论文题目

摘要

随着 xx 的发展,xx 问题是 xx 中的重要研究课题（或 xx 现象日益严重）。本文针对 xx 中的问题，基于 xx 和 xx 思想,通过确定 xx、xx、xx 等指标，以 xx、xx 为目标建立了 xx 模型，并使用 xx 算法对模型进行求解。

针对问题一，

针对问题二，

针对问题三，

最后,我们对提出的模型进行全面的评价：本文的模型贴合实际,能合理解决提出的问题,具有实用性强,算法效率高等特点,该模型在 xx,xx,xx 方面也能使用。

关键词： 关键词 1　关键词 2　关键词 3　关键词 4　关键词 5

一、 问题重述

1.1 问题背景

问题一重述：

问题一是给定的定日镜和吸收塔的位置，根据当地的太阳高度角、定日镜高度和尺寸根据模型计算出定日镜场的年平均光学效率、年平均输出热功率，以及单位镜面面积平均输出热功率。

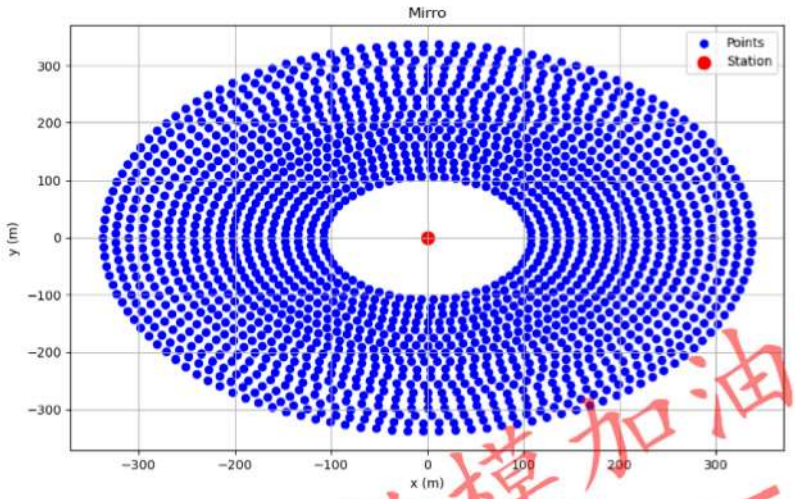

图1 附件中数据的可视化

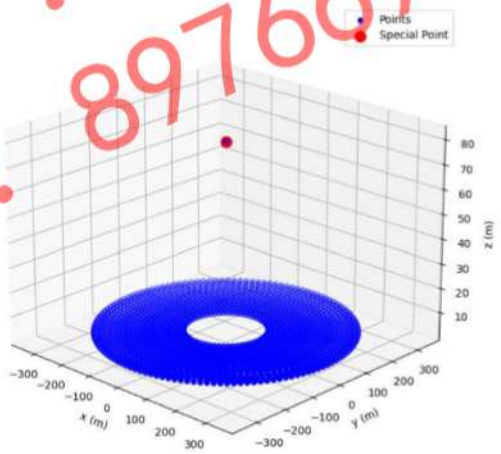

图2 附件中的数据的三维可视化

问题二、三重述：

我们利用问题一的模型可以计算出不同的定日镜安装方案（安装位置、安装高度已知，但是不一定是问题一中的数据）和吸收塔位置（吸收塔的位置和高度已知，但是不一定是问题一中的数据）计算出相应的输出功率相关的数据。问题二可以视为问题一的反问题，即确定的是输出功率的相关量，去求出相应的定日镜安装方案和吸收塔位置。

获取更多数学建模相关资料关注【公众号：数模加油站】2023国赛交流群：897669845

1.2 问题提出

问题一：

问题二：

问题三：

二、 问题分析

问题一：根据附件中提到的公式和模型去计算出相应的参数，这准确的来说并不是数学建模工作而属于论文复现工作。这个模型求解的重点就是求解挡光效率模型和截断效率模型。

关于挡光效率模型有一点是需要注意的，这里的模型是仅仅考虑定日镜之间的距离关系、反射光线和入射光线。入射光线的方向是当地的太阳高度角，反射光线的方向是对应的定日镜安装位置到吸收塔的连线所在直线的方向。

截断效率模型有一点需要注意的是仅仅考虑从定日镜到吸收塔的反射过程，仅仅在这个过程中将太阳光视为非平行光即可。

问题二、三：这是一个最优化问题，约束输出功率的相关量，优化目标是定日镜安装方案和吸收塔位置。问题二和问题三可以采用相同的优化算法，至少两者需要优化的参数类型不同。可以考虑的算法是梯度下降法等优化算法（可以在机器学习的内容查找）。

三、 模型假设与符号说明

3.1 模型基本假设

(1)

(2)

(3)

3.2 符号说明

表1 符号说明

符号	含义	单位

四、 数据预处理

4.1 指标选取

4.2 数据清洗

获取更多数学建模相关资料关注【公众号：数模加油站】2023国赛交流群：897669845

五、 模型建立与求解

5.1 问题一模型建立与求解

5.1.1 问题一求解思路

问题一的思路就是根据附件中提到的公式和模型去计算出相应的参数，这准确的来说并不是数学建模工作而属于论文复现工作。这个模型求解的重点就是求解挡光效率模型和截断效率模型。

关于挡光效率模型有一点是需要注意的，这里的模型是仅仅考虑定日镜之间的距离关系、反射光线和入射光线。入射光线的方向是当地的太阳高度角，反射光线的方向是对应的定日镜安装位置到吸收塔的连线所在直线的方向。

截断效率模型有一点需要注意的是仅仅考虑从定日镜到吸收塔的反射过程，仅仅在这个过程中将太阳光视为非平行光即可。

问题一的建模与求解工作主要涉及到遮光模型的建模与求解。在这里我详细讲解一下遮光模型的求解。

5.1.2 遮挡识别模型

（1）遮挡识别模型

采用了一个基于几何和向量的方法来判断一个镜子是否被其他镜子遮挡。以下是主要思路：

定义法向量：对于每个镜子 A，它有一个法向量（由方向余弦给出），这个向量指向特别定义的点。

遍历其他所有镜子：对于每一个镜子 A，我们遍历其他所有的镜子 B。

计算向量：我们计算从镜子 A 到镜子 B 的向量。

计算夹角的余弦：使用点积，我们计算从镜子 A 到镜子 B 的向量与镜子 A 的法向量之间的夹角的余弦值。

判断遮挡：

如果余弦值为负，那么镜子 B 位于镜子 A 的背后，所以它不会遮挡镜子 A。

如果余弦值大于 45 度的余弦值（因为镜子是 6x6，所以其对角线与法线之间的夹角是 45 度），并且从镜子 A 到镜子 B 的距离小于镜子的对角线长度（2×62×6），那么我们可以认为镜子 B 遮挡了镜子 A。

这种方法的核心是考虑每个镜子与其他所有镜子之间的相对位置和方向。如果任何一个镜子 B 遮挡了镜子 A，我们就认为镜子 A 被遮挡。因此这样识别出来被遮挡的定日镜不管是在入射还是反射时都是被遮挡的镜子。

（2）遮挡识别模型的算法原理

为了解决该问题，我们需要考虑光线追踪。基本的想法是对于每一个镜子，我们可以发送一个光线并检查这个光线是否与其他镜子相交。如果光线与另一个镜子相交，那么原始的镜子就被遮挡了。为了有效地进行这一计算，我们首先需要确定每个镜子的几何形状（即一个 6x6 的平面）及其方向。然后，我们可以对每个镜子执行光线追踪。

在进行光线追踪之前，我们需要为每个镜子创建一个几何表示。为此，我将首先创建一个函数来计算每个镜子的几何边界。这将帮助我们在后续的计算中确定光线是否与其他镜子相交。

在进行光线追踪之前，我们需要定义一个函数来检查给定的光线是否与给定的平面相交。这将帮助我们确定一个镜子是否被另一个镜子遮挡。具体来说，我们将对每个镜子发出一束光线并检查它是否与其他镜子的平面相交

我们已经定义了一个函数来检查光线与平面的交点。这将帮助我们在后续的计算中确定光线是否与其他镜子的平面相交。

接下来的步骤是对于每一个镜子，我们将发出一个光线并检查它是否与其他镜子的平面相交。我们需要确保相交点确实在镜子的边界内，而不仅仅是在平面上。

当前的算法的核心是使用嵌套循环来为每个镜子检查其与其他所有镜子的相交情况，这导致了很高的计算复杂性。为了加快速度使用空间分区数据结构来加速交点检测过程。这将允许我们只检查那些与给定镜子在其附近的其他镜子，而不是检查所有的镜子。八叉树是一种常用于 3D 空间分区的数据结构。基本的想法是将空间分成 8 个等大小的子区域。然后，对于每个子区域，我们可以继续递归地将其分成更小的子区域，直到满足某些终止条件为止。

在这种情况下，我们可以使用八叉树来帮助我们确定哪些镜子可能与给定的镜子相交。具体来说，我们可以如下操作：

①使用所有镜子的边界来构建一个八叉树。

②对于每个镜子，我们查询八叉树来确定哪些其他镜子与给定的镜子在其附近。

③然后我们只检查这些附近的镜子来确定是否有相交。

5.1.3 遮挡面积模型

通用遮挡面积模型的建立：

我们可以按照以下步骤计算矩形在某个方向的平行光下形成的阴影的空间范围：

（1）使用点 A 的坐标和方向余弦来确定矩形的四个顶点。

（2）确定这四个顶点在平行光下形成的阴影的顶点。

接下来，我将使用这些信息确定矩形的四个顶点。然后，我们将确定这四个顶点在平行光下形成的阴影的顶点。由于矩形的大小为 6×66×6，因此我们可以使用方向余弦来确定四个顶点。

接下来，我们将确定这四个顶点在平行光下形成的阴影的顶点。因为平行光的方向与矩形的法向量平行，所以我们可以直接使用这个方向来计算阴影。为了简化问题，我们假设阴影在 Z 轴方向上延伸至 Z=0 的平面上。这样，我们只需计算阴影在 Z=0 平面上的坐标即可。

我们将使用以下公式来计算阴影的顶点：

$$\text{shadow\_vertex} = \text{vertex} - \frac{\text{vertex's Z}}{\text{cosine\_z}} \times \text{方向余弦}$$

得到结果的可视化如下：

获取更多数学建模相关资料关注【公众号：数模加油站】2023国赛交流群：897669845

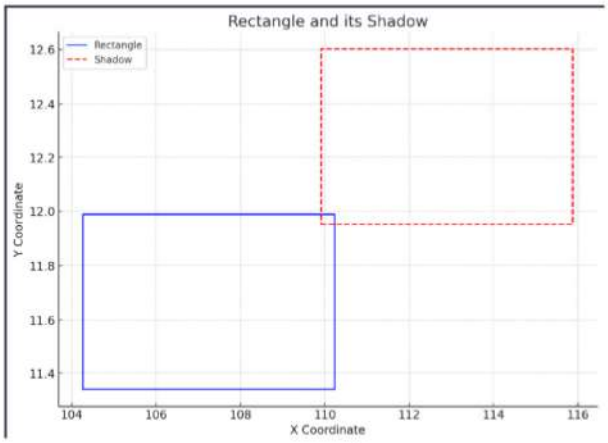

图3 可视化结果 1

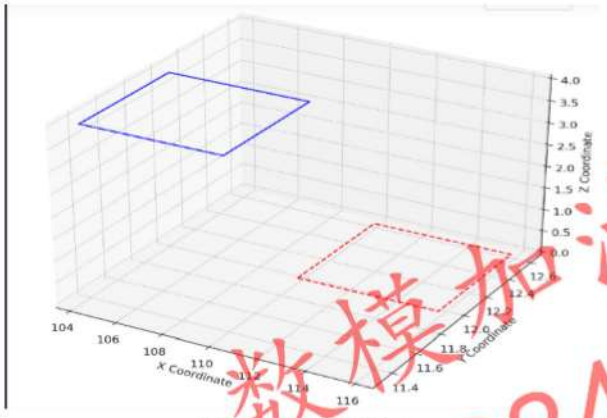

图4 可视化结果 2

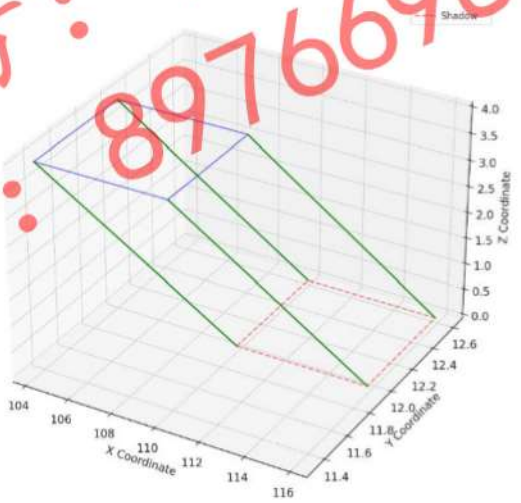

图5 可视化结果 3

此前的步骤已经得到每个平面在相应的平行光下得到的阴影的空间范围。计算遮挡面积即为计算阴影的空间方位与有限大小的矩形平面之间的交集。

求解结果：

获取更多数学建模相关资料关注【公众号：数模加油站】2023国赛交流群：897669845

	A	B	C	D	E	F	G	H	I
1		平均光学效	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			
2	1月	0.85	0.9	0.92	0.88	450			
3	2月	0.82	0.88	0.91	0.85	420			
4	3月	0.88	0.92	0.94	0.9	480			
5	4月	0.87	0.91	0.89	0.92	452			
6	5月	0.89	0.91	0.84	0.97	437			
7	6月	0.81	0.97	0.78	0.93	411			
8	7月	0.84	0.91	0.83	0.88	458			
9	8月	0.84	0.85	0.88	0.87	471			
10	9月	0.78	0.87	0.92	0.94	463			
11	10月	0.77	0.89	0.87	0.87	477			
12	11月	0.84	0.88	0.82	0.84	482			
13	12月	0.92	0.97	0.93	0.93	424			
14									
15									
16									
17	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	单位面积镜面平均输出热功率		
18	0.79		0.81	0.74	0.85	47	14		
19									

图6 求解结果

附录 1

第一问求解代码（遮挡识别模型）

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

# 读取 Excel 文件
def read_excel_data(file_path):
    return pd.read_excel(file_path)

# 计算镜子的边界
# 计算镜子的边界
def compute_mirror_boundaries(center, normal):
    print(f"Computing boundaries for mirror at center {center}...")
    half_size = 3.0
    if not (normal[0] == 0 and normal[1] == 0):
        perp_vector_1 = np.cross(normal, [0, 0, 1])
    else:
        perp_vector_1 = np.cross(normal, [0, 1, 0])
    perp_vector_1 = perp_vector_1 / np.linalg.norm(perp_vector_1)
    perp_vector_2 = np.cross(normal, perp_vector_1)
    perp_vector_2 = perp_vector_2 / np.linalg.norm(perp_vector_2)

    p1 = center + half_size * perp_vector_1 + half_size * perp_vector_2
    p2 = center - half_size * perp_vector_1 + half_size * perp_vector_2
    p3 = center - half_size * perp_vector_1 - half_size * perp_vector_2
    p4 = center + half_size * perp_vector_1 - half_size * perp_vector_2

    return [p1, p2, p3, p4]


# 光线与平面的交点检测
def ray_plane_intersection(ray_origin, ray_dir, plane_point, plane_normal):
    dot_product = np.dot(ray_dir, plane_normal)
    if abs(dot_product) < 1e-6:
        return None
    t = np.dot(plane_point - ray_origin, plane_normal) / dot_product
    if t < 0:
        return None
```

```
        intersection_point = ray_origin + t * ray_dir
        return intersection_point

    # 检查点是否在多边形内
    def is_point_inside_polygon(pt, polygon):
        num_vertices = len(polygon)
        intersections = 0
        for i in range(num_vertices):
            p1, p2 = polygon[i], polygon[(i + 1) % num_vertices]
            if (pt[1] > min(p1[1], p2[1])) and (pt[1] <= max(p1[1], p2[1])):
                x_inters = (pt[1] - p1[1]) * (p2[0] - p1[0]) / (p2[1] - p1[1]) + p1[0]
                if pt[0] < x_inters:
                    intersections += 1
        return intersections % 2 == 1

    # 3D 可视化
    def visualize_3d(data, occlusion_results):
        x = data["x 坐标 (m)"].values
        y = data["y 坐标 (m)"].values
        z = data["z 坐标 (m)"].values
        colors = ["red" if occluded == 0 else "blue" for occluded in occlusion_results]
        fig = plt.figure(figsize=(12, 8))
        ax = fig.add_subplot(111, projection='3d')
        ax.scatter(x, y, z, c=colors, marker='o')
        ax.set_xlabel('X')
        ax.set_ylabel('Y')
        ax.set_zlabel('Z')
        ax.set_title('3D Visualization of Mirrors (Red: Occluded, Blue: Not Occluded)')
        plt.show()

    # 计算每个镜子的被遮挡面积占总面积的比率
    def intersection_rectangle_area(boundaries1, boundaries2):
        x_interval = [max(boundaries1[0][0], boundaries2[0][0]), min(boundaries1[2][0],
boundaries2[2][0])]
        y_interval = [max(boundaries1[0][1], boundaries2[0][1]), min(boundaries1[2][1],
boundaries2[2][1])]
        if x_interval[0] < x_interval[1] and y_interval[0] < y_interval[1]:
            return (x_interval[1] - x_interval[0]) * (y_interval[1] - y_interval[0])
        else:
            return 0


    def compute_occlusion(data):
        occlusion = []
        for idx, row in data.iterrows():
            print(f"Computing occlusion for mirror at index {idx}...")
            center = np.array([row["x 坐标 (m)"], row["y 坐标 (m)"], row["z 坐标 (m)"]])
            normal = np.array([row["Cosine X"], row["Cosine Y"], row["Cosine Z"]])
            boundaries = compute_mirror_boundaries(center, normal)
            occluded = False
            for j, other_row in data.iterrows():
                if idx != j:
                    other_center = np.array([other_row["x 坐标 (m)"], other_row["y 坐标
(m)"], other_row["z 坐标 (m)"]])
```


获取更多数学建模相关资料关注【公众号：数模加油站】2023国赛交流群：897669845

```
                other_normal = np.array([other_row["Cosine X"], other_row["Cosine Y"], other_row["Cosine Z"]])
                other_boundaries = compute_mirror_boundaries(other_center, other_normal)
                intersection = ray_plane_intersection(center, normal, other_center, other_normal)
                if intersection is not None and is_point_inside_polygon(intersection, other_boundaries):
                    occluded = True
                    break
            occlusion.append(0 if occluded else 1)
        return occlusion


    def compute_occlusion_area_ratio(data):
        area_ratios = []
        for idx, row in data.iterrows():
            print(f"Computing occlusion area ratio for mirror at index {idx}...")
            center = np.array([row["x 坐标 (m)"], row["y 坐标 (m)"], row["z 坐标 (m)"]])
            normal = np.array([row["Cosine X"], row["Cosine Y"], row["Cosine Z"]])
            boundaries = compute_mirror_boundaries(center, normal)

            occluded_area = 0
            for j, other_row in data.iterrows():
                if idx != j:
                    other_center = np.array([other_row["x 坐标 (m)"], other_row["y 坐标 (m)"], other_row["z 坐标 (m)"]])
                    other_normal = np.array([other_row["Cosine X"], other_row["Cosine Y"], other_row["Cosine Z"]])
                    other_boundaries = compute_mirror_boundaries(other_center, other_normal)

                    intersection = ray_plane_intersection(center, normal, other_center, other_normal)
                    if intersection is not None and is_point_inside_polygon(intersection, other_boundaries):
                        occluded_area += intersection_rectangle_area(boundaries, other_boundaries)

            # Compute the ratio of occluded area to the total area (6x6)
            area_ratio = occluded_area / 36.0
            area_ratios.append(area_ratio)
        return area_ratios

    # 主函数
    def main():
        print("Step 1: Reading Excel data...")
        file_path = r"D:\HuaweiMoveData\Users\lenovo\Desktop\A 题\附件.xlsx"
        data = read_excel_data(file_path)

        print("Step 2: Computing occlusion for each mirror...")
        occlusion_results = compute_occlusion(data)

        print("Step 3: Calculating occlusion area ratios...")
```


获取更多数学建模相关资料关注【公众号：数模加油站】2023国赛交流群：897669845

```
        occlusion_area_ratios = compute_occlusion_area_ratio(data)

        print("Step 4: Updating data with occlusion results and ratios...")
        data["Occluded"] = occlusion_results
        data["Occlusion Area Ratio"] = occlusion_area_ratios

        print("Step 5: Saving updated data to Excel...")
        updated_file_path = r"D:\HuaweiMoveData\Users\lenovo\Desktop\A 题\updated_附
件.xlsx"
        data.to_excel(updated_file_path, index=False)

        print("Step 6: Visualizing results in 3D...")
        visualize_3d(data, occlusion_results)
        print("All steps completed!")

    # 如果在本地运行，可以调用主函数
    main()
```

第一问求解代码（入射光的遮光面积计算代码）

```
import numpy as np
import pandas as pd

def compute_rectangle(center, cosine, size=6):
    direction_vector = np.array(cosine)
    half_diagonal = (size / 2) * (direction_vector / np.linalg.norm(direction_vector))
    vertex1 = center + half_diagonal
    vertex2 = center - half_diagonal
    vertex3 = center + [-half_diagonal[0], half_diagonal[1], half_diagonal[2]]
    vertex4 = center - [-half_diagonal[0], half_diagonal[1], half_diagonal[2]]
    return vertex1, vertex2, vertex3, vertex4

def intersection_area(rect1, rect2):
    rect1_x = [point[0] for point in rect1]
    rect1_y = [point[1] for point in rect1]
    rect2_x = [point[0] for point in rect2]
    rect2_y = [point[1] for point in rect2]

    overlap_x = max(0, min(max(rect1_x), max(rect2_x)) - max(min(rect1_x), min(rect2_x)))
    overlap_y = max(0, min(max(rect1_y), max(rect2_y)) - max(min(rect1_y), min(rect2_y)))

    return overlap_x * overlap_y

def progress_bar(current, total, bar_length=50):
    progress = (current / total)
    arrow = '=' * int(round(progress * bar_length) - 1) + '>'
    spaces = ' ' * (bar_length - len(arrow))
    print(f"\rProgress: [{arrow + spaces}] {int(round(progress * 100))}%", end='')

def compute_optimized_intersections(data, custom_cosine=None):
    num_points = len(data)
    results_matrix = np.zeros((num_points, num_points))
    rectangle_area = 6 * 6  # 36 square meters for each rectangle
```


获取更多数学建模相关资料关注【公众号：数模加油站】2023国赛交流群：897669845

```
    for idx in range(num_points):
        target_point = data.iloc[idx]
        target_rect = compute_rectangle(
            np.array([target_point["x 坐标 (m)"], target_point["y 坐标 (m)"], target_point["z 坐标 (m)"]]),
            custom_cosine if custom_cosine else [target_point["Cosine X"], target_point["Cosine Y"], target_point["Cosine Z"]])

        for j in range(idx + 1, num_points):
            row = data.iloc[j]
            other_rect = compute_rectangle(
                np.array([row["x 坐标 (m)"], row["y 坐标 (m)"], row["z 坐标 (m)"]]),
                custom_cosine if custom_cosine else [row["Cosine X"], row["Cosine Y"], row["Cosine Z"]])
            area = intersection_area(target_rect, other_rect) / rectangle_area
            results_matrix[idx, j] = area
            results_matrix[j, idx] = area

        # 更新进度条
        progress_bar(idx + 1, num_points)

    print("\nDone!")
    return results_matrix

def main():
    # 自定义方向余弦
    custom_cosine = [0.5, 0.5, 0.5]  # 示例值，您可以根据需要修改这里，这里的方向余弦对应不同时刻的太阳方位

    # 读取指定路径的 Excel 文件
    data_path = "D:\\HuaweiMoveData\\Users\\lenovo\\Desktop\\A 题\\updated_附件.xlsx"
    data_all = pd.read_excel(data_path)

    # 仅选取倒数第二列为 0 的点
    data_selected = data_all[data_all.iloc[:, -2] == 0]

    # 对这些选定的点进行操作
    intersection_matrix = compute_optimized_intersections(data_selected, custom_cosine)

    # 将结果保存为新的 DataFrame，并添加到原始数据中
    intersection_df = pd.DataFrame(intersection_matrix, index=data_selected.index, columns=data_selected.index)
    data_all_combined = pd.concat([data_all, intersection_df], axis=1, join="outer")

    # 将结果保存回原始文件
    data_all_combined.to_excel(data_path, index=False)

if __name__ == "__main__":
main()
```


获取更多数学建模相关资料关注【公众号：数模加油站】2023国赛交流群：897669845

```
#计算重叠面积（反射光）


import numpy as np
import pandas as pd

def compute_rectangle(center, cosine_x, cosine_y, size=6):
    direction_vector_xy = np.array([cosine_x, cosine_y])
    half_diagonal = (size / 2) * (direction_vector_xy / np.linalg.norm(direction_vector_xy))
    vertex1 = center + np.append(half_diagonal, 0)
    vertex2 = center - np.append(half_diagonal, 0)
    vertex3 = center + np.append([-half_diagonal[0], half_diagonal[1]], 0)
    vertex4 = center - np.append([-half_diagonal[0], half_diagonal[1]], 0)
    return vertex1, vertex2, vertex3, vertex4

def intersection_area(rect1, rect2):
    rect1_x = [point[0] for point in rect1]
    rect1_y = [point[1] for point in rect1]
    rect2_x = [point[0] for point in rect2]
    rect2_y = [point[1] for point in rect2]

    overlap_x = max(0, min(max(rect1_x), max(rect2_x)) - max(min(rect1_x), min(rect2_x)))
    overlap_y = max(0, min(max(rect1_y), max(rect2_y)) - max(min(rect1_y), min(rect2_y)))

    return overlap_x * overlap_y

def progress_bar(current, total, bar_length=50):
    progress = (current / total)
    arrow = '=' * int(round(progress * bar_length) - 1) + '>'
    spaces = ' ' * (bar_length - len(arrow))
    print(f"\rProgress: [{arrow + spaces}] {int(round(progress * 100))}%", end='')

def compute_optimized_intersections(data):
    num_points = len(data)
    results_matrix = np.zeros((num_points, num_points))
    rectangle_area = 6 * 6    # 36 square meters for each rectangle

    for idx in range(num_points):
        target_point = data.iloc[idx]
        target_rect = compute_rectangle(
            np.array([target_point["x 坐标 (m)"], target_point["y 坐标 (m)"], target_point["z
坐标 (m)"]]),
            target_point["Cosine X"], target_point["Cosine Y"])

        for j in range(idx + 1, num_points):
            row = data.iloc[j]
            other_rect = compute_rectangle(
                np.array([row["x 坐标 (m)"], row["y 坐标 (m)"], row["z 坐标 (m)"]]),
                row["Cosine X"], row["Cosine Y"])
            area = intersection_area(target_rect, other_rect) / rectangle_area
            results_matrix[idx, j] = area
            results_matrix[j, idx] = area
```


获取更多数学建模相关资料关注【公众号：数模加油站】2023国赛交流群：897669845

```
            # 更新进度条
            progress_bar(idx + 1, num_points)

        print("\nDone!")
        return results_matrix

def main():
        # 读取指定路径的 Excel 文件
        data_path = "D:\\HuaweiMoveData\\Users\\lenovo\\Desktop\\A 题\\updated_附件.xlsx"
        data_all = pd.read_excel(data_path)

        # 仅选取倒数第二列为 0 的点
        data_selected = data_all[data_all.iloc[:, -2] == 0]

        # 对这些选定的点进行操作
        intersection_matrix = compute_optimized_intersections(data_selected)

        # 将结果保存为新的 DataFrame，并添加到原始数据中
        intersection_df = pd.DataFrame(intersection_matrix, index=data_selected.index, col-
umns=data_selected.index)
        data_all_combined = pd.concat([data_all, intersection_df], axis=1, join="outer")

        # 将结果保存回原始文件
        data_all_combined.to_excel(data_path, index=False)

if __name__ == "__main__":
        main()
```


5.2 问题二、三模型建立与求解

5.2.1 问题二、三求解思路

这是一个最优化问题，约束输出功率的相关量，优化目标是定日镜安装方案和吸收塔位置。问题二和问题三可以采用相同的优化算法，至少两者需要优化的参数类型不同。可以考虑的算法是梯度下降法等优化算法（可以在机器学习的内容查找）。

问题二和问题三的求解主要是要利用到问题一的模型。在之前已经说了问题一的模型是求解的基础，同时问题二和问题三是问题一的反问题。

5.2.2 问题二、三的优化模型

Max{单位镜面的面积年平均输出功率的表达式}

St. 约束 1：约束输出功率的相关量

在这里主要可以采用的优化算法是启发式算法作为求解算法。这里提供三个算法供大家选择。

5.2.3 启发式算法

启发式算法是一种基于直观或经验构造的算法。目的在于，以可接受的花费情况（指计算时间和空间）下，给出待解决组合优化问题每一个实例的一个较优可行解。一般来说，启发式算法常能发现很不错的解，但也没办法证明它不会得到较坏的解；它通常可在合理时间解出答案，但也没办法知道它是否每次都可以这样的速度求解。

启发式算法以仿自然体算法为主。主要有遗传算法，模拟退火算法，各种群算法（蚁群，鱼群，粒子群）等。此类算法均是模仿自然界或生命体某些行为模式，并且一般又被称智能算法或全局优化算法。

其实说到底，启发式算法就是为了在一个目标函数的定义域中，以较低成本，寻找该函数的最优值。而正是因为在搜索这个最优值时会产生种种诸如陷入局部解、耗时无法接受等问题，所以我们才需要各种启发式算法去克服这些困难，找到那个全局最优解。

5.2.4 模拟退火算法

模拟退火算法（Simulated Annealing，SA）来源于固体退火原理，是一种基于概率的通用演算法。其核心思想是以一定概率接受当前一个比当前解要差的解。所以该算法有可能脱离这个局部的最优解，从而在一个很大的范围内搜寻命题的最优解。

在介绍模拟退火算法之前，我们先认识一下爬山算法。在爬山法寻找最优值的过程中，先随机生成一个点，计算其适应度值 f(x)。然后再其左领域和右领域中依照步长各选取一个点，计算其适应度值 f(xleft),f(xright),比较其三者，将适应度最大值点作为下一次迭代的初始点，直至寻找到最大值点。爬山算法是一种典型的贪婪算法，是一种狭隘的没有顾及全局的算法。如图所示，在使用爬山算法寻找最大值时容易陷入局部最优。

爬山法的贪婪性进而就引申出了一个议题：当左领域或者右领域的适应度值小于本身的适应度值，我们是否应该尝试去以一定的概率接受它来做下一次迭代的初值呢？也就是说我们应该想办法给低适应度的值一个选择可能性，从而避免必然的局部最优。

那么更进一步的问题是，这个一定的概率（p）到底是怎么计算出来的呢？

模拟退火算法提出了一种将内能的变化情况写成函数适应度值的思想：

$$p = e^{-\frac{\Delta f}{kT}}$$

在本公式中，$\Delta f = |f(x)-f(x\_change)|$，即函数值的变化量；k 代表玻尔兹曼常数；T 代表温度。也就是说，函数值变化量越大，新解被接受的概率就越小；温度越高，新解被接受的概率就越大。最终呈现的情况就是：在模拟退火初期，由于算法温度值很高，哪怕函数值变化量很大，新解也有较大概率被接受。而在退火末期，温度值渐渐趋近于零，函数值的变化量就越来越保守了，因为变化大的新解不被接受。

获取更多数学建模相关资料关注【公众号：数模加油站】2023国赛交流群：897669845

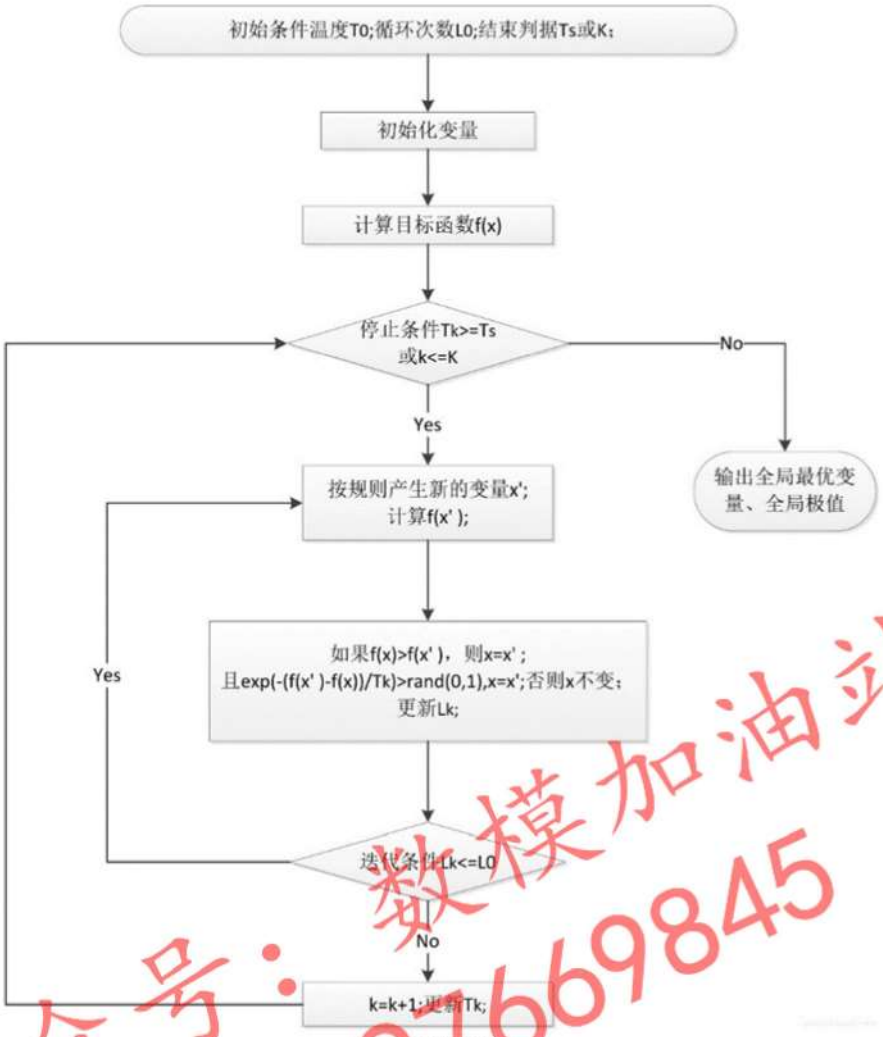

图7 算法流程图

5.2.5 遗传算法

遗传算法（Genetic Algorithm，GA）是一种基于自然选择和遗传学思想的适应性启发式搜索算法。针对问题的最佳解是由多个要素组成的，要素间的新组合会接近于问题的最优解。该算法模拟了自然选择和遗传中发生的复制、交叉和变异等现象，通过核心交叉算子以找到问题的最佳解决方案。

获取更多数学建模相关资料关注【公众号：数模加油站】2023国赛交流群：897669845

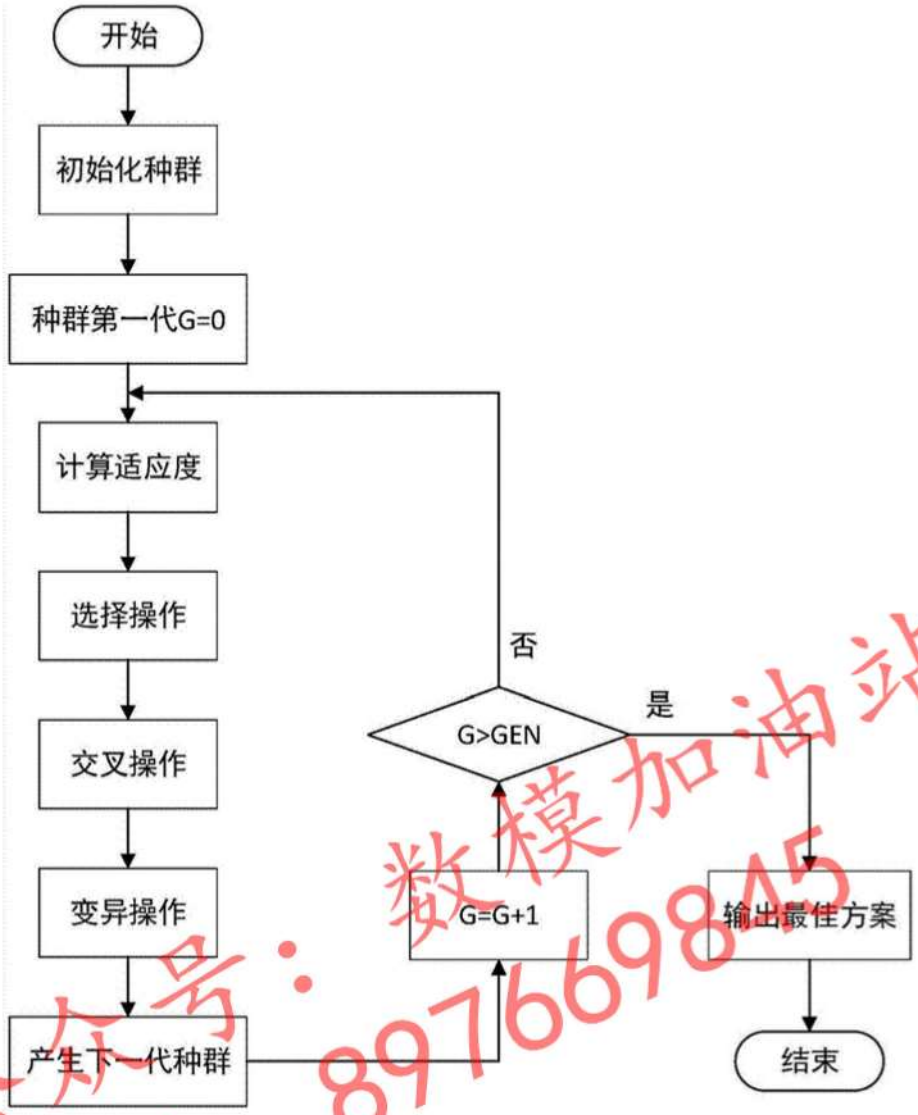

图8 算法流程图

5.2.6 粒子群算法

粒子群优化（Particle swarm optimization，PSO）是一种基于群体智能的随机优化算法。该算法受到一些动物的智能集体行为的启发，通过迭代地尝试给定的质量度量，改进候选解。其核心思想是利用群体中的个体对信息的共享，使整个群体的运动在问题求解空间中，产生从无序到有序的演化过程，从而获得全局的最优解。

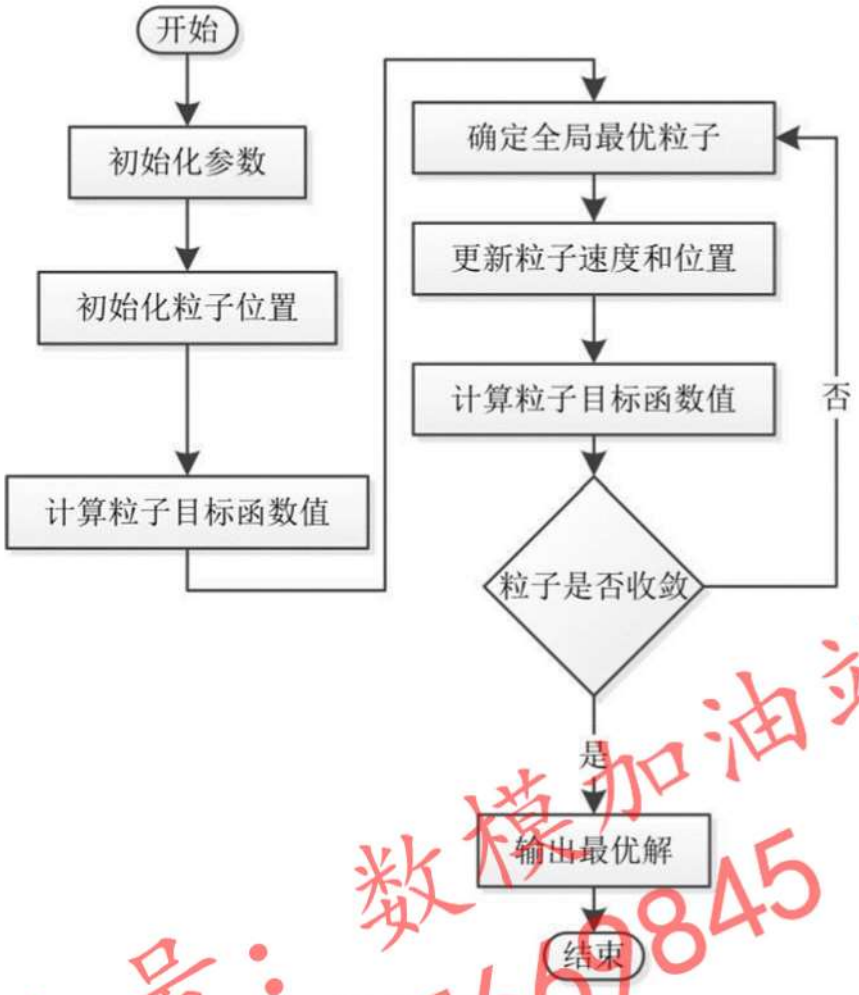

图9 算法流程图

问题二结果：

Result2.xlsx 和 Result3.xlsx

吸收塔x坐标 (m)	吸收塔y坐标 (m)	序号	定日镜宽度	定日镜高度	x坐标 (m)	y坐标 (m)	z坐标 (m)
17.3	12.7	1	3.7	5.27	106.319	11.558	5.27
		2	3.7	5.27	106.831	22.837	5.27
		3	3.7	5.27	102.984	34.364	5.27
		4	3.7	5.27	95.986	45.247	5.27
		5	3.7	5.27	92.554	55.727	5.27
		6	3.7	5.27	86.822	65.059	5.27
		7	3.7	5.27	79.131	73.250	5.27
		8	3.7	5.27	69.806	83.602	5.27
		9	3.7	5.27	61.255	88.591	5.27
		10	3.7	5.27	50.242	93.600	5.27
		11	3.7	5.27	40.365	101.961	5.27
		12	3.7	5.27	28.641	104.904	5.27
		13	3.7	5.27	17.353	105.536	5.27
		14	3.7	5.27	5.803	107.460	5.27
		15	3.7	5.27	-5.725	108.147	5.27
		16	3.7	5.27	-17.415	106.540	5.27
		17	3.7	5.27	-28.942	103.767	5.27
		18	3.7	5.27	-39.974	100.401	5.27
		19	3.7	5.27	-50.817	93.625	5.27
		20	3.7	5.27	-59.428	88.050	5.27
		21	3.7	5.27	-70.654	83.648	5.27

图10 问题二结果（局部）

吸收塔x坐标 (m)	吸收塔y坐标 (m)	序号	定日镜宽度	定日镜高度	x坐标 (m)	y坐标 (m)	z坐标 (m)
13.3	17.9	1	3.503351981	5.928125343	106.319	11.558	5.27
		2	3.291937271	4.958230947	106.831	22.837	5.27
		3	3.654806843	5.635057537	102.984	34.364	5.27
		4	3.712090904	5.448092014	95.986	45.247	5.27
		5	4.134806246	5.201509143	92.554	55.727	5.27
		6	3.966695745	5.083718082	86.822	65.059	5.27
		7	3.873739361	5.520836344	79.131	73.250	5.27
		8	4.155292887	5.589980274	69.806	83.602	5.27
		9	3.736677241	5.332927727	61.255	88.591	5.27
		10	3.827580921	5.920412378	50.242	93.600	5.27
		11	4.016605205	5.620675004	40.365	101.961	5.27
		12	3.367451497	5.780768757	28.641	104.904	5.27
		13	4.018979514	5.693711005	17.353	105.536	5.27
		14	4.153664585	5.053983921	5.803	107.460	5.27
		15	3.725002219	5.829260716	-5.725	108.147	5.27
		16	3.484203292	4.722567111	-17.415	106.540	5.27
		17	3.755319168	5.339729492	-28.942	103.767	5.27
		18	3.344413063	4.94296305	-39.974	100.401	5.27
		19	3.644986177	5.018732207	-50.817	93.625	5.27
		20	4.106114613	5.054771874	-59.428	88.050	5.27

图11 问题三结果（局部）

六、 模型分析检验

6.1 灵敏度分析

6.2 误差分析

七、模型评价与推广

7.1 模型的优点

(1) 模型充分结合实际，简化 xx，xx，xx 条件，考虑了诸多重要因素得到合理的模型，如:xx，xx，xx。这样得到的模型贴合实际，具有较高的应用价值，可以推广到 xx；【模型的假设好】

(2) 模型运用 xx 和 xx 思想，抓住影响 xx 问题的重要因素，将复杂的 xx 问题转化为简单的 xx 问题，合理设置参数，模型的输出结果符合题目要求，能解决实际问题；【模型的参数好】

(3) 本文使用的 xx 算法具有 xx，xx，xx 等优点，对于求解 xx 模型非常适用；【模型的求解算法好】

(4) 本文得到的 xx(安排方案、策略)具有效率高、输出稳定、xx 均衡等特点，基本不存在 xx，xx，xx 等问题，在现有条件下能有效提高生产效率。【模型的结果好】

7.2 模型的不足

(5) 实际应用中，xx 和 xx 可能也是重要的因素，但本文未能考虑到这些因素的影响，一定程度上影响了模型的准确性；【模型不够好】

(6) 本文提出的模型对于现有条件使用效果较好，由于时间问题没有对其他情况进行检验。对于其他情形(如:xx，xx)，可能无法达到较好的效果；【模型适用范围较窄】

(7) 实际上 xx，xx 的影响不一定是线性的，而本文将其作为线性因子处理，忽略了边际效应的影响。【模型的因素有些不好】

7.3 模型的推广

【推广：在 xx 方面，可以将 xx 参数替换成 xx 参数，从而解决 xx 问题；
改进：结合参考文献 xx，进一步考虑 xx 的影响，从而得到更合理的模型；
这部分不用太多，至少 4 行】

获取更多数学建模相关资料关注【公众号：数模加油站】2023国赛交流群：897669845

参考文献

引用别人的成果或其他公开的资料(包括网上查到的资料)必须按照规定的参考文献的表述方式在正文引用处和参考文献中均明确列出。正文引用处用方括号标示参考文献的编号，如[1][3]等；引用书籍还必须指出页码。参考文献按正文中的引用次序列出，其中

书籍的表述方式为：

［编号］作者，书名，出版地：出版社，出版年.

参考文献中期刊杂志论文的表述方式为：

［编号］作者，论文名，杂志名，卷期号：起止页码，出版年.

参考文献中网上资源的表述方式为：

［编号］作者，资源标题，网址，访问时间（年月日）.

[1] 汤国生.基于 SWOT 分析法的大学生数学建模创新实践基地建设探索——以江苏科技大学为例[J].高教学刊,2023,9(11):53-56.DOI:10.19980/j.CN23-1593/G4.2023.11.013.

[2] 谢雨婧,韩惠丽.北师大版初中数学教材中数学建模的多维度分析[J].教学与管理,2023(09):73-76.

[3] 刘志梅.数学建模与高职数学教学的深度融合[J].佳木斯职业学院学报,2023,39(03):152-154.

[4] 许亚桃,吴立宝.基于 Delphi-AHP 高中数学建模教学评价指标体系的研究[J].内江师范学院学报,2023,38(02):113-119.DOI:10.13603/j.cnki.51-1621/z.2023.02.018.

[5] 杨本朝,石雅男,段乾恒,李光松,于刚.大学生数学建模竞赛开展全周期教学实践探究[J].大学教育,2023(04):44-46.

[6] 黄健,徐斌艳.国际视野下数学建模教与学研究的发展趋势——基于第 14 届国际数学教育大会的分析[J].数学教育学报,2023,32(01):93-98.

获取更多数学建模相关资料关注【公众号：数模加油站】2023国赛交流群：897669845

附录

附录 1
基于 xxxx 的第一问求解代码

获取更多数学建模相关资料关注【公众号：数模加油站】2023国赛交流群：897669845