**基于输出效率最大的定日镜场的优化问题**

# 摘要

随着新能源发展迅速，以新能源为主的电力系统成为我国实现“碳中和”的主要措施，其中塔式太阳能发电是一项重要的技术。

这道问题要求我们针对于不同的情况对于定日镜场的参数做出最优的配置求解，并且求出此时对应的输出功率等等信息。然而我们在求解这些问题之前我们需要做出一些合理的简化，并且对于一些损耗功率做出合理的估计，从而得到比较切合实际的最优解。本文我们根据不同的要求和主要考虑因素的不同我们做出了层层分析与求解。

针对问题一：我们在假设定日镜的尺寸参数都是固定好的情况下，计算了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。对于附录中的许多条件我们逐步的进行分析，最终算出所有的效率。我们最终求得的结果见表1和表2.

针对问题二：这是一类优化问题，给我们的要求是在定日镜场的额定年功率为60MW不变的前提，并假设所有定日镜的尺寸与安装高度都相同时，自主的设计定日镜场的参数，使得单位镜面面积年平均输出热功率最大。我们利用不同的约束条件，对于要求的目标函数进行了最优化的求解，最终我们求得的结果在三个表和附带的文件之中。

针对问题三：我们在第二问的基础之上，考虑到一个定日镜之间的不同也会给最终的输出功率带来影响，所以在第三问我们考虑到了不同的定日镜的尺寸与安装高度，使得最终得到的结果更可能的接近真是情况下的最大化，如果定日镜尺寸可以不同，安装高度也可以不同，额定功率设置同问题2，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。

关键词：最优化配置、L-BFGS-B算法、python语言

# 问题重述

时代的进步，环保越来越受到人们的重视，新能源技术的发展已经是重中之重。本问题的定日镜场便是其中的一种，我们要根据定日镜不同的参数，来求出此时能够输出的功率，并且能够设计参数来保证求出的输出功率最大化。

问题一：在给定的定日镜场参数的条件下，通过对于实际情况的计算，从而求得所有需要求的效率值。

问题二：在定日镜场额定年平均功率不变的情况下，要求我们改变其他的参数来使得定日镜场的单位镜面面积的年平均输出功率达到最大。

问题三：如果定日镜尺寸可以不同，安装高度也可以不同，额定功率设置同问题2，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。，

# 问题分析

## 问题一的分析

此问定日镜场的问题，我们在定日镜参数给定的情况之下，我们需要根据题目的条件和额外的资料查阅来解决相关的数值计算。我们首先将吸收塔位于定日场的圆心处，定日镜的尺寸均为6m\*6m，安装高度均为4m，然后有附录1开始，从D开始倒推计算，再有时间ST 算得太阳时角， DNI为变量，针对不同时间节点，太阳高度角并非定值，可由上条依次计算，从而可以求得附录1中的所有值，之后在对于所有效率进行逐一的计算即可，最后可以求得年平均光学效率，有了上面的量便可以求得输出热效率。

## 问题二的分析

在建模中，需要定义一个目标函数，作为优化调度的目标，目标函数应该考虑到使单位镜面面积年平均输出热功率最大，已知定日镜的尺寸和安装高度都一致的情况下，考虑定日镜尺寸、安装高度、数目、安装位置，以它们为约束条件，来确立约束方程组，从而得到目标函数在约束条件下的极值。

## 问题三的分析

在第二问的基础之上，考虑到定日镜不一样的实际情况，所有尺寸与安装位置都可以不同，额定功率同问题二一样，让我们重新调整每一个定日镜的参数，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。

# 模型假设

1. **一年中所考虑时间节点天气条件相同，且均有充足光照，即太阳辐射度为定值**
2. **定日镜场内除吸收塔和定日镜，无其它建筑物可能产生阴影遮挡**
3. **定日镜场内地形平坦，场内不存在海拔高度差**
4. **集热器的光学效率稳定**
5. **在列入计算的时间节点，定日镜场内设备都以正常工作状态运转**

# 符号说明

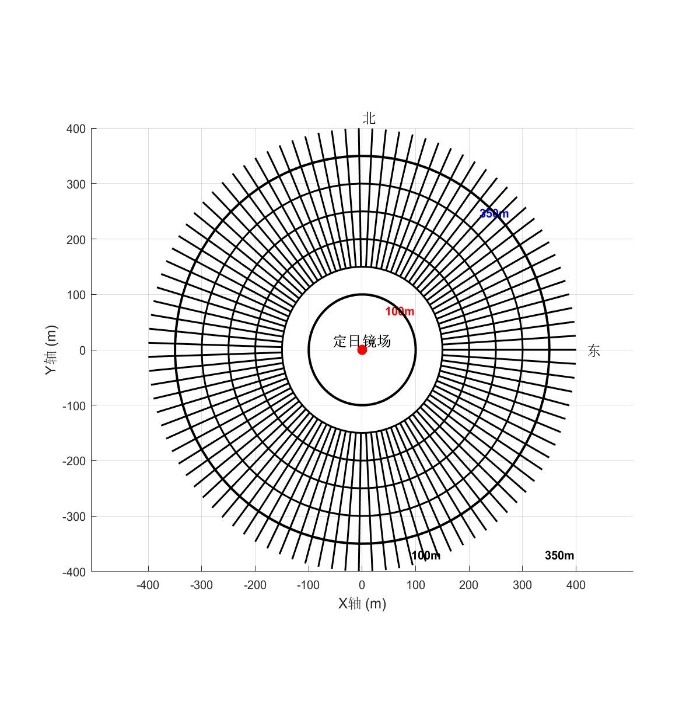
|  |  |  |  |
| --- | --- | --- | --- |
| 年平均输出热功率 | |  | |
| 定日镜尺寸（镜面面积） | |  | |
| 单位镜面面积年平均输出热功率 | |  | |
| 定日镜i位置坐标 | | , | |
| 镜面中心到集热器中心距离 | |  | |
| 安装高度 | |  | |
| 镜面高度 | |  | |
| 镜面宽度 | |  | |
| 太阳赤纬角 | | 𝛿 | |
| 太阳时角 | | 𝜔 | |
| 太阳高度角 |  | | | | |
| 当地纬度 |  | | | | |
| 太阳方位角 |  | | | | |
| 镜面反射率 | 𝜂ref | | | | |
| 余弦效率 | 𝜂cos | | | | |
| 阴影遮挡效率 | 𝜂sb | | | | |
| 集热器截断效率 | 𝜂trunc | | | | |
| 大气透射率 | 𝜂at | | | | |
| 法向直接辐射辐照度 | | | DNI | |
| 定日镜场的输出热功率 | | | 𝐸field | |
| 海拔高度 | | | H | |
| 定日镜光学效率 | | | 𝜂 | |
|  | | |  | |
|  | | |  | |

# 模型的建立与求解

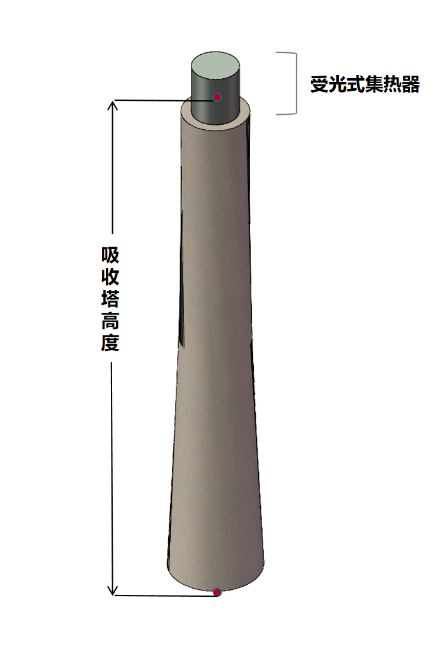
## 问题一的模型建立与求解

首先我们先解释一些概念**，吸收塔高度：集热器中心离地的高度。定日镜的安装高度：纵向转轴与水平转轴的交点离地的高度**。

然后我们画出一些示意图以便于更好的理解这个问题



定日镜场示意图



吸收塔示意图

### 太阳赤纬角的计算

针对题干中附录1的信息我们由D开始倒推：

D为距离春分的天数，这里我们应当理解为绝对值的意思始终取正值，由公式得即可到所求的值的正弦值

|  |  |
| --- | --- |
| sin 𝛿 = sin 2π𝐷 365 sin ( 2π 360 23.45) | （1） |

### 太阳时角的计算

由当地时间ST（9，10.5，12，13.5，15五个值），可以得到太阳时角𝜔，公式如下：

|  |  |
| --- | --- |
| 𝜔 = 𝜋 12 (𝑆𝑇 − 12) | （2） |

### 太阳高度角的计算

题干中已经给出了当地的经度与纬度，为当地纬度，然后在通过前面求得的𝜔与𝛿便可以求出太阳高度角，公式如下：

|  |  |
| --- | --- |
| sin 𝛼 = cos 𝛿 cos𝜑 cos 𝜔 + sin 𝛿 sin | （3） |

### 太阳方位角的计算

同理由上述已经求得的值，通过公式：

|  |  |
| --- | --- |
| 𝛿-sin | (4) |

### 阴影遮挡效率计算

光伏电站的阴影遮挡主要来自于以下几项：组件间阵列遮挡，远景及近景的障碍物遮挡。我们针对此问题做出了一定的简化，考虑俩方面的因素：

1. 定日镜与定日镜之间的遮挡。
2. 吸收塔与定日镜的遮挡。

我们对于每个定日镜遮挡的情况不同，进行了不同的分区，每一个分区采用代表的样本来进行遮挡性问题分析，吸收塔遮光部分应单独分区

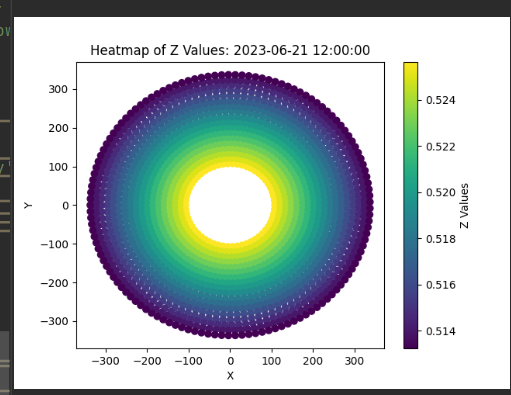
代码的实现：不同月的不同时间9:00、10:30、12:00、13:30、15:00 因为太阳高度角影响影子长度不同，使用遮挡效率函数 def n（solar） 返回0-1。

最终阴影遮挡效率计算：

|  |  |
| --- | --- |
| 𝜂sb = 1 − 阴影遮挡损失， | （5） |

### 余弦效率的计算

余弦效率：由于太阳光向与镜面采光口法线方向不平行引起的接收能量减少。[2]余弦损失与太阳方位角有成函数关系，经过编程得到的数据如下示意图：



余弦损失示意图

### 集热器截断效率的计算

题干中可知镜面反射率 𝜂ref 可取为常数，例如 0.92。然后再考虑到集热器是圆柱结构，故其实际吸收能量我们以有效投影面积和理论接受的太阳光面积。

由公式

|  |  |
| --- | --- |
| 集热器截断效率= 集热器接收能量/镜面全反射能量 − 阴影遮挡损失能量 | (6) |

可得求出集热器接受能量，镜面全反射能量与阴影遮挡损失能量即可，其中

集热器接受能量=0.92\*DNI\*6

阴影遮挡损失能量=阴影遮挡损失\*DNI\*6\*6

镜面全反射=DNI\*6\*6

最终化简上式可得

|  |  |
| --- | --- |
| 𝜂trunc==0.92/(1-阴影遮挡损失能量) | （7） |

### 法向辐射度DNI和输出功率𝐸field的计算

DNI是指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，可按照如下的公式来计算：

|  |  |
| --- | --- |
| DNI = 𝐺[𝑎 + 𝑏 exp (− 𝑐 /sin)], | （8） |
| 𝑎 = 0.4237 − 0.00821(6 − 𝐻) 2 | （9） |
| 𝑏 = 0.5055 + 0.00595(6.5 − 𝐻) 2 | （10） |
| 𝑐 = 0.2711 + 0.01858(2.5 − 𝐻) 2 , | （11） |

其中G为太阳常数，其值取为 1.366 kW/m2。

输出功率我们可以由下面的公式求得:

|  |  |
| --- | --- |
| 𝐸field = DNI ·∑𝐴𝑖𝜂𝑖 𝑁 𝑖 , | (12) |

### 大气透气率计算

我们可以由下面的公式直接算出𝜂at，

|  |  |
| --- | --- |
| 𝜂at = 0.99321 − 0.0001176𝑑HR + 1.97 × 10−8 × 𝑑HR 2 (𝑑HR ≤ 1000)[3] | （13） |

其中dHR为示镜面中心到集热器中心的距离。

### 定日镜的光学效率 𝜂的计算

由公式可得：

|  |  |
| --- | --- |
| 𝜂 = 𝜂sb𝜂cos𝜂at𝜂trunc𝜂ref | （14） |

## 问题二模型建立与求解

此问要求我们以定日镜的额定年平均输出热功率不变60MV为前提，假设所有定日镜尺寸及安装高度相同，来设计定日镜以下的参数：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、使得定日镜再达到额定功率的条件下，单位镜面面积年平均输出功率最大。

### 目标函数的建立

根据本题题意我们以所求的变量为目标函数，构建得到目标函数

|  |  |
| --- | --- |
|  | （15） |

### 总功率约束

由题干可以直接得到：

|  |  |
| --- | --- |
| =60MW | （16） |

### 定日镜范围约束

由题干得可以根据不同的要求得到约束定日镜i距吸收塔的欧式距离大于100公式如下：

|  |  |
| --- | --- |
| 00 | （17） |

### 安装高度、镜面尺寸和触底约束

由题干可得以下的公式

|  |  |
| --- | --- |
|  | （18） |
| ，8 | （19） |
|  | （20） |

### 维护约束

相邻定日镜之间的距离需要满足让清洗车通过的条件：

|  |  |
| --- | --- |
| 5 | （21） |

### 最终的函数模型

由以上的分析可得，在定日镜尺寸安装高度一样的前提下，输出功率的目标函数为：

|  |  |
| --- | --- |
|  | （22） |

其约束条件为：

|  |  |
| --- | --- |
| =60MW |  |
| 00 |  |
|  |  |
| ，8 | (23) |
|  |  |
| 5 |  |

## 问题三模型的建立与求解

此问要求我们更加的切合实际生活，再考虑每一个定日镜可以不同的前提下，额定功率同第二问一样，让我们重新求出能够使单位镜面面积年平均输出热功率尽量大的各个定日镜的参数值。公式与第二问的目标函数和约束条件一样。求得的结果

# 模型的检验

# 模型的评价与改进

# 附录

solarCalculations.py

import datetime  
import pytz  
import pysolar.solar as solar  
import math  
  
def calculate\_solar\_altitude(latitude, longitude, year, month, day, hour, minute, second):  
 # 北京时区  
 beijing\_timezone = pytz.timezone('Asia/Shanghai')  
  
 # 指定特定日期和时间（在北京时区）  
 beijing\_time = beijing\_timezone.localize(datetime.datetime(year, month, day, hour, minute, second))  
  
 # 计算太阳高度角  
 altitude = solar.get\_altitude(latitude, longitude, beijing\_time)  
  
 # 将太阳高度角转化为度数  
 altitude\_deg = round(altitude, 2)  
  
 return altitude\_deg  
  
  
  
  
  
  
  
def calculate\_solar\_azimuth(latitude, longitude, year, month, day, hour, minute, second):  
 # 北京时区  
 beijing\_timezone = pytz.timezone('Asia/Shanghai')  
  
 # 指定特定日期和时间（在北京时区）  
 beijing\_time = beijing\_timezone.localize(datetime.datetime(year, month, day, hour, minute, second))  
  
 # 计算太阳方位角  
 azimuth = solar.get\_azimuth(latitude, longitude, beijing\_time)  
  
 # 将太阳方位角转化为度数  
 azimuth\_deg = round(azimuth, 2)  
  
 return azimuth\_deg  
  
  
  
  
  
  
  
def calculate\_dni( altitude, solar\_altitude):  
 G0 = 1.366 # 太阳常数，kW/m^2  
  
 a = 0.4237 - 0.00821 \* (6 - altitude)\*\*2  
 b = 0.5055 + 0.00595 \* (6.5 - altitude)\*\*2  
 c = 0.2711 + 0.01858 \* (2.5 - altitude)\*\*2  
  
 dni = G0 \* (a + b \* math.exp(-c / math.sin(math.radians(solar\_altitude))))  
  
 return dni  
  
  
  
  
  
  
def calculate\_field\_output\_power(dni, mirror\_areas, optical\_efficiencies):  
 N = len(mirror\_areas)  
  
 if N != len(optical\_efficiencies):  
 raise ValueError("镜面数量和光学效率数量必须相等")  
  
 total\_power = dni \* sum(mirror\_areas[i] \* optical\_efficiencies[i] for i in range(N))  
  
 return total\_power  
  
  
  
  
def calculate\_optical\_efficiency(distance\_to\_receiver,solar\_elevation\_angle):  
 # 阴影遮挡效率，这里假设阴影遮挡损失为0  
 eta\_sb = 1.0  
  
 # 余弦效率，  
 # 计算余弦损失  
 cosine\_loss = math.cos(math.radians(solar\_elevation\_angle))  
 eta\_cos = 1.0 -cosine\_loss  
  
 # 计算大气透射率  
 if distance\_to\_receiver <= 1000:  
 eta\_at = 0.99321 - 0.0001176 \* distance\_to\_receiver + 1.97e-8 \* distance\_to\_receiver \*\* 2  
 else:  
 eta\_at = 0.99321 # 大气透射率的默认值  
  
 # 集热器截断效率，这里假设截断效率为1  
 eta\_trunc = 1.0  
  
 # 镜面反射率  
 eta\_ref = 0.92  
  
 # 计算光学效率  
 eta = eta\_sb \* eta\_cos \* eta\_at \* eta\_trunc \* eta\_ref  
  
 return eta  
  
  
  
  
  
def set\_distance(x,y,h):  
  
 d = math.sqrt(x\*\*2+y\*\*2 + (80-h)\*\*2)  
 return d

question1\_basedata.py

import pandas as pd  
from utils import \*  
from utils.solarCalculations import \*  
  
  
  
# 指定Excel文件路径  
excel\_file\_path = './ms/附件.xlsx' # 替换成你的Excel文件路径  
  
# 使用pandas读取Excel文件  
df = pd.read\_excel(excel\_file\_path)  
  
# 使用iterrows()逐行迭代  
for index, row in df.iterrows():  
 # 访问每一行的数据  
  
 dis = set\_distance(row[0], row[1],4)  
 df.at[index, 'distance'] = dis  
  
  
  
 print(row[0],row[1] , row[2],) # 替换成你的列名  
  
# 保存DataFrame到Excel文件  
df.to\_excel(excel\_file\_path, index=False)

question1\_sol\_time.py

from utils.solarCalculations import \*  
from datetime import datetime, timedelta  
import pandas as pd  
# 设置起始日期和结束日期  
start\_date = datetime(2023, 1, 21) # 起始日期为2023年1月21日  
end\_date = datetime(2023, 12, 21) # 结束日期为2023年12月21日  
  
# 设置时间列表，您可以根据需要修改  
# 定义时间点  
times = ['9:00', '10:30', '12:00', '13:30', '15:00']  
  
# 初始化日期时间列表  
date\_times\_list = []  
  
# 生成日期时间列表  
current\_date = start\_date  
while current\_date <= end\_date:  
 for time in times:  
 date\_time\_str = f"{current\_date.strftime('%Y-%m-%d')} {time}"  
 date\_time = datetime.strptime(date\_time\_str, '%Y-%m-%d %H:%M')  
  
 # 分解日期时间为 [年，月，日，小时，分钟，秒]  
 date\_time\_list = [date\_time.year, date\_time.month, date\_time.day, date\_time.hour, date\_time.minute,  
 date\_time.second]  
  
 date\_times\_list.append(date\_time\_list)  
 # 增加一个月，以便生成下一个月的日期  
 if current\_date.month == 12:  
 current\_date = current\_date.replace(year=current\_date.year + 1, month=1)  
 else:  
 current\_date = current\_date.replace(month=current\_date.month + 1)  
  
# 打印结果列表  
for date\_time in date\_times\_list:  
 # print(date\_time)  
  
 year, month, day, hour, minute, second = date\_time  
 # print(f"Year: {year}, Month: {month}, Day: {day}, Hour: {hour}, Minute: {minute}, Second: {second}")  
 altitude = calculate\_solar\_altitude(39.6, 98.5, year, month, day, hour, minute, second)  
 dni = calculate\_dni(3.0,altitude)  
 date\_time.append(altitude)  
 date\_time.append(dni)  
 print(  
 f"Year: {year}, Month: {month}, Day: {day}, Hour: {hour}, Minute: {minute}, Second: {second}, "  
 f"solar\_altitude: {altitude}, dni: {dni}, ")  
  
  
for date\_time in date\_times\_list:  
 year, month, day, hour, minute, second,altitude,dni = date\_time  
 efflist = []  
 eta\_trunclist = []  
 eta\_sb=[]  
 # 指定Excel文件路径  
 excel\_file\_path = 'ms/附件.xlsx' # 替换成你的Excel文件路径  
  
 # 使用pandas读取Excel文件  
 df = pd.read\_excel(excel\_file\_path)  
 data\_points = []  
 sumdata=0  
 # 使用iterrows()逐行迭代  
 for index, row in df.iterrows():  
 efficiency = calculate\_optical\_efficiency(df.at[index, 'distance'],altitude)  
 df.at[index, 'efficiency'] = efficiency  
 data\_points.append([row[0], row[1],df.at[index, 'efficiency']])  
 efflist.append(efficiency)  
 area = 6\*6  
 sumdata += area \*efficiency  
  
 efficiency\_values = [element[2] for element in data\_points]  
 # 使用内置函数sum()计算总和  
 total = sum(efficiency\_values)  
  
 # 使用len()函数计算数字的数量  
 count = len(efficiency\_values)  
  
 # 计算平均数  
 average = total / count  
 efficiency\_values = []  
  
  
  
 import matplotlib.pyplot as plt  
  
  
  
 x\_values = [point[0] for point in data\_points]  
 y\_values = [point[1] for point in data\_points]  
 z\_values = [point[2] for point in data\_points]  
  
 plt.scatter(x\_values, y\_values, c=z\_values, cmap='viridis')  
 color\_bar = plt.colorbar()  
 color\_bar.set\_label('Z Values')  
 # 创建日期时间对象  
 str\_date\_time = datetime(year, month, day, hour, minute, second)  
  
 # 使用strftime将每个组件转换为字符串并拼接  
 date\_time\_str = str\_date\_time.strftime("%Y-%m-%d %H:%M:%S")  
  
 plt.title('Heatmap of Z Values: '+date\_time\_str)  
 plt.xlabel('X')  
 plt.ylabel('Y')  
 # 显示热力图  
 plt.show()  
  
 field\_output\_power = dni \* sumdata  
 date\_time.append(field\_output\_power)  
 print(  
 f"Year: {year}, Month: {month}, Day: {day}, Hour: {hour}, Minute: {minute}, Second: {second}, "  
 f"solar\_altitude: {altitude}, dni: {dni}, field\_output\_power: {field\_output\_power},average: {average}")  
  
  
  
# 创建DataFrame  
df = pd.DataFrame(date\_times\_list, columns=['Year', 'Month', 'Day', 'Hour', 'Minute', 'Second','altitude','dni','field\_output\_power'])  
  
# 保存DataFrame到Excel文件  
excel\_file = 'ms/date\_times.xlsx'  
df.to\_excel(excel\_file, index=False, engine='openpyxl')  
  
print(f'Data saved to {excel\_file}')

question2.py

import numpy as np  
from scipy.optimize import minimize  
  
# 设置初始设计变量  
N = 10 # 假设定日镜数量为10  
x0 = np.random.rand(N) \* 200 - 100 # 随机初始化x坐标，范围在-100到100之间  
y0 = np.random.rand(N) \* 200 - 100 # 随机初始化y坐标，范围在-100到100之间  
A\_mirror = 4.0 # 假设镜面面积为4平方米  
h\_m = 4.0 # 假设安装高度为4米  
h\_mirror = 4.0 # 假设镜面高度为4米  
w\_mirror = 4.0 # 假设镜面宽度为4米  
  
# 设置参数  
E\_y = 60.0 # 额定功率  
min\_distance = 100.0 # 定日镜范围约束的最小距离  
min\_h\_m, max\_h\_m = 2.0, 6.0 # 安装高度约束范围  
min\_h\_mirror, max\_h\_mirror = 2.0, 8.0 # 镜面高度约束范围  
min\_w\_mirror, max\_w\_mirror = 2.0, 8.0 # 镜面宽度约束范围  
maintenance\_distance = 5.0 # 维护约束的最小距离  
  
# 设置设计变量范围  
bounds = [(1, 100)] \* (2 \* N) + [(A\_mirror, A\_mirror), (h\_m, h\_m), (h\_mirror, h\_mirror), (w\_mirror, w\_mirror)]  
  
# 计算单位镜面面积年平均输出热功率 E\_(y-mirror)  
def calculate\_mirrored\_power(x, y, A\_mirror, h\_m, h\_mirror, w\_mirror):  
 # 在这里编写计算 E\_(y-mirror) 的代码，根据问题描述进行计算  
 # 返回 E\_(y-mirror) 作为标量值  
 # 这只是一个示例，你需要根据实际问题进行计算  
 mirrored\_power = sum(x) # 假设单位镜面的年平均输出热功率是 x + y  
 return -mirrored\_power # 取负号以便最大化  
def calculate\_total\_power(N, A\_mirror):  
 return N \* A\_mirror # 简化为所有镜面面积之和  
  
# 计算定日镜之间的距离  
def calculate\_mirror\_distance(x, y):  
 distances = []  
 for i in range(N):  
 for j in range(i + 1, N):  
 distance = np.sqrt((x[i] - x[j])\*\*2 + (y[i] - y[j])\*\*2)  
 distances.append(distance)  
 return distances  
  
# 定义目标函数，最大化单位镜面面积年平均输出热功率 E\_(y-mirror)  
def objective(variables):  
 x = variables[:N]  
 y = variables[N:2\*N]  
 A\_mirror, h\_m, h\_mirror, w\_mirror = variables[2\*N:]  
 return -calculate\_mirrored\_power(x, y, A\_mirror, h\_m, h\_mirror, w\_mirror)  
  
# 定义约束条件  
def constraint1(variables):  
 x = variables[:N]  
 y = variables[N:2\*N]  
 A\_mirror, h\_m, h\_mirror, w\_mirror = variables[2\*N:]  
 return calculate\_total\_power(N, A\_mirror) - E\_y  
  
def constraint2(variables):  
 x = variables[:N]  
 y = variables[N:2\*N]  
 return min\_distance - min(calculate\_mirror\_distance(x, y))  
  
def constraint3(variables):  
 h\_m = variables[2\*N + 1]  
 return h\_m - min\_h\_m  
  
def constraint4(variables):  
 h\_m = variables[2\*N + 1]  
 return max\_h\_m - h\_m  
  
def constraint5(variables):  
 h\_mirror = variables[2\*N + 2]  
 return h\_mirror - min\_h\_mirror  
  
def constraint6(variables):  
 h\_mirror = variables[2\*N + 2]  
 return max\_h\_mirror - h\_mirror  
  
def constraint7(variables):  
 w\_mirror = variables[2\*N + 3]  
 return w\_mirror - min\_w\_mirror  
  
def constraint8(variables):  
 w\_mirror = variables[2\*N + 3]  
 return max\_w\_mirror - w\_mirror  
  
def constraint9(variables):  
 x = variables[:N]  
 y = variables[N:2\*N]  
 return maintenance\_distance - min(calculate\_mirror\_distance(x, y))  
  
# 调用优化器进行优化  
variables = list(x0) + list(y0) + [A\_mirror, h\_m, h\_mirror, w\_mirror]  
constraints = [  
 {'type': 'eq', 'fun': constraint1},  
 {'type': 'ineq', 'fun': constraint2},  
 {'type': 'ineq', 'fun': constraint3},  
 {'type': 'ineq', 'fun': constraint4},  
 {'type': 'ineq', 'fun': constraint5},  
 {'type': 'ineq', 'fun': constraint6},  
 {'type': 'ineq', 'fun': constraint7},  
 {'type': 'ineq', 'fun': constraint8},  
 {'type': 'ineq', 'fun': constraint9}  
]  
  
# 计算定日镜的宽度和高度  
def calculate\_mirror\_size(x, y, h\_m, h\_mirror, w\_mirror):  
 # 在这里编写计算定日镜宽度和高度的代码，根据问题描述进行计算  
 # 返回定日镜宽度和高度作为标量值  
 # 这只是一个示例，你需要根据实际问题进行计算  
 mirror\_width = w\_mirror  
 mirror\_height = h\_mirror  
 return mirror\_width, mirror\_height  
  
  
result = minimize(objective, variables, bounds=bounds, constraints=constraints, method='SLSQP')  
  
# 输出结果  
print("最优解：", result.x)  
print("最大单位镜面面积年平均输出热功率：", -result.fun)  
  
  
# 输出吸收塔和定日镜的坐标  
x\_absorber = result.x[:N]  
y\_absorber = result.x[N:2\*N]  
x\_mirror = result.x[2\*N + 4-1] # 假设这是吸收塔和定日镜的x坐标  
y\_mirror = result.x[2\*N + 4-1] # 假设这是吸收塔和定日镜的y坐标  
print("吸收塔x坐标 (m):", x\_absorber)  
print("吸收塔y坐标 (m):", y\_absorber)  
  
# 计算并输出定日镜的宽度和高度  
mirror\_width, mirror\_height = calculate\_mirror\_size(x\_mirror, y\_mirror, h\_m, h\_mirror, w\_mirror)  
print("定日镜宽度 (m):", mirror\_width)  
print("定日镜高度 (m):", mirror\_height)

│ question1\_basedata.py

│ question1\_sol\_time.py

│ question2.py

│ README.md

│

├─ms

│ A题.pdf

│ date\_times.xlsx

│ result2.xlsx

│ result3.xlsx

│ 数模国赛(1).docx

│ 附件.xlsx

│

└─utils

│ solarCalculations.py

# 参考文献

[1]佚名. "塔式太阳能镜场阴影与遮挡效率的改进算法." 太阳能学报 37.8(2016):6.

[2] 舟丹. 太阳能光热发电[J]. 建筑玻璃与工业玻璃, 2012(11):42-46.

[3] 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.

[4]