火过程，温度高的时候更容易接受其他局部最优解，其接受的概率为

$$p \propto e^{\Delta f * C(t)} \quad (22)$$

C(t)为温度函数。

模拟退火算法能基于接受局部最优解的概率自发寻找更优的局部最优解，并经过多次迭代后趋于全局最优，我们的目的是计算最大的镜面平均输出热功率，则接受的概率为

$$p=\begin{cases}1 & F(x+1)>F(x)\\ e^{-\frac{[F(x+1)-F(x)]}{kT}} & F(x+1)\le F(x)\end{cases} \tag{23}$$

利用 matlab 编程（见附录 1）多次迭代进行求解。

5.2.2 基于粒子群的模型求解

粒子群算法是基于个体对群体的信息共享，通过局部最优解共享信息素，使得求解过程逐渐从无序到有序，从而更快寻找全局最优解，其核心公式为：

该例子第 d 步的速度 =上一步自身的速度惯性+自我认知部分+社会认知部分

$$v_i^j = wv_i^{j-1} + c_1r_1\left(pbest_i^j - x_i^j\right) + c_2r_2(gbest^j - -x_i^j) \tag{24}$$

v_i^j为第 j 次迭代时第 i 个粒子的速度矢量；x_i^j为第 j 次迭代时第 i 个粒子所在的位置；w 为惯性权重；c_1为粒子的个体学习因子；c_2为粒子的社会学习因子；$pbest_i^j$到第 j 次迭代第 i 个粒子的最好的位置；$gbest^j$为到第 j 次迭代所有粒子经过的最好的位置。

第 j+1 次迭代解为：

$$x_i^{j+1} = x_i^j + v_i^j \tag{25}$$

为了避免粒子过多在局部搜索，造成过度的计算冗余与粒子缺乏寻找下一最优解的自主性，对此我们加入非对称学习因子[6]，随着迭代次数的增加，个体学习因子c_1线性递增，社会学习因子c_2线性递减；相比固定的学习因子，能够促进粒子向全局最优点的收敛。

$$c_1^j = c_1^i - (c_1^f - c_1^i) \times \frac{j}{K} \tag{26}$$

$$c_2^j = c_2^i + (c_2^f - c_2^i) \times \frac{j}{K} \tag{27}$$

c^i为学习因子的初始值，c^f学习因子的最终值，K 为总迭代次数。

利用 matlab 编程（见附录 3），以吸收塔为坐标原点建立坐标系，并计算其余参数的最优解，进行定日镜的均匀排布（见附录 4）。

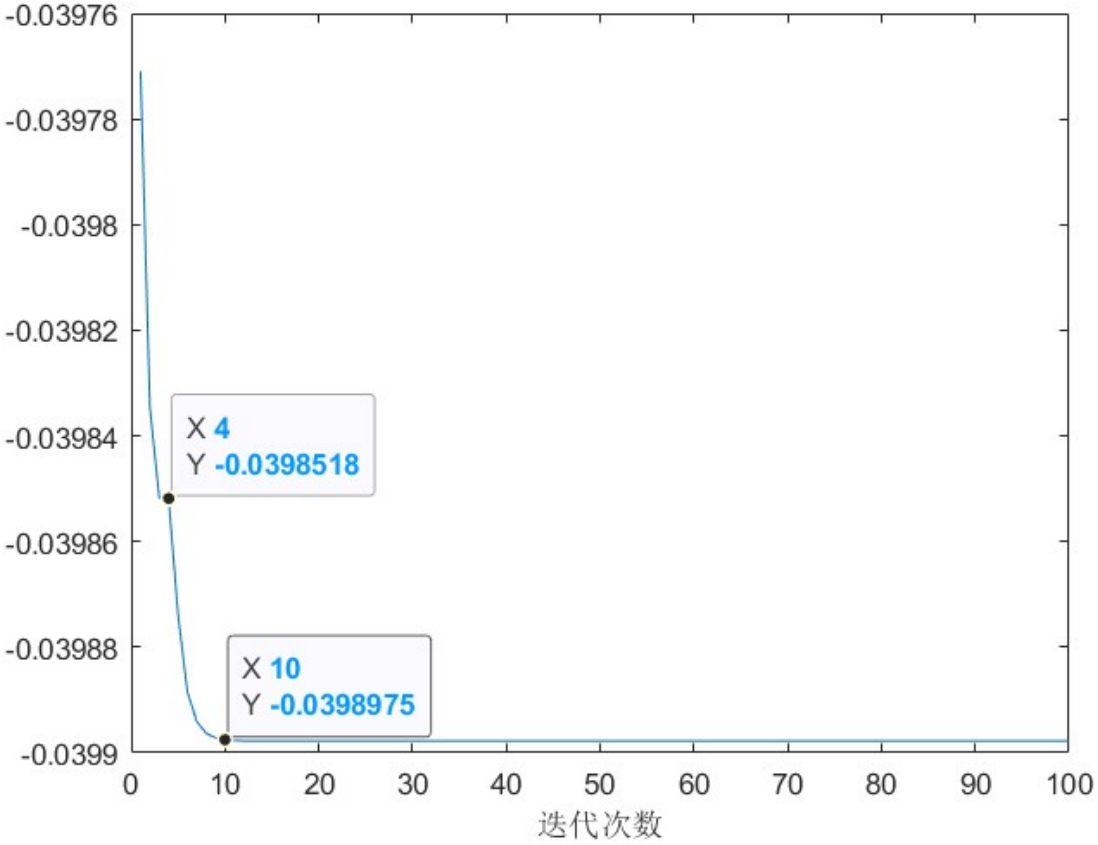


图 3、粒子群迭代求解

5.2.3 模型比较

在模拟退火中，对比非对称学习因子以及模拟退火的区别在于，模拟退火反倒更容易陷入局部最优解的情况，其次在编写算法的同时，无约束的条件的添加在运行时间比较短的情况之下并未有找到关于适应性函数的全局最优解，在计算最小值时，作为自变量的的两个坐标经常容易寻找到定义域之外的值，在计算定义域外的值，但在减少自变量的范围之下，关于模拟退火计算关于适应性函数的最优化问题却能够得以缓解，可见其关于全局搜索能力在多维方程之中显得比较差，且受因素影响比较大，由下图的表格可知：

模拟退火维度升高的差异		
变量个数	五维	三维
1	-350	-183.425
2	-350	30.96046
3	1500	1500
4	2	6(人为)
5	2	6(人为)
适应性函数值	-29.584527	-0.03578

接下来是粒子群算法的误差分析:

PSO 粒子群算法的维度差别		
变量个数	五维	六维
1	-169.7	109.6
2	-274.9	9.7
3	1500	1500
4	8	8
5	8	2.1
6	4(人为)	3
适应性函数值	-0.0396	-0.0399

可见实际上在多个变量的运算时应该选用 PSO 粒子群算法，在计算多个维度的变量的时候具有明显的低误差性。其次更有效地防止自变量落入定义域之外而导致计算结果算错以及巨大的误差。粒子群在这方面显著优于模拟退火

日期	平均光学效率	平均余弦效率	平均阴影遮挡效率	平均截断效率	单位面积镜面平均输出热功率 (kW/m2)
1 月 21 日	0.7003	0.8133	0.8965	0.9999	0.7393
2 月 21 日	0.7162	0.8245	0.9039	0.9999	0.7561
3 月 21 日	0.7179	0.8328	0.8967	0.9999	0.7579
4 月 21 日	0.7069	0.8366	0.8796	0.9999	0.7463
5 月 21 日	0.7091	0.8354	0.8932	0.9999	0.7486
6 月 21 日	0.7096	0.8339	0.8958	0.9999	0.7491
7 月 21 日	0.7090	0.8354	0.8930	0.9999	0.7484
8 月 21 日	0.7067	0.8365	0.8791	0.9999	0.7461
9 月 21 日	0.7184	0.8325	0.8977	0.9999	0.7583
10 月 21 日	0.7145	0.8232	0.9031	0.9999	0.7543
11 月 21 日	0.6987	0.8123	0.8926	0.9999	0.7376
12 月 21 日	0.6921	0.8077	0.8926	0.9998	0.7306

年平均光学效率	年平均余弦效率	年平均阴影遮挡效率	年平均截断效率	年平均输出热功率 (MW)	单位面积镜面年平均输出热功率 (kW/m2)
0.7083	0.8270	0.8934	0.9999	67.53	0.7477

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 (m2)
(0，-4.91)	7.8×5.9	4	1750	80535

5.3 问题三模型的建立与求解

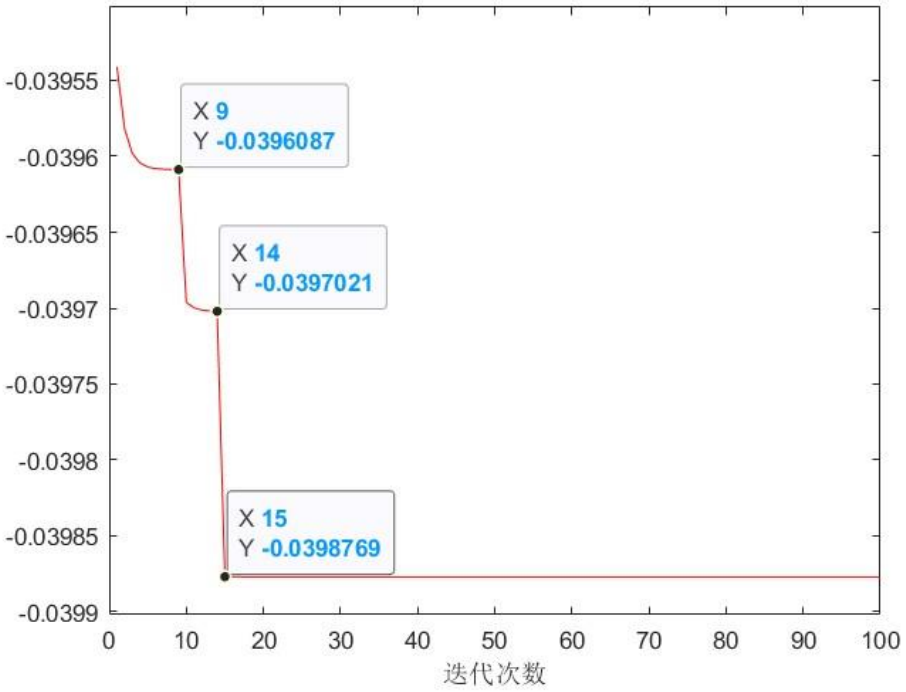
5.3.1 基于问题二的模型思想构建

通过在第二问的求解，问题三就是在第二问的基础上增加更多的自变量，即更高的维度的第二问的解法去求解关于第三问的相关自变量，在进行第三问建模前，必须要重启关于 DNI 与海拔的精确计算，再将其返回至关于最大输出热效率的公式之中以此建立关于第三问的模型。考虑到部分参数与各个定日镜尺寸无关，直接引用问题二结果。

5.3.2 基于粒子群的模型求解

由于函数的相关维度在原来第二问还要高的基础之上，局部最优解的寻找是尤为重要的，通过引用非对称学习因子的粒子群模型，在迭代的最初时间，我们通过该模型有利于在全局扩大搜索能力，以高效率的迭代次数寻找局部最优解，再然后缩短寻找时间对函数局部搜索能力进行放大。以此来找到关于最优解的数值，以及返回相关变量的求解

以下，附上关于更高以为都的迭代求解图：



由图可以清晰的看出非对称学习因子寻找局部最优解能力。

日期	平均光学效率	平均余弦效率	平均阴影遮挡效率	平均截断效率	单位面积镜面平均输出热功率 (kW/m2)
1 月 21 日	0.7393	0.8174	0.9046	0.9999	0.8038
2 月 21 日	0.7481	0.8204	0.9120	0.9999	0.8134
3 月 21 日	0.7440	0.8224	0.9048	0.9999	0.8089
4 月 21 日	0.7444	0.8389	0.8875	0.9999	0.8094
5 月 21 日	0.7657	0.8497	0.9012	0.9999	0.8324

6 月 21 日	0.7789	0.8618	0.9039	0.9999	0.8468
7 月 21 日	0.7732	0.8582	0.9010	0.9999	0.8406
8 月 21 日	0.7540	0.8501	0.8870	0.9999	0.8197
9 月 21 日	0.7637	0.8432	0.9058	0.9999	0.8303
10 月 21 日	0.7625	0.8369	0.9112	0.9999	0.8290
11 月 21 日	0.7446	0.8278	0.8997	0.9998	0.8096
12 月 21 日	0.7344	0.8157	0.9006	0.9997	0.7984

年平均光学效率	年平均余弦效率	年平均阴影遮挡效率	年平均截断效率	年平均输出热功率 (MW)	单位面积镜面年平均输出热功率 (kW/m2)
0.7544	0.836875	0.9017	0.9999	61.618	0.8202

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 (m2)
（0，-4.91)	\	\	1750	75125.592

六、 模型的分析与检验

6.1 模型的分析

在进行阴影遮挡效率的计算时，我们研究定日镜的遮阳模式，使用了相邻坐标投影判断法和光线向量转换判断法，其中相邻坐标投影判断法是利用相邻定日镜在同一坐标轴的重叠度来推算阴影遮挡效率，光线向量转换法则是通过判断反射光向量的坐标是否会落在吸热器表面的方法，来判断定日镜的阴影遮挡效率。通过比较两种方法的结果，我们发现相邻坐标投影判断法可以避免由于定日镜分布不均导致的系统误差，并且，对于每个时刻的定日镜反射分析，光线向量转换法在使用 Matlab 编程时，程序运算过于冗长。因此，经过误差分析，我们采用相邻坐标投影判断法，对阴影遮挡模型进行建模分析；

6.2 模型的检验

在得到定日镜的平均截断效率时，计算得出的理论效率与实际效率有着较大误差，因此，为了消除误差疑虑，我们对各个定日镜在某一时刻的截断效率进行三维建模（图 2），通过实际演算，可以发现图形连续且分布规律，利用图形检验的方法，我们对截断效率的验证得到了有力的支撑。

七、 模型的评价与改进

7.1 模型的优点

1、 利用粒子群算法计算，并引入非对称学习因子，促进模型从局部最优解向全局最优解收敛，提高模型精度。

2、 利用微分的思想求截断效率和阴影遮挡效率，简化计算流程，提高模型效率。

3、 由于计算结果均为平均值，利用等效近似和旋转思想，选取最具代表性的定日镜组进行分析，减少运算量，提高模型效率。

4、 由于求平均时样本数据不是线性，同时考虑太阳轨迹，样本数据由对称性，直接求和除样本数求平均无法精确表示样本特性，使用拟合曲线并积分求平均能提高模型效率。

7.2 模型的缺点

1、 等效近似和旋转思想可以减少运算量，但是选取的定日镜无法完全准确表示所有定日镜的平均参数，必然存在误差。

2、 仅选用每日 5 个时间点和 12 个日期进行运算，计算全年平均值时存在误差。

3、 求截断效率和阴影遮挡效率时，结果的精度受矩阵大小所限制。

7.3 模型的改进

1、 将模型 2 的运算结果作为模型 1 的参数进行编程，以提高模型的自动化程度。

2、 基于 MATLAB 对于本次文章使用算法模型的改进，本文多半以求取函数的最小值为主要，忽略了大值的求取，可以适当增加对于概率的选取，决定是对一个值进行选大或者取小的处理，以及一些判定的需求

3、 基于文章所用的模型，我们可以首先为算法加入随机权重，或者惯性权重，在前期可以提高全局搜索能力，后期打捞式搜索，排除局部最优解所带来的影响；其次可以改变学习因子，让线性的算法因子复杂度提高，防止粒子过早收敛，压缩因子，还可以确保 PSO 的收敛性，取消对速度边界的限制，提高运算速度

4、 可以提早在计算得出的算法进行加入全局检查，假如已搜索到全局最优，则立马结束计算，提高计算机的运行效率

7.4 模型的推广

1、 对于阴影遮挡效率的模型，由于其本质是利用转换坐标系投影并计算重叠度的核心思想，可以将该模型推广到计算表面强度问题、二次曲面重叠问题等其他与计算重叠度的问题中；

2、 对于粒子群算法的模型，由于其本质是通过迭代优化找极值的中心思想，可以将该算法模型推广到旅行商问题、生物与自然迁徙问题、布置拓展问题等分析研究运动极值的问题中；

八、 参考文献

[1]. 百度百科，太阳方位角，https://baike.baidu.com/item/%E5%A4%AA%E9%98%B3%E6%96%B9%E4%BD%8D%E8%A7%92/2772684

[2]. 张平,奚正稳,华文瀚等.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(06):5-8.

[3]. 张红鑫,卢振武,李凤有等.重叠复眼光学模型的建立与分析[J].光子学报,2007(06):1106-1109.

[4]. 史雷,赵滨滨,徐晓萌等.分布式电源对区域配电网合理线损标杆计算的影响[J].电力系统及其自动化学报,2018,30(10):132-138.DOI:10.19635/j.cnki.csu-epsa.000065.

[5]. 丁婷婷,祝雪妹等.塔式系统圆形镜场中余弦效率分布的研究[A].科学技术与工程,2012(11):1671-1815.

[6]. 毛开富, 包广清, 徐驰. 基于非对称学习因子调节的粒子群优化算法[J]. 计算机工程, 2010(19):188-190

附录

附录 1

介绍：由 matlab 编写，问题一所需数学模型

```
load 国赛.mat
x=input("请输入月份");
for i=1:5
    omega(i,1)=pi*(ST(i,1)-12)/12;
end

for i=1:12
    delta(i,1)=asin(sin(2*pi*D(i,1)/365)*sin(2*pi*23.45/360));
end

for j=1:5
    alpha(j,1)=asin(cos(delta(x,1))*cosd(39.4)*cos(omega(j,[1])) ...
         +sin(delta(x,1))*sind(39.4));
end

clear i;
clear j;

%%计算太阳方位角
for i=1:5
    gamma(i,1)=acos((sin(delta(x,1))-sin(alpha(i,1))*sind(39.4)) ...
       /(cos(alpha(i,1))*cosd(39.4)));
end
clear i;
gamma(3,1)=real((gamma(3,1)));

%%计算太阳至定向镜向量
for i=1:5
    sun_vector(i,1)=sin(gamma(i,1));
end
for i=1:5
    sun_vector(i,2)=cos(gamma(i,1));
end
for i=1:5
    sun_vector(i,3)=sin(alpha(i,1));
end

%%计算塔至定向镜向量
for i=1:1745
    tower_vector(i,1)=-location(i,1);
```

```
    tower_vector(i,2)=-location(i,1);
end

tower_vector(:,3)=72;
%%计算入射角
for i=1:1745
    for j=1:5

seta(i,j)=acos((sun_vector(j,1)*tower_vector(i,1)+sun_vector(j,2)*tower_v
ector(i,2) ...

+sun_vector(j,3)*tower_vector(i,3))/(sqrt(tower_vector(i,1)^2+tower_vecto
r ...

(i,2)^2+tower_vector(i,3)^2)*sqrt(sun_vector(j,1)^2+sun_vector(j,2)^2+ ..
.
       sun_vector(j,3)^2)));
       seta(i,j)=seta(i,j)/2;
    end
end
%%导入选取的定日镜坐标信息
Tlocation=[-337.132 0;-310.161 0;-296.676 0;-269.706 0;-256.22 0;-229.25
0;-202.179 -6.354;-202.179 6.354;-188.794 0;-175.186 -6.555;-175.186
6.555;-161.823 0;-148.218 -5.973;-148.218 5.973;-134.713 -6.138;-134.713
6.138;-121.368 0;-107.882 0;];
%%计算定日镜追踪太阳的定位角
%%%太阳向量的计算
for k=1:5
    sunlocation(k,1)=sin(gamma(k,1));
end

for l=1:5
    sunlocation(l,2)=cos(gamma(l,1));
end

for m=1:5
    sunlocation(m,3)=sin(alpha(m,1));
end

clear k,l,m
%%%标准化太阳向量
for k=1:5
  for l=1:3
   ssunlocation(k,l)=sunlocation(k,l)/(sqrt(sunlocation(k,1)^2+ ...
```

```
   sunlocation(k,2)^2+sunlocation(k,3)^2));
  end
end
clear k,l
%%%标准化塔向量
for k=1:1745
 for l=1:3
    ttlocation(k,l)=tower_vector(k,l)/(sqrt(tower_vector(k,1)^2+ ...
    tower_vector(k,2)^2+tower_vector(k,3)^2));
 end
end

%%%计算追踪太阳的定位角（9：00）
for i=1:1745
    xita(i,1)=tower_vector(i,1)+sunlocation(1,1);
    xita(i,2)=tower_vector(i,2)+sunlocation(1,2);
    tans(i,1)=tan(xita(i,1)/xita(i,2));
end
%%%计算追踪太阳的定位角（10：30）
for i=1:1745
    xita(i,1)=tower_vector(i,1)+sunlocation(2,1);
    xita(i,2)=tower_vector(i,2)+sunlocation(2,2);
    tans(i,2)=atan(xita(i,1)/xita(i,2));
end
%%%计算追踪太阳的定位角（12：00）
for i=1:1745
    xita(i,1)=tower_vector(i,1)+sunlocation(3,1);
    xita(i,2)=tower_vector(i,2)+sunlocation(3,2);
    tans(i,3)=atan(xita(i,1)/xita(i,2));
end
%%%计算追踪太阳的定位角（13：30）
for i=1:1745
    xita(i,1)=tower_vector(i,1)+sunlocation(4,1);
    xita(i,2)=tower_vector(i,2)+sunlocation(4,2);
    tans(i,4)=atan(xita(i,1)/xita(i,2));
end
%%%计算追踪太阳的定位角（15：00）
for i=1:1745
    xita(i,1)=tower_vector(i,1)+sunlocation(5,1);
    xita(i,2)=tower_vector(i,2)+sunlocation(5,2);
    tans(i,5)=atan(xita(i,1)/xita(i,2));
end
%%计算追踪点定位坐标
%%%计算横坐标 Xmi
```

```
%%%%九组相邻定日镜在 X0Y0Z0 坐标系的坐标
for n=1:9
   Alocation(n,1)=Tlocation(2*n,1)-Tlocation(2*n-1,1);
end
for j=1:9
    Alocation(j,2)=Tlocation(2*j,2)-Tlocation(2*j-1,2);
    Alocation(j,3)=0;
end
%%%转换为入射坐标系坐标
%%%%五个时刻的横坐标
for b=1:9
    for c=1:5
    Xi(b,c)=(-
Alocation(b,2)*sin(gamma(c,1)))+Alocation(b,1)*cos(gamma(c,1));
    Yi(b,c)=-
sin(alpha(c,1))*((Alocation(b,2)*cos(gamma(c,1))+(Alocation(b,1)*sin(gamm
a(c,1))));

Zi(b,c)=cos(alpha(c,1))*((Alocation(b,1)*sin(gamma(c,1)))+(Alocation(b,2)
*cos(gamma(c,1))));
    end
end
%%%%计算在镜面的投影 Z 坐标
%%%%%计算 Zm 的距离
for g=1:9
    Zms(g,1)=sqrt((Alocation(g,1))^2+(Alocation(g,2))^2);
end
E=[seta(39,:);seta(60,:);seta(122,:);seta(157,:);seta(192,:);seta(131,:);
seta(309,:);seta(350,:);seta(392,:);]
for h=1:9
    for u=1:5
        Zcut(h,u)=(Zms(h,1))/(cos(E(h,u)));
    end
end
Zis=Zi-Zcut;
%%%%计算阴影定日镜的镜面中心沿入射光方向在被阴影镜面上的投影坐标
%%%%确定 EH 的值
%%%%%整理 EH
%%%%%%算九组的镜面法线的坐标
cttlocation=[ttlocation(39,:);ttlocation(60,:);ttlocation(122,:);ttlocati
on(157,:);ttlocation(192,:);ttlocation(131,:);
ttlocation(309,:);ttlocation(350,:);ttlocation(392,:)];
for i=1:9
    for j=1:5
```

```
        XX(i,j)=cttlocation(i,1)+sunlocation(j,1);
        YY(i,j)=cttlocation(i,2)+sunlocation(j,2);
        ZZ(i,j)=cttlocation(i,3)+sunlocation(j,3);
    end
end
%%%%%%计算底边长度和 EH
for i=1:9
    for j=1:5
        log(i,j)=sqrt(XX(i,j)^2+YY(i,j)^2);
        EH(i,j)=atan(log(i,j)/ZZ(i,j));
    end
end
clear i,j

%%%%计算 AH-A(AA)的值
%%%%%整理 AH
P=pi/2;
M=repmat(P,1745,5);
tanss=M-tans;
F=[tanss(39,:);tanss(60,:);tanss(122,:);tanss(157,:);tanss(192,:);tanss(1
31,:);
tanss(309,:);tanss(350,:);tanss(392,:);]
%%%%%计算 AH-A（AA）的矩阵
A=[(gamma(1,1)) (gamma(2,1)) (gamma(3,1)) (gamma(4,1)) (gamma(5,1))];
A1=repmat(A,9,1);
AA=F-A1;
%%%%%算投影坐标的横坐标
for i=1:9
   for j=1:5
    Xmi(i,j)=Xi(i,j)*cos(AA(i,j))+Yi(i,j)*sin(alpha(j,1))*sin(AA(i,j))-
Zis(i,j)*cos(alpha(j,1))*sin(AA(i,j));
   end
end
%%%%%算投影坐标的竖坐标
for i=1:9
   for j=1:5
    Ymi(i,j)=-((Xi(i,j))*cos(EH(i,j))*sin(AA(i,j)))
+Yi(i,j)*(((cos(EH(i,j)))*(sin(alpha(j,1)))*(cos(AA(i,j))))+(sin(EH(i,j))
)*cos(alpha(j,1)))
-(Zis(i,j))*((cos(EH(i,j)))*(cos(alpha(j,1)))*(cos(AA(i,j)))-
sin(EH(i,j))*sin(alpha(j,1)));
   end
end
clear i,j,b,c,n
```

```
sum=0;
for i=1:9
 for j=1:5
  for k=1:1000
   for l=1:1000
     Sum(k,l)=1;
 if Ymi(i,j)<0&&Xmi(i,j)<0
  if (0.5-1000*Ymi(i,j)/6)<k&&0.5<l&&l<1000.5+1000*Xmi(i,j)/6
     Sum(k,l)=0;
end
 elseif Ymi(i,j)>0&&Xmi(i,j)<0
  if 0.5<k&&k<1000.5-1000*Ymi(i,j)/6&&0.5<l&&l<1000+0.5+1000*Xmi(i,j)/6
     Sum(k,l)=0;
end
 elseif Ymi(i,j)<0&&Xmi(i,j)>0
  if (0.5-1000*Ymi(i,j)/6)<k&&0.5+1000*Xmi(i,j)/6<l
     Sum(k,l)=0;
end
 elseif Ymi(i,j)>0&&Xmi(i,j)>0
  if 0.5<k&&k<1000.5-1000*Ymi(i,j)/6&&0.5+1000*Xmi(i,j)/6<l
     Sum(k,l)=0;
end
end
sum=Sum(k,l)+sum;
end
end
eta_sb(i,j)=sum/1000000;
if eta_sb(i,j)<1;
end
sum =0;
end
end
clear sum
%%%%计算平均阴影遮挡效率
for i=1:9
    sun1(i,1)=sum(eta_sb(i,:))/5;
    sun1(i,2)=i;
end
eta_sb1(:,1)=sun1(:,1);
P=polyfit(sun1(:,2),sun1(:,1), 6);
func=@(x)(P(1,1)*power(x,6)+P(1,2)*power(x,5)+P(1,3)*power(x,4)+P(1,4)*po
wer(x,3)+P(1,5)*power(x,2)+P(1,6)*x+P(1,7));
eta_sb_n = integral(func,1,9)/8;
clear P,func,sun1,j,l;
```

```
S=pi*(350^2-100^2);
for i=1:5
  eta_sb2(i,2)=1-7*80*cot(alpha(i,1))/S;
end
eta_sb2(1,1)=sum(eta_sb2(:,2))/5;
clear S,eta_sb2(:,2);


%%计算余弦效率
Tlocation=[-337.132 0;-310.161 0;-296.676 0;-269.706 0;-256.22 0;-229.25
0;
    -202.179 -6.354;-202.179 6.354;-188.794 0;-175.186 -6.555;-175.186
6.555;-161.823 0;-148.218 -5.973;-148.218 5.973;-134.713 -6.138;-134.713
6.138;-121.368 0;-107.882 0;];
Tlocation(:,3)=72;
for i=1:5
    eta_cos(1,i)=cos(seta(30,i));
end
for i=1:5
    eta_cos(2,i)=cos(seta(60,i));
end
for i=1:5
    eta_cos(3,i)=cos(seta(122,i));
end
for i=1:5
    eta_cos(4,i)=cos(seta(157,i));
end
for i=1:5
    eta_cos(5,i)=cos(seta(192,i));
end
for i=1:5
    eta_cos(6,i)=cos(seta(131,i));
end
for i=1:5
    eta_cos(7,i)=cos(seta(309,i));
end
for i=1:5
    eta_cos(8,i)=cos(seta(350,i));
end
for i=1:5
    eta_cos(9,i)=cos(seta(392,i));
end
```

```
for i=1:9
    eta_cos1(i,1)=sum(eta_cos(i,:))/5;
end
eta_cos_n=sum(eta_cos1(:,1))/9;

%%计算大气透射率
for i=1:9
    dHR(i,1)=sqrt(Tlocation(2*i-1,1)^2+Tlocation(2*i-1,2)^2+Tlocation(2*i-
1,3)^2);
end

clear i;
for i=1:9
    eta_at(i,1)=0.99321-0.0001176*dHR(i,1)+1.97*dHR(i,1)^2/100000000;
end

eta_at_n=sum(eta_at(:,1))/9;
clear dHR;

%%%%由镜面坐标系转换到地面坐标系的转换矩阵 T10
%%%number=input('请输入定日镜编号');
number=356;
%%%%%镜面法线的空间向量
for i=1:5
    station(1,i)=tower_vector(number,1)+sunlocation(i,1);
    station(2,i)=tower_vector(number,2)+sunlocation(i,2);
    station(3,i)=tower_vector(number,3)+sunlocation(i,3);
end
clear i
%%%%%%标准化法线向量
for j=1:5
    for i=1:3

sstation(i,j)=station(i,j)/sqrt((station(1,j)^2)+((station(2,j)^2))+((sta
tion(3,j))^2));
    end
end
clear i j
%%%%%%求镜面法线与地面系 X 轴的方向余弦 lz
xcut=[1;0;0];
lcut=repmat(xcut,1,5);
XXLDC=sstation.*lcut;
for i=1:5
XXLDCH(1,i)=sum(XXLDC(:,i));
```

```
end
for j=1:5

lz(1,j)=XXLDCH(1,j)/sqrt((sstation(1,j)^2)+((sstation(2,j)^2))+((sstation
(3,j))^2));
end
clear i j
%%%%%%求镜面法线与地面系 Y 轴的方向余弦 mz
ycut=[0;1;0];
mcut=repmat(ycut,1,5);
YXLDC=sstation.*mcut;
for i=1:5
YXLDCH(1,i)=sum(YXLDC(:,i));
end
for j=1:5

mz(1,j)=YXLDCH(1,j)/sqrt((sstation(1,j)^2)+((sstation(2,j)^2))+((sstation
(3,j))^2));
end
clear i j
%%%%%%求镜面法线与地面系 Z 轴的方向余弦 nz
zcut=[0;0;1];
ncut=repmat(zcut,1,5);
ZXLDC=sstation.*ncut;
for i=1:5
ZXLDCH(1,i)=sum(ZXLDC(:,i));
end
for j=1:5

nz(1,j)=ZXLDCH(1,j)/sqrt((sstation(1,j)^2)+((sstation(2,j)^2))+((sstation
(3,j))^2));
end
clear i j
%%%%%%求转换矩阵 T10
for i=1:5
 T31(1,i)=mz(1,i)*(-sstation(1,i));
 T32(1,i)=nz(1,i)*sstation(3,i)+lz(1,i)*sstation(1,i);
 T33(1,i)=-mz(1,i)*sstation(3,i);
end
T101=[sstation(1,1) sstation(3,1) T31(1,1);sstation(2,1) 0
T32(1,1);sstation(3,1) -sstation(1,1) T33(1,1)];

T102=[sstation(1,2) sstation(3,1) T31(1,2);sstation(2,1) 0
T32(1,2);sstation(3,1) -sstation(1,1) T33(1,2)];
```

```
T103=[sstation(1,3) sstation(3,1) T31(1,3);sstation(2,1) 0
T32(1,3);sstation(3,1) -sstation(1,1) T33(1,3)];

T104=[sstation(1,4) sstation(3,1) T31(1,4);sstation(2,1) 0
T32(1,4);sstation(3,1) -sstation(1,1) T33(1,4)];

T105=[sstation(1,5) sstation(3,1) T31(1,5);sstation(2,1) 0
T32(1,5);sstation(3,1) -sstation(1,1) T33(1,5)];

clear i j
for i=1:3
   for j=1:3
      T10(i,j)=(T101(i,j)+T102(i,j)+T103(i,j)+T104(i,j)+T105(i,j))/5;
   end
end
T30=[0 0 1;-1 0 0;0 -1 0];
%%%%%微分定日镜
SSS=0;
for a=1:100
    for b=1:100
        DRJ(a,b)=1;
        P0=[a b 0];
        P=P0*T10*T30;
    if -4<=P(1,2)&&P(1,2)<=4&&P(1,1)>=-3.5&&P(1,1)<=3.5
        DRJ(a,b)=0;
    end
sss=DRJ(a,b)+SSS;
    end
end
JD=1-(sss/10000);

eta_sb_n=eta_sb_n*(sum(eta_sb2(:,2))/5);
%%计算平均效率
for i=1:9
    eta(i,1)=JD*eta_sb1(i,1)*0.92*eta_sb2(1,1)*eta_cos1(i,1)*eta_at(i,1);
end

%%单位面积镜面平均输出热功率
for i=1:5
    DNI(i,1)=1.366*(0.34981+0.5783875*exp(-0.275745/sin(alpha(i,1))));
end
DNI_n=sum(DNI(:,1))/5;
clear DNI;
```

```
eta(1,2)=sum(eta(:,1))/9;
e_n=DNI_n*eta(1,2);
```

附录 2

介绍：由 python 编写，用于模拟退火求解问题二

```
"""导入计算程序所选用的的模块包"""
import math
import random
import numpy as np

"""模拟退火进行无约束分析"""


def fitness_cal(x, num_var):
    """计算适应性函数的值"""
    for i in range(num_var):  # 36 可以更改为更高维度的变量
        y = 36*0.8696*(0.0698*0.7631*(0.99321-
0.0001176*math.sqrt(x[i]**2+x[i]**2+72**2)+
                                                1.92e-
8*(x[i]**2+x[i]**2+72**2))*0.92*0.99)*5/x[i]-60/x[i] # 36 为人为控制的量
    return y


def simulate_sa_algorithm(num_var, x_max, x_min,
                          temperature_begin, temperature_final,
                          low_t_rate, iteration, scale):
    """进行模拟退火操作"""
    rand_num = random.randint(1, 100)  # 随机数选取
    random.seed(rand_num)  # 随机设置种子控制值的生成生成
    x_begin = np.zeros((num_var))
    for v in range(num_var):
        x_begin[v] = random.uniform(x_min[v], x_max[v])  # 随机为初始的
x 值赋值
    y_value = fitness_cal(x_begin, num_var)  # 计算初始的适应性函数值
    """初始化所有后面所需要的值"""
    x_new = np.zeros((num_var))
    x_now = np.zeros((num_var))
    x_best = np.zeros((num_var))
    x_now[:] = x_begin[:]
    y_value_now = y_value
    y_value_best = y_value
    out_iter = 0
    total_iteration = 0
```

```
    iteration = iteration  # 迭代次数
    t_now = temperature_begin
    while t_now >= temperature_final:  # 外部温度循环
        for iter in range(iteration):
            total_iteration += iter  # 内部迭代的时间序列
            x_new[:] = x_now[:]
            v = random.randint(0, num_var-1)
            x_new[v] = x_now[v] + scale*(x_max[v]-
x_min[v])*random.normalvariate(0, 1)
            x_new[v] = max(min(x_new[v], x_now[v]), x_min[v])

            y_value_new = fitness_cal(x_new, num_var)
            less = abs(y_value_new - y_value_now)
            """Metropolibs 采样方法，以用于选取小于值"""
            if y_value_new < y_value_now:  # 以一定的概率选取新产生的值
以跳出局部最优解
                accept = True
            else:
                p = math.exp(less/t_now)
                if p > random.random():
                    accept = True
                else:
                    accept =False
            if accept == True:  # 为新的值进行变换
                x_now[:] = x_new[:]
                y_value_now = y_value_new
                if y_value_new < y_value_best:
                    y_value_best = y_value_new
                    x_best[:] = x_new[:]
        """根据模拟炉内退火进行降温操作"""
        t_now = t_now * low_t_rate
        out_iter += 1
    return out_iter, x_best, y_value_best, y_value_now  # 返回迭代结果


def output_result(num_var,x_best,y_best_value):
    # ====== 优化结果校验与输出 ======
    y_check = fitness_cal(x_best, num_var)
    if abs(y_best_value - y_check)>1e-3:   # 检验目标函数
        print("Error 2: Wrong total millage!")
        return
    else:
        print("\nOptimization by simulated annealing algorithm:")
        for i in range(num_var):
```

```
            print('\tx[{}] = {:.6f}'.format(i,x_best[i]))
        print('\n\tf(x):{:.6f}'.format(y_best_value))
    return


"""变量设定"""
num_var1 = 3  # 若增加变量、维度总群数量需要更改
x_max = [350, 350, 2500]  # 形如金属上界取值有五个自变量
x_min = [-350, -350, 1500]  # 形如金属下界取值
temperature_begin = 100  # 开始的初始温度
temperature_final = 1  # 最终温度
low_t_rate = 0.98  # 降温系数
iteration = 100  # 迭代次数(内循环)
scale = 0.5



def main_prog():
    """主程序启动函数"""
    [out_iter, x_best, y_value_best, y_value_now] \
        = simulate_sa_algorithm(num_var1, x_max, x_min,
temperature_begin, temperature_final, low_t_rate, iteration, scale)

    # 结果校验与输出
    output_result(num_var1, x_best,y_value_best)


if __name__ == '__main__':  # 调试函数
    main_prog()
```

附录 3

介绍：由 matlab 编写，用于粒子群算法求解问题二

```
%%%求解五元目标函数 y = 12*0.8696*x(4)*x(5)*(0.7631*(0.99321-0.0001176*...
% sqrt((x(1))^2+(x(2))^2+72^2)+1.92e-
8*((x(1))^2+(x(2))^2+72^2))*0.92*0.99)*x(3)/5 - 5 的最小值
% x(4)--> 长:2~8m
% x(5)--> 宽:2~8m
% x(1) --> x 坐标
% x(2) --> y 坐标
% x(3) --> 定日镜数目
clear; clc
```

```
%%写出变量的参考范围
n = 40; % 粒子个数
narvs = 5; % 变量范围
c1_first = 2.5; % 个体学习因子初始值
c1_final = 0.5; % 个体学习因子末值
c2_first = 1.0; % 社会学习因子初始值
c2_fianl = 2.25; % 社会学习因子末值
omega_max = 0.9; % 最大惯性权重
omega_min = 0.4; % 最小惯性权重
K = 100; % 迭代次数
vmax = [8, 8, 8, 8, 8]; % 粒子最大的速度
x_lb = [-350, -350, 1500, 2, 2]; % 粒子扰动的下界
x_ub = [350, 350, 2500, 8, 8]; % 粒子扰动的上界
%% 初始化粒子的速度以及位置
x = zeros(n, narvs);
for i = 1:narvs
    x(:,i) = x_lb(i) + (x_ub(i)-x_lb(i))*rand(n, 1); % 随机初始化粒子所在位
置
end
v = -vmax + 2*vmax .* rand(n, narvs); % 随机初始化粒子的速度

%% 计算适应度
fitness = zeros(n, 1);
for i = 1:n % 遍历整个粒子群
    fitness(i) =  y_60(x(i,:));  % 调用适应性函数来计算适应度
end
pbest = x;  % 初始化 n 个粒子的最佳调用位置
index = find(fitness == min(fitness),1); % 寻找适应性最小的下标
gbest = x(index, :); % 定义所有粒子目前最好位置

%% 迭代 K 次来更新速度与位置
fitnessbest = ones(K, 1);
for d = 1:K
    c1 = c1_first-d*(c1_final-c1_first);
    c2 = c2_first-d*(c2_fianl-c2_first);
    for i = 1:n
        f_i = fitness(i);
        f_average = sum(fitness)/n;
        f_min = min(fitness);
        if f_i <= f_average
            if f_average ~= f_min
                omega = omega_min + (omega_max - omega_min)*...
                    (f_i - f_min)/(f_average - f_min);
            else
```

```
                omega = omega_min;
            end
        else
            omega = omega_min;
        end
        v(i, :) = omega*v(i, :) + c1*rand(1)*(pbest(i,:) - x(i, :))+...
            c2*rand(1)*(gbest - x(i, :));
        for j = 1:narvs
            if v(i, j) < -vmax(j)
                v(i, j) = -vmax(j);
            elseif v(i, j) > vmax(j)
                v(i, j) = vmax(j);
            end
        end
        x(i, :) = x(i,:) + v(i, :);
        for j = 1:narvs
            if x(i, j) < x_lb(j)
                x(i,j) = x_lb(j);
            elseif x(i,j) > x_ub(j)
                x(i,j) = x_ub(j);
            end
        end
        fitness(i) = y_60(x(i, :));
        if fitness(i) < y_60(pbest(i, :))
            pbest(i, :) = x(i, :);
        end
        if fitness(i) < y_60(gbest)
            gbest = pbest(i,:);
        end
    end
    fitnessbest(d) = y_60(gbest);
end


%% 输出的判定条件
if gbest(1, 1).^2 + gbest(1, 2) > 10000 % 设置条件为坐标平方和大于 10000 才输
出
    if gbest(1, 3) == 1500 % 设置条件当拥有定日镜数目最小时才输出
        plot(fitnessbest)
        xlabel('迭代次数')
        disp('最佳的位置是');disp(gbest)
        disp('此时的最优解是');disp(y_60(gbest))
    end
end
```

附录 4

介绍：由 matlab 编写，均匀排布定日镜

```
N = 1750; R1 = 100; R2 = 350;
r = unifrnd(1,(R2/R1)^2,N,1);
theta = 2*pi*rand(N,1);
x = R1*sqrt(r).*cos(theta);
y = R1*sqrt(r).*sin(theta);
h1 = ezplot(@(x,y) x.^2+y.^2-R1^2,[-R2,R2,-R2,R2]); set(h1, 'Color',
'k'); hold on
h2 = ezplot(@(x,y) x.^2+y.^2-R2^2,[-R2,R2,-R2,R2]); set(h2, 'Color',
'b');hold on
plot(x,y,'ro');
axis equal
title('Parametric method')
```

附录 5

介绍：问题三的代码

```
%%%求解五元目标函数 y = 12*0.8696*x(4)*x(5)*(0.7631*(0.99321-0.0001176*...
% sqrt((x(1))^2+(x(2))^2+72^2)+1.92e-
8*((x(1))^2+(x(2))^2+72^2))*0.92*0.99)*x(3)/5 - 5 的最小值
% x(4)--> 长:2~8m
% x(5)--> 宽:2~8m
% x(1) --> x 坐标
% x(2) --> y 坐标
% x(3) --> 定日镜数目
clear; clc

%%写出变量的参考范围
n = 40; % 粒子个数
narvs = 6; % 变量范围
c1_first = 2.5; % 个体学习因子初始值
c1_final = 0.5; % 个体学习因子末值
c2_first = 1.0; % 社会学习因子初始值
c2_fianl = 2.25; % 社会学习因子末值
omega_max = 0.9; % 最大惯性权重
omega_min = 0.4; % 最小惯性权重
K = 100; % 迭代次数
vmax = [8, 8, 8, 8, 8, 8]; % 粒子最大的速度
x_lb = [-350, -350, 1500, 2, 2, 2]; % 粒子扰动的下界
x_ub = [350, 350, 2500, 8, 8, 6]; % 粒子扰动的上界
%% 初始化粒子的速度以及位置
```

```
x = zeros(n, narvs);
for i = 1:narvs
    x(:,i) = x_lb(i) + (x_ub(i)-x_lb(i))*rand(n, 1); % 随机初始化粒子所在位置
end
v = -vmax + 2*vmax .* rand(n, narvs); % 随机初始化粒子的速度

%% 计算适应度
fitness = zeros(n, 1);
for i = 1:n % 遍历整个粒子群
    fitness(i) =  y_new_60(x(i,:));  % 调用适应性函数来计算适应度
end
pbest = x;  % 初始化 n 个粒子的最佳调用位置
index = find(fitness == min(fitness),1); % 寻找适应性最小的下标
gbest = x(index, :); % 定义所有粒子目前最好位置

%% 迭代 K 次来更新速度与位置
fitnessbest = ones(K, 1);
for d = 1:K
    c1 = c1_first-d*(c1_final-c1_first); % 个体学习因子随时间而变的值
    c2 = c2_first-d*(c2_fianl-c2_first); % 社会学习因子随时间而变的值
    for i = 1:n
        f_i = fitness(i);
        f_average = sum(fitness)/n;
        f_min = min(fitness);
        if f_i <= f_average
            if f_average ~= f_min
                omega = omega_min + (omega_max - omega_min)*...
                    (f_i - f_min)/(f_average - f_min); % 权重计算
            else
                omega = omega_min;
            end
        else
            omega = omega_min;
        end
        v(i, :) = omega*v(i, :) + c1*rand(1)*(pbest(i,:) - x(i, :))+...
            c2*rand(1)*(gbest - x(i, :)); % 速度计算
        for j = 1:narvs
            if v(i, j) < -vmax(j)
                v(i, j) = -vmax(j);
            elseif v(i, j) > vmax(j)
                v(i, j) = vmax(j);
            end
        end
        x(i, :) = x(i,:) + v(i, :);
```

```
        for j = 1:narvs
            if x(i, j) < x_lb(j)
                x(i,j) = x_lb(j);
            elseif x(i,j) > x_ub(j)
                x(i,j) = x_ub(j);
            end
        end
        fitness(i) = y_60(x(i, :));
        if fitness(i) < y_new_60(pbest(i, :)) % 适应度和个体最优比较
            pbest(i, :) = x(i, :);
        end
        if fitness(i) < y_new_60(gbest) % 适应度和全局最优进行比较
            gbest = pbest(i,:);
        end
    end
    fitnessbest(d) = y_new_60(gbest);
end


%% 输出的判定条件
if gbest(1, 1).^2 + gbest(1, 2).^2 >= 10000 && gbest(1, 1).^2 +...
        gbest(1, 2).^2 <= 350*350% 设置条件为坐标平方和大于 10000 才输出
    if gbest(1, 3) == 1500 % 设置条件当拥有定日镜数目最小时才输出
        plot(fitnessbest, 'r-')
        xlabel('迭代次数')
        disp('最佳的位置:');disp(gbest)
        disp('这时的最优解是:');disp(y_60(gbest))
    end
end
```